Dr.WEB

Gateway Security Suite (Unix)

Administrator Manual



© Doctor Web, 2026. All rights reserved

This document is intended for information and reference purposes regarding the Dr.Web software discussed herein. This document is not a basis for exhaustive conclusions about the presence or absence of any functional and/or technical features in Dr.Web software and cannot be used to determine whether Dr.Web software meets any requirements, technical specifications and/or parameters, and other third-party documents.

This document is the property of Doctor Web and may be used solely for the personal purposes of the purchaser of the product. No part of this document may be reproduced, published or transmitted in any form or by any means, without proper attribution, for any purpose other than the purchaser's personal use.

Trademarks

Dr.Web, SpIDer Mail, SpIDer Guard, CureIt!, CureNet!, AV-Desk, KATANA and the Dr.WEB logo are trademarks and registered trademarks of Doctor Web in Russia and/or other countries. Other trademarks, registered trademarks and company names used in this document are the property of their respective owners.

Disclaimer

In no event shall Doctor Web and its resellers or distributors be liable for any errors or omissions, or for any loss of profit or any other damage caused or alleged to be caused directly or indirectly by this document, or by the use of or inability to use the information contained in this document.

Dr.Web Gateway Security Suite (Unix)
Version 11.1
Administrator Manual
10/7/2026

For information on regional and international offices of Doctor Web, please visit the official website.

Doctor Web

Doctor Web develops and distributes Dr.Web information security solutions that provide effective protection against malicious software and spam.

Doctor Web customers include home users around the world, government agencies, small businesses, and nationwide corporations.

Since 1992, Dr.Web anti-virus solutions have been known for their continuous excellence in malware detection and compliance with international information security standards.

The state certificates and awards received by Dr.Web solutions, as well as the worldwide use of our products, are the best evidence of exceptional trust in the company products.

We thank all our customers for their support and devotion to Dr.Web products!

Table of Contents

1. Introduction 9

2. Conventions and Abbreviations 10

3. About This Product 11

3.1. Basic Features of Dr.Web Gateway Security Suite 11

3.2. Structure of Dr.Web Gateway Security Suite 13

3.2.1. Basic Anti-Virus Components 14

3.2.2. Threat Search Components 16

3.2.3. Service Components 17

3.2.4. Interface Components 18

3.3. Putting in Quarantine 21

3.4. File Permissions 23

3.5. Operation Modes 24

4. System Requirements and Compatibility 27

5. Licensing 32

6. Installing and Uninstalling 33

6.1. Installing Dr.Web Gateway Security Suite 34

6.1.1. Installing the Universal Package 35

6.1.1.1. Installing from the Command Line 37

6.1.2. Installing from the Repository 37

6.2. Upgrading Dr.Web Gateway Security Suite 41

6.2.1. Getting Current Updates 41

6.2.2. Upgrading to a Newer Version 42

6.2.3. Updating Offline 45

6.3. Uninstalling Dr.Web Gateway Security Suite 46

6.3.1. Uninstalling the Universal Package 46

6.3.1.1. Uninstalling from the Command Line 47

6.3.2. Uninstallation of Dr.Web Gateway Security Suite Installed from the Repository 47

6.4. Additional Information 50

6.4.1. Dr.Web Gateway Security Suite Packages and Files 50

6.4.2. Custom Installation and Uninstallation of Components 53

6.5. Configuring Security Subsystems 59

6.5.1. Configuring SELinux Security Policies 59

6.5.2. Starting in CSE Mode (Astra Linux SE 1.6 and 1.7) 63

7. Getting Started 64
7.1. Registration and Activation 64
7.1.1. Key File 66
7.2. Testing Product Operation 67
7.3. Integration with a Squid Proxy Server 68
7.4. Protecting a Local Web Server 73
7.5. Using SpIDer Gate in Proxy Mode 75
8. Brief Instructions 78
9. Dr.Web Gateway Security Suite Components 83
9.1. Dr.Web ConfigD 83
9.1.1. Operating Principles 83
9.1.2. Command-Line Arguments 85
9.1.3. Configuration Parameters 86
9.2. Dr.Web Ctl 89
9.2.1. Command-Line Call Format 91
9.2.2. General Options 91
9.2.3. Commands 91
9.2.3.1. Anti-Virus Scanning Commands 92
9.2.3.2. Commands to Manage Detected Threats and Quarantine 105
9.2.3.3. Configuration Management Commands 110
9.2.3.4. Advanced Commands 113
9.2.3.4.1. Commands to Manage Updates and Operation in Centralized Protection Mode 113
9.2.3.4.2. Information Commands 115
9.2.4. Usage Examples 120
9.2.5. Configuration Parameters 123
9.3. Dr.Web Management Web Interface 124
9.3.1. Managing the Components 126
9.3.2. Threat Management 127
9.3.3. Managing Settings 128
9.3.3.1. Managing Centralized Protection 134
9.3.4. Scanning Local Files 136
9.4. Dr.Web ICAPD 140
9.4.1. Operating Principles 140
9.4.2. Command-Line Arguments 143
9.4.3. Configuration Parameters 144

9.4.4. HTTP Messages Processing in Lua 159
9.5. SplDer Gate 176
9.5.1. Operating Principles 177
9.5.2. Command-Line Arguments 178
9.5.3. Configuration Parameters 179
9.6. Dr.Web Firewall for Linux 181
9.6.1. Operating Principles 181
9.6.2. Command-Line Arguments 188
9.6.3. Configuration Parameters 189
9.6.4. Processing Connections in Lua 215
9.7. Dr.Web ClamD 223
9.7.1. Operating Principles 223
9.7.2. Command-Line Arguments 223
9.7.3. Configuration Parameters 224
9.7.4. Integration with External Applications 230
9.8. Dr.Web File Checker 232
9.8.1. Operating Principles 232
9.8.2. Command-Line Arguments 233
9.8.3. Configuration Parameters 234
9.9. Dr.Web Network Checker 236
9.9.1. Operating Principles 236
9.9.2. Command-Line Arguments 237
9.9.3. Configuration Parameters 239
9.9.4. Creating a Scanning Cluster 244
9.10. Dr.Web Scanning Engine 247
9.10.1. Operating Principles 247
9.10.2. Command-Line Arguments 248
9.10.3. Configuration Parameters 252
9.11. Dr.Web Updater 255
9.11.1. Operating Principles 255
9.11.2. Command-Line Arguments 256
9.11.3. Configuration Parameters 257
9.12. Dr.Web ES Agent 263
9.12.1. Operating Principles 263
9.12.2. Command-Line Arguments 264
9.12.3. Configuration Parameters 265

9.13. Dr.Web HTTPD **267**
9.13.1. Operating Principles 267
9.13.2. Command-Line Arguments 268
9.13.3. Configuration Parameters 269
9.13.4. Description of the HTTP API 272
9.14. Dr.Web SNMPD **293**
9.14.1. Operating Principles 293
9.14.2. Command-Line Arguments 294
9.14.3. Configuration Parameters 295
9.14.4. Integration with Monitoring Systems 299
9.14.5. Dr.Web SNMP MIB 308
9.15. Dr.Web MeshD **331**
9.15.1. Operating Principles 331
9.15.2. Command-Line Arguments 333
9.15.3. Configuration Parameters 334
9.16. Dr.Web URL Checker **337**
9.16.1. Operating Principles 338
9.16.2. Command-Line Arguments 338
9.16.3. Configuration Parameters 339
9.17. Dr.Web CloudD **341**
9.18. Dr.Web LookupD **346**
9.18.1. Operating Principles 346
9.18.2. Command-Line Arguments 347
9.18.3. Configuration Parameters 348
9.19. Dr.Web StatD **363**
9.19.1. Operating Principles 363
9.19.2. Command-Line Arguments 363
9.19.3. Configuration Parameters 364
10. Appendices **366**
10.1. Appendix A. Types of Computer Threats **366**
10.2. Appendix B. Neutralizing Computer Threats **370**
10.3. Appendix C. Technical Support **372**
10.4. Appendix D. Dr.Web Gateway Security Suite Configuration File **375**
10.4.1. File Structure 375
10.4.2. Parameter Types 376
10.4.3. Traffic Monitoring Rules 379

10.5. Appendix E. Generating SSL Certificates 399
10.6. Appendix F. Known Errors 402
10.7. Appendix G. List of Abbreviations 452

1. Introduction

The Dr.Web Gateway Security Suite solution offers reliable protection of your server from distribution of all possible types of computer threats by using the most advanced threat detection and neutralization technologies. This improves the quality of services provided by the server.

This manual is intended to help administrators of the servers running Unix-like OSes such as GNU/Linux and FreeBSD (hereinafter, they will be referred to as Unix), to install and use Dr.Web Gateway Security Suite.

If the previous version of Dr.Web Gateway Security Suite is already installed on your computer and you wish to upgrade the product to an up-to-date version, follow the steps described in the upgrade procedure (see the Upgrading to a Newer Version section).

2. Conventions and Abbreviations

The following symbols and text conventions are used in this guide:

Convention	Comment
	An important note or instruction
	A warning about possible errors or important notes that require special attention
Anti-virus network	A new term or an emphasis on a term in descriptions
<IP-address>	Placeholders
Save	Names of buttons, windows, menu items and other program interface elements
CTRL	Names of keyboard keys
`/home/user`	Names of files and folders, code examples
Appendix A	Cross-references to document chapters or internal hyperlinks to webpages



Commands entered in the command line using a keyboard (in a terminal or a terminal emulator) are marked with the command prompt character `$` or `#` in the manual. The character indicates the privileges required for running the specified command. According to the standard convention for Unix-based systems:

`$`—command can be run with user rights.

`#`—command must be run with superuser (usually *root*) privileges. To elevate the privileges, use `su` and `sudo` commands.

The list of abbreviations is provided in the Appendix G. List of Abbreviations section.

3. About This Product

Function

Dr.Web Gateway Security Suite is designed to protect network traffic of physical and virtual internet gateways and/or proxy running Unix-like OSes (GNU/Linux and FreeBSD) from viruses and other types of malware, as well as to prevent distribution of threats for various platforms. When operating in centralized protection mode, Dr.Web Gateway Security Suite is controlled by a server managed by Dr.Web Enterprise Security Suite.

Data protection

Main components (scan engine and virus databases) are not only highly effective and resource-sparing, but also cross-platform, which lets Doctor Web experts create reliable anti-virus solutions protecting computers running popular operating systems from threats that target various platforms. Currently, along with Dr.Web Gateway Security Suite, Doctor Web offers other anti-virus solutions for Unix-like operating systems (GNU/Linux and FreeBSD), macOS and Windows. Moreover, other anti-virus products have been developed to deliver protection for devices running Android and Aurora OSes.

Components of Dr.Web Gateway Security Suite are constantly updated, and virus databases and databases of web resource categories are regularly supplemented with new records to ensure up-to-date protection of network traffic of physical and virtual internet gateways and/or proxy. For additional protection against unknown malware, heuristic analysis methods implemented by the anti-virus engine are used. The product also uses the Dr.Web Cloud service that stores information about the latest threats, signatures of which are absent in databases.

Additional information:

- Basic Features of Dr.Web Gateway Security Suite
- Structure of Dr.Web Gateway Security Suite
- Putting in Quarantine
- File Permissions
- Operation Modes

3.1. Basic Features of Dr.Web Gateway Security Suite

1. **Detection and neutralization of threats.** Scanning for malicious programs of any kind (various viruses, including those that infect mail files and boot records, trojans, email worms and so on) and unwanted software (adware, joke programs and dialers). For details on threat types, refer to Appendix A. Types of Computer Threats.

   Threat detection methods:

- a *signature analysis*—a scan method allowing to detect known threats registered in virus databases;
- a *heuristic analysis*—a set of scan methods allowing to detect threats that are not known yet;
- *cloud-based threat detection technologies* using the Dr.Web Cloud service that collects up-to-date information about recent threats detected by various Dr.Web anti-virus products.



The heuristic analyzer may cause false-positive detections. Thus, objects that contain threats detected by the analyzer are considered "suspicious". It is recommended that you quarantine such files and send them for analysis to the Doctor Web anti-virus laboratory. For details on methods used to neutralize threats, refer to Appendix B. Neutralizing Computer Threats.

When scanning the file system on the user request, it is possible to perform either a full scan of all the file system objects available to the user, or a custom scan of the specified objects only (individual directories or files that meet the specified criteria). In addition, it is possible to perform an individual check of boot records of volumes and executable files which started the processes that are currently active in the system. In the latter case, when a threat is detected, a malicious executable file is not only neutralized, but all processes started by it are forcibly terminated. On systems that implement a mandatory model of file access with a set of different access levels, the scanning of files that are not available at the current access level can be done in special standalone instance mode.

All objects containing threats detected in the file system are registered in a permanent threat registry, except those threats that were detected in standalone instance mode.

The Dr.Web Ctl command-line tool included in Dr.Web Gateway Security Suite allows to scan file systems of remote network hosts providing remote terminal access via SSH or Telnet for threats.



Remote scanning can be used only for detection of malicious or suspicious files on a remote host. To eliminate detected threats on the remote host, use administration tools provided directly by this host. For example, update firmware on routers and other "smart" devices, connect to computing machines (including in remote terminal mode) and perform corresponding operations in their file system (deleting or moving files, and so on), or run anti-virus software installed on them.

2. **Analyzing data transmitted to the internet**. Not only user requests are monitored (that is, attempts to connect to a web server and upload a file to it), but also data sent by web servers in response to user requests. To analyze requests and return data, Dr.Web Gateway Security Suite connects via ICAP as an external filter to a proxy server processing HTTP connections of local network users. Furthermore, using the SpIDer Gate component, it is possible to utilize barrier functions that prevent receiving and sending infected files by a public web server of the organization (*this option is available only on* GNU/Linux). To restrict access to unwanted websites, the product uses an automatically updated database of web resources separated into categories, which is included in Dr.Web Gateway Security Suite, and white and black lists created by a system administrator manually. The product also makes a

request to the Dr.Web Cloud service to check whether an internet resource is marked as malicious by other Dr.Web products.

3. **Reliable isolation of infected or suspicious objects** detected within the server file system in a special storage known as quarantine to prevent any harm to the system. When quarantined, objects are renamed according to special rules and, if necessary, they can be restored to their original location only on user demand.

   Threats detected by the Dr.Web ICAPD component in HTTP messages are not quarantined on an internet gateway. Instead their load and transfer to a recipient are blocked, and the user is informed of it with a custom HTML page containing a message about blocking.

4. **Automatic update** of the scanning engine, virus databases, databases of web resource categories to maintain the high level of protection against malware.
5. **Collection of statistics** on scans and threat events. Logging detected threats. Sending of notifications of detected threats via SNMP to external monitoring systems and a centralized protection server if Dr.Web Gateway Security Suite operates in centralized protection mode, as well as to the Dr.Web Cloud service.
6. **Operation in centralized protection mode** to implement single security policies adopted within an anti-virus network comprising this server and managed by Dr.Web Enterprise Security Suite. It can be a corporate network, a private network (VPN) or a network of a service provider (for example, an internet service provider).

3.2. Structure of Dr.Web Gateway Security Suite

Dr.Web Gateway Security Suite is a software suite consisting of a set of components, where each component has its own set of functions. The components are separated into the following categories according to their objectives:

- Basic anti-virus components that form the Dr.Web Gateway Security Suite core. In the absence of the components under this category, the product cannot scan files (and other data) for viruses and other threats.
- Threat search components. These components are used to solve Dr.Web Gateway Security Suite basic tasks—detecting threats and potentially dangerous objects. In their operation the components falling under this category use basic anti-virus components.
- Service components that complete auxiliary tasks to maintain anti-virus protection (updating anti-virus databases, connecting to centralized protection servers, managing general Dr.Web Gateway Security Suite operation and so on).
- Interface components that provide (the user or third-party applications) with the Dr.Web Gateway Security Suite management interface.

Below is the list of Dr.Web Gateway Security Suite components.

3.2.1. Basic Anti-Virus Components

Component	Description
Dr.Web Virus-Finding Engine	Anti-virus engine. Implements algorithms to detect viruses and other malware (by using signature and heuristic analyses). Managed by the Dr.Web Scanning Engine component Library file: `drweb32.dll`. Logged internal name: `CoreEngine`
Dr.Web Scanning Engine	Scanning engine. This component loads Dr.Web Virus-Finding Engine and virus databases. • Passes the contents of files and boot records to the anti-virus engine for scanning. • Manages a queue of the files to be scanned. • Cures threats to which this action is applicable. Operates under the control of the Dr.Web ConfigD daemon or in standalone mode. Used by the Dr.Web File Checker and Dr.Web Network Checker components. Also can be used by the Dr.Web MeshD component (in particular modes) and by external (in relation to Dr.Web Gateway Security Suite) applications using directly the Dr.Web Scanning Engine API Executable file: `drweb-se`. Logged internal name: `ScanEngine`
Virus databases	Automatically updated database of signatures of viruses and other threats, as well as of malware detection and neutralization algorithms. Used by Dr.Web Virus-Finding Engine and is supplied with it
Databases of web resource categories	Automatically updated database containing a list of categorized web resources and being used to identify unwanted websites. Used by components that scan network activity of users and applications, such as SpIDer Gate, Dr.Web ICAPD
Dr.Web File Checker	Component for scanning file system objects and a quarantine manager. • Receives tasks from the threat scanning component on scanning files in the local (relative to Dr.Web Scanning Engine) file system. • Surfs the file system directories according to the task, sends files for scanning to Dr.Web Scanning Engine and notifies the client components about the scanning progress. • Deletes infected files, puts them in and restores them from quarantine, manages quarantine directories.

Component	Description
	• Builds the cache and keeps it up-to-date. The cache contains information about previously scanned files to reduce the frequency of rescanning files. Used by components that scan file system objects Executable file: `drweb-filecheck`. Logged internal name: `FileCheck`
Dr.Web Network Checker	Network data scanning agent. • Used to send data to the scanning engine for actual scanning. The data is sent by components of the product via the network (such components as Dr.Web ClamD, SpIDer Gate, Dr.Web ICAPD). • Allows Dr.Web Gateway Security Suite to manage distributed file scanning: to receive/transmit files for scanning from/to remote hosts. For that purpose, remote hosts must feature an installed and running Dr.Web for Unix operating systems. In the distributed scanning mode, it allows automatic distribution of scanning load among available hosts by reducing load on hosts with a large number of scanning tasks (for example, on mail servers and internet gateways). If the network contains partner hosts that can receive data for scanning, the components that use Dr.Web Network Checker for scanning may operate without local Dr.Web Scanning Engine. Thus, local Dr.Web Scanning Engine, Dr.Web Virus-Finding Engine and virus databases may be absent. For security reasons, files are transmitted over the network using SSL Executable file: `drweb-netcheck`. Logged internal name: `NetCheck`
Dr.Web URL Checker	Component designed to check URLs for belonging to unwanted or potentially dangerous categories Executable file: `drweb-urlcheck`. Logged internal name: `UrlCheck`
Dr.Web MeshD	Component that connects Dr.Web Gateway Security Suite to a local cloud, which allows Dr.Web for Unix products to exchange updates, results of file scanning, transmit files to each other for scanning, as well as to provide scanning engine services directly. If this component is included in the product and the local cloud to which it is connected contains hosts providing scanning engine services, local Dr.Web Scanning Engine, Dr.Web Virus-Finding Engine and virus databases may be absent Executable file: `drweb-meshd`. Logged internal name: `MeshD`

3.2.2. Threat Search Components

Component	Description
Dr.Web ICAPD	ICAP server analyzing requests and network traffic passing through HTTP proxies supporting ICAP (such as Squid). It prevents transferring infected files and accessing network hosts belonging to unwanted categories of internet resources or added to black lists created by the system administrator. If the access to external servers is forbidden or transmitted data contains a threat, the proxy server is instructed to return to the user a special page informing that it is impossible to access the requested resource or download the infected file. Uses Dr.Web Network Checker to scan data received from a proxy server Executable file: `drweb-icapd`. Logged internal name: `ICAPD`
SpIDer Gate	Component for monitoring network traffic and URLs. It is designed to scan data downloaded from the network to the local host and passed from it to the external network for threats. The component also prevents connections with the network hosts added to the unwanted categories of web resources or black lists created by the system administrator. Uses the Dr.Web Network Checker component to scan received data. If allowed by the user, sends requested URLs to the Dr.Web Cloud service for scanning. The component is supplied only with the distributions designed for GNU/Linux OSes. Executable file: `drweb-gated`. Logged internal name: `GateD`
Dr.Web Firewall for Linux	Network connection monitor. Used by SpIDer Gate and provides connection routing for applications that operate on a host to scan traffic of these connections. The component is supplied only with the distributions designed for GNU/Linux OSes. Executable file: `drweb-firewall`. Logged internal name: `LinuxFirewall`

3.2.3. Service Components

Component	Description
Dr.Web CloudD	Dr.Web Cloud interaction component. Sends URLs visited by the user and information about the scanned files to Dr.Web Cloud to scan them for threats not yet covered by virus databases Executable file: `drweb-cloudd`. Logged internal name: `CloudD`
Dr.Web ConfigD	Dr.Web Gateway Security Suite configuration daemon. • Starts and stops other product components depending on the settings. • Restarts components if a failure in their operation occurs. Starts components at the request of other components. Informs active components when another component starts or shuts down. • Stores information about current license keys and settings and provides this information to all components. Receives adjusted settings and license keys from the designated components of Dr.Web Gateway Security Suite. Notifies other components of changes in license keys and settings Executable file: `drweb-configd`. Logged internal name: `ConfigD`
Dr.Web ES Agent	Centralized protection agent. Ensures product operation in the centralized protection and mobile modes. • Provides connection between the product and the centralized protection server, receives a license key file, updates of the virus databases and anti-virus engine. • Sends the information about the Dr.Web Gateway Security Suite components, their status and statistics on threat events to the server Executable file: `drweb-esagent`. Logged internal name: `ESAgent`
Dr.Web LookupD	Component for retrieving data from external data sources. Retrieves data from external data sources (directory services, files, relative databases, and so on) to be used in rules of traffic monitoring Executable file: `drweb-lookupd`. Logged internal name: `LookupD`
Dr.Web StatD	Component for storing Dr.Web Gateway Security Suite component operation events.

Component	Description
	Receives and stores product component events, such as unexpected shutdown, threat detection and so on. Executable file: `drweb-statd`. Logged internal name: `StatD`
Dr.Web Updater	Updating component. Downloads updates of virus databases and databases of web resource categories, the anti-virus engine from Doctor Web servers. The updates can be downloaded automatically, on schedule, and on user demand (via the Dr.Web Ctl utility or management web interface) Executable file: `drweb-update`. Logged internal name: `Update`

3.2.4. Interface Components

Component	Description
Dr.Web HTTPD	Dr.Web Gateway Security Suite component management web server. Provides a custom HTTP API for managing Dr.Web Gateway Security Suite components. This API is used by the management web interface. For security reasons, the component uses HTTPS to connect to the management web interface. Uses Dr.Web Network Checker to send data for scanning to Dr.Web Scanning Engine Executable file: `drweb-httpd`. Logged internal name: `HTTPD`
Dr.Web Management Web Interface	Management Web Interface. Can be accessed using any browser on a local or remote host. The management web interface enables the product not to use third-party web servers, such as Apache HTTP Server, or remote administration tools, such as Webmin. The functionality is ensured by the Dr.Web HTTPD web server
Dr.Web Ctl	Tool designed to manage Dr.Web Gateway Security Suite from the command line. Allows the user to start file scanning, view and manage quarantined objects, start a virus database update procedure, connect Dr.Web Gateway Security Suite to or disconnect it from a centralized protection server, view and change product parameters

Component	Description
	Executable file: `drweb-ctl`. Logged internal name: `Ctl`
Dr.Web SNMPD	SNMP agent. Designed for integration of Dr.Web Gateway Security Suite into external monitoring systems over SNMP. Such integration allows you to monitor the state of the product components and to collect statistics on threat detection and neutralization. Supports SNMP v2c and v3 protocols Executable file: `drweb-snmpd`. Logged internal name: `SNMPD`
Dr.Web ClamD	Component emulating the interface of the `clamd` anti-virus daemon (the component of the ClamAV® anti-virus). Allows all applications supporting ClamAV® to use Dr.Web Gateway Security Suite transparently for anti-virus scanning. Depending on the mode, uses Dr.Web File Checker or Dr.Web Network Checker to pass data for scanning to Dr.Web Scanning Engine Executable file: `drweb-clamd`. Logged internal name: `ClamD`

The figure below shows the structure of Dr.Web Gateway Security Suite and its interaction with external applications.

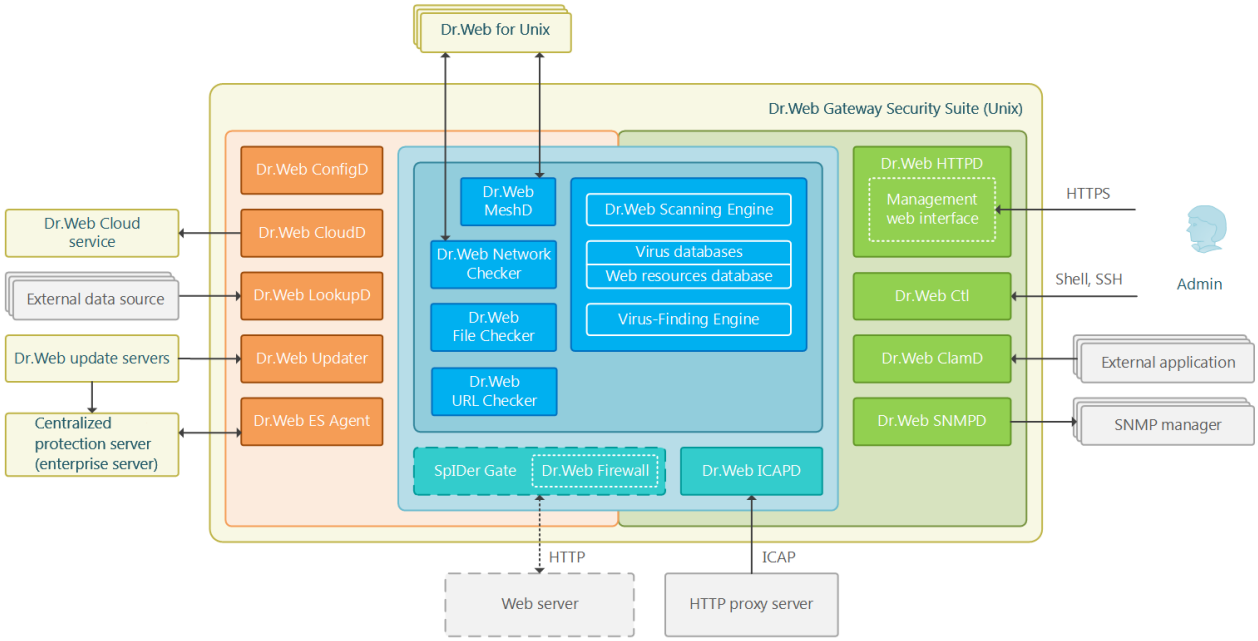


Figure 1. The structure of Dr.Web Gateway Security Suite

In this scheme, the following notations are used:

	Dr.Web Gateway Security Suite as a whole and external Dr.Web applications not distributed with it
	Programs external to Dr.Web Gateway Security Suite and products that integrate with it
	Service components that perform particular anti-virus protection tasks (updating anti-virus databases, connecting to centralized protection servers, overall coordination of operation and so on)
	Interface components that provide (the user or third party applications) with the Dr.Web Gateway Security Suite management interface
	Components used for anti-virus scanning
	Basic anti-virus components that form the Dr.Web Gateway Security Suite core. Used by the components that perform data and file scans

Components marked with a dotted line can be absent depending on the Dr.Web Gateway Security Suite distribution or usage.

For details on Dr.Web Gateway Security Suite components, refer to Dr.Web Gateway Security Suite Components.

3.3. Putting in Quarantine

The quarantine of Dr.Web Gateway Security Suite is a system of directories designed to isolate files containing detected threats that cannot be currently cured for some reason. For example, a detected threat can be incurable because Dr.Web Gateway Security Suite is still unaware of it (for example, the threat was detected by the heuristic analyzer, but the virus databases do not cover the threat signature and a method to cure) or curing causes errors. Moreover, a file can be quarantined on user demand if the user selected the corresponding action in the list of detected threats or specified this action in settings as a reaction to threats of a specific type.

When a file is quarantined, it is renamed according to special rules to prevent its identification by users and applications and inhibit accessing it without quarantine management tools implemented in Dr.Web Gateway Security Suite. Moreover, when a file is quarantined, its execution bit is always reset to prevent running this file.

Quarantine directories are located in:

- *user home directory* (if multiple user accounts exist on the computer, a separate quarantine directory can be created for each of the users);
- *root directory of each logical volume* mounted on the file system.

Dr.Web Gateway Security Suite quarantine directories are always named `.com.drweb.quarantine` and are not created until the "**Quarantine**" (`QUARANTINE`) action is applied, that is, quarantine directories are not created until a threat is detected. At that, only a directory required for isolation of the file is created. When selecting the directory, the name of the file owner is used. Search is performed upwards from the directory containing the file to the file system root `/`; if the home directory of the owner is reached, the file is isolated in the quarantine directory under the home directory. Otherwise, the file is isolated in the quarantine directory created under the volume root directory (which is not always the same as the file system root directory). Thus, any infected file put in quarantine is always kept on the same volume, which provides for correct operation of quarantine in case there are removable data storage devices and other volumes that can be mounted in the file system occasionally and on different mount points.

A user can manage quarantine contents in command-line mode using the Dr.Web Ctl utility, or via the management web interface. At that, all currently available quarantine directories containing isolated objects are always processed as a single entity. From the point of view of a user who views the contents of the combined quarantine, the quarantine directory located in the home directory is a *User* quarantine, and all other quarantine directories are a *System* quarantine.

You can manage quarantine even if no active license was found; however, isolated objects cannot be cured in this case.

Not all anti-virus components of Dr.Web Gateway Security Suite use quarantine for threat isolation. For example, it is not used by Dr.Web ClamD and Dr.Web ICAPD components.

3.4. File Permissions

To scan objects of the file system and neutralize threats, Dr.Web Gateway Security Suite (or rather the user under whom it runs) requires the following permissions:

Action	Required rights
Listing all detected threats	Unrestricted. No special permission required.
Output of container contents (an archive, email file, and so on) (display only corrupted or malicious elements)	Unrestricted. No special permission required.
Moving to quarantine	Unrestricted. The user can quarantine all infected files regardless of read or write permissions on them.
Deleting threats	The user needs to have write permissions for the file that is being deleted. If a threat is detected in a file inside a container (an archive, an email message and so on), the container is quarantined and not deleted.
Curing	Unrestricted. The access permissions and owner of a cured file remain the same after curing. The file can be removed if deletion can cure the detected threat.
Restoring a file from quarantine	The user should have permissions to read the file and to write to the restore directory.
Deleting a file from quarantine	The user must possess write permissions to the file that was moved to quarantine.

To start the Dr.Web Ctl command-line management tool with superuser privileges (usually *root*), use the `su` command to change the user or the `sudo` command to run the command as another user.

3.5. Operation Modes

In this section

- Centralized protection concept
- Connecting to a centralized protection server
- Disconnecting from a centralized protection server

Dr.Web Gateway Security Suite can operate either in standalone mode or as a part of a corporate or private *anti-virus network* managed by a *centralized protection server*. Such operation mode is called a *centralized protection mode*. Using this mode does not require installation of additional software or Dr.Web Gateway Security Suite re-installation or uninstallation.

- In *standalone mode*, a protected computer is not connected to the anti-virus network and its operation is managed locally. In this mode, configuration and license key files are located on local disks and Dr.Web Gateway Security Suite is fully managed by the protected computer. Updates for virus databases are received from Doctor Web update servers.
- In *centralized protection mode*, protection of the computer is managed by the centralized protection server. In this mode, some functions and settings of Dr.Web Gateway Security Suite can be adjusted in accordance with the general (corporate) anti-virus protection policy implemented on the anti-virus network. The license key file used for operating in centralized protection mode is received from the centralized protection server to which Dr.Web Gateway Security Suite is connected. The license or demo key file stored on the local computer, if any, is not used. Statistics on threat events together with information on Dr.Web Gateway Security Suite operation are sent to the centralized protection server. Updates for virus databases are also received from the centralized protection server. Configuring the centralized protection mode is described in the Dr.Web Enterprise Security Suite Administrator Manual for managing Dr.Web Gateway Security Suite.
- In *mobile mode*, Dr.Web Gateway Security Suite receives updates from Doctor Web update servers, but uses settings stored locally and a custom license key file that were received from the centralized protection server.

A possibility of configuring the settings of the SpIDer Guard file system monitor as well as enabling or disabling it while Dr.Web Gateway Security Suite is controlled by the centralized protection server are dependent on permissions specified on the server.

Centralized protection concept

Doctor Web solutions for managing centralized protection use a client-server model (see the figure below).

Corporate computers or computers of clients of an IT service provider are protected by *local anti-virus components* (in this case, by Dr.Web Gateway Security Suite), which ensure anti-virus protection and maintain connection to the centralized protection server.

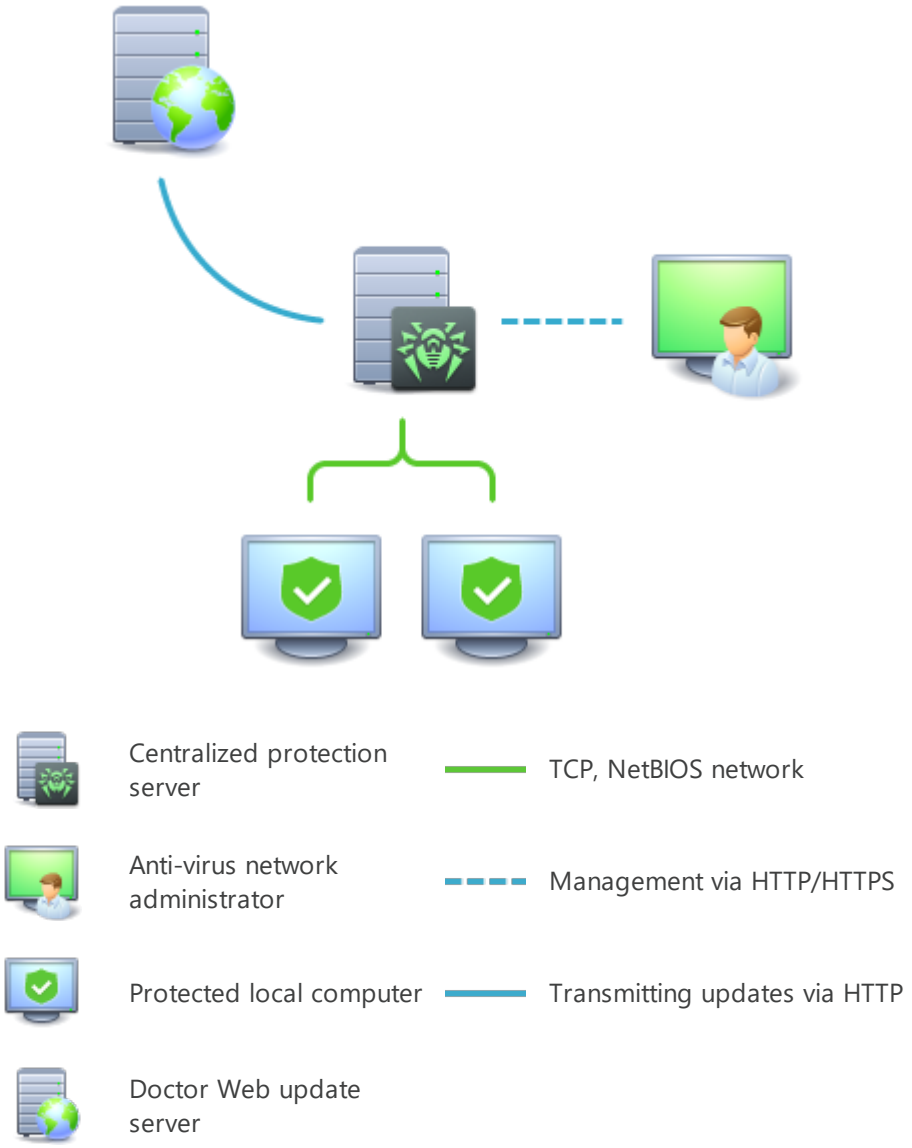


Figure 2. Logical structure of the anti-virus network

Local components are updated and configured from the *centralized protection server*. The entire stream of instructions, data and statistics in the anti-virus network also passes the centralized protection server. The volume of traffic between protected computers and the centralized protection server can be significant, therefore an option for traffic compression is provided. Using encryption while sending data prevents a leak of sensitive data or substitution of software downloaded onto protected computers.

All necessary updates are downloaded to the centralized protection server from Doctor Web update servers.

Changes in the configuration of local anti-virus components and command transfer are performed by anti-virus network administrators using the centralized protection server. The administrators manage configuration of the centralized protection server and topology of the

anti-virus network (for example, they validate connection of a local station to the network) and configure operation of individual local anti-virus components when necessary.



Local anti-virus components are incompatible with anti-virus products of other companies or Dr.Web anti-virus solutions if the latter do not support operation in the centralized protection mode (for example, Dr.Web Anti-virus version 5.0). Installation of two anti-virus programs on the same computer can cause a system crash or a loss of important data.

The centralized protection mode allows exporting and saving Dr.Web Gateway Security Suite operation reports using the centralized protection server. Reports can be exported and saved in the following formats: HTML, CSV, PDF and XML.

Connecting to a centralized protection server

Dr.Web Gateway Security Suite can be connected to the centralized protection server of the anti-virus network using the `esconnect` command of the `drweb-ctl` command-line management tool.



For the verification of the centralized protection server the certificate corresponding to the unique public key of the server is used. By default, the Dr.Web ES Agent centralized protection agent will not allow you to connect to the server unless you specify a certificate file. The certificate file must first be obtained from the administrator of the anti-virus network served by the server to which you want to connect Dr.Web Gateway Security Suite.

If Dr.Web Gateway Security Suite is connected to the centralized protection server, you can switch the product into the mobile mode or switch it back into the centralized protection mode. Switching the mobile mode on or off is accomplished with the help of the `MobileMode` configuration parameter of the Dr.Web ES Agent component.



Dr.Web Gateway Security Suite can switch to the mobile mode only if this is allowed by the settings of the centralized protection server.

Disconnecting from a centralized protection server

Dr.Web Gateway Security Suite can be disconnected from the centralized protection server of the anti-virus network using the `esdisconnect` command of the `drweb-ctl` command-line management tool.

4. System Requirements and Compatibility

In this section

- System requirements
- List of supported operating system versions
  - GNU/Linux
  - FreeBSD
- Required additional packages and components
- Disclaimer
- Supported HTTP proxy servers
- Compatibility with security subsystems

System requirements

You can use Dr.Web Gateway Security Suite on a computer that meets the following requirements:

Parameter	Requirements
Platform	Processors of the following architectures and command systems are supported: • Intel/AMD: 32-bit (*IA-32*, *x86*); 64-bit (*x86-64*, *x64*, *amd64*) • ARM64 • E2K (*Elbrus*) • IBM POWER9, Power10 (*ppc64el*)
RAM	At least 500 MB of free RAM (1 GB or more is recommended)
Free disk space	At least 2 GB of free disk space on a volume where the product directories are located
Operating system	GNU/Linux (based on kernel version 2.6.37 or later, using `glibc` library 2.13 or later, `systemd` initialization system ver. 209 or later), FreeBSD. The supported operating system versions are listed below. The operating system must support the PAM authentication mechanism
Other	The following valid network connections: • valid Internet connection to enable updates for virus databases and Dr.Web components; • when operating in the centralized protection mode, connection to the server on the local network is enough; connection to the Internet is not required



To enable the correct operation of the Dr.Web Firewall for Linux component, the OS kernel must be built with the following options:

- *CONFIG_NETLINK_DIAG, CONFIG_INET_TCP_DIAG;*
- *CONFIG_NF_CONNTRACK_IPV4, CONFIG_NF_CONNTRACK_IPV6, CONFIG_NF_CONNTRACK_EVENTS;*
- *CONFIG_NETFILTER_NETLINK_QUEUE, CONFIG_NETFILTER_NETLINK_QUEUE_CT, CONFIG_NETFILTER_XT_MARK.*

The set of required options from the specified list can depend on the relevant OS version.

To enable the correct operation of Dr.Web Gateway Security Suite, open the following ports:

Purpose	Direction	Port numbers
To receive updates	outgoing	80
To connect to the Dr.Web Cloud service	outgoing	2075 (including those for UDP), 3010 (TCP), 3020 (TCP), 3030 (TCP), 3040 (TCP)

List of supported operating system versions

GNU/Linux

Platform	Supported GNU/Linux versions
x86_64	• ALT 8 SP • ALT Server 9, 10, 11 • ALT Workstation 9, 10, 11 • Astra Linux Common Edition (Orel) 2.12 • Astra Linux Special Edition 1.6 (with cumulative patch 20200722SE16), 1.7, 1.8 • CentOS 7, 8 • Debian 9, 10, 11, 12, 13 • Fedora 37, 38 • GosLinux IC6 • Red Hat Enterprise Linux 7, 8, 9 • RED OS 7.2 MUROM, RED OS 7.3 MUROM, RED OS 8 • SUSE Linux Enterprise Server 12 SP3 • Ubuntu 18.04, 20.04, 22.04, 24.04

Platform	Supported GNU/Linux versions
x86	• ALT 8 SP • ALT Workstation 9, 10 • CentOS 7 • Debian 10
ARM64	• ALT 8 SP • ALT Server 9, 10, 11 • ALT Workstation 9, 10, 11 • Astra Linux Special Edition (Novorossiysk) 4.7 • CentOS 7, 8 • Debian 11, 12 • Ubuntu 18.04
E2K	• ALT 8 SP • ALT Server 10 • ALT Workstation 10 • Astra Linux Special Edition (Leningrad) 8.1 (with cumulative patch 8.120200429SE81) • Elbrus-D MCST 1.4 • GS CS Elbrus 8.32 TVGI.00311-28
ppc64el	• CentOS 8 • Ubuntu 20.04



In ALT 8 SP, Elbrus-D MCST 1.4 and GosLinux IC6 mandatory access control is not supported.

For other GNU/Linux versions that meet the abovementioned requirements, full compatibility with Dr.Web Gateway Security Suite is not guaranteed. If a compatibility issue occurs, contact our technical support.

FreeBSD

Platform	Supported FreeBSD versions
x86	11, 12, 13, 14
x86_64	11, 12, 13, 14, 15

Dr.Web Gateway Security Suite can be installed on FreeBSD only from a universal package.

The correct operation of Dr.Web Gateway Security Suite on FreeBSD 13, 14 and 15 requires that the `compat12x` compatibility package is installed.

Required additional packages and components

Dr.Web Gateway Security Suite does not require installation of additional packages and OS components (except for the protected server software, see below).

For convenient work with Dr.Web Gateway Security Suite from the command line, it is recommended to enable command auto-completion in your command shell (if disabled).

If you encounter any issue with installation of additional packages and components, refer to the documentation of your operating system version.

Disclaimer

- SpIDer Gate *can have conflicts* with other firewalls installed in your operating system (such as Shorewall and SuseFirewall2 in SUSE Linux Enterprise Server and FirewallD in Fedora, CentOS, Red Hat Enterprise Linux). The conflict manifests itself in the SpIDer Gate error message with code x109 or the Dr.Web Firewall for Linux error message with code x102. Methods to resolve the conflict are described in the section Appendix F. Known Errors, see errors x109 and x102.
- If the OS includes NetFilter earlier than 1.4.15, SpIDer Gate can operate incorrectly. This issue is related to the internal error of NetFilter, and looks like as follows: after disabling SpIDer Gate, the network becomes unstable. If you encounter this issue, it is recommended that you upgrade your OS to a version that includes NetFilter 1.4.15 or later. The ways to resolve this issue are described in the Appendix F. Known Errors section.

Supported HTTP proxy servers

For integration with an HTTP proxy server, the installed and configured HTTP proxy server with Squid 3.0 and later is required. Squid must be built with the support of ICAP (compiled with the `--enable-icap-client` option).

In the mode of an internet barrier and a transparent proxy, there are no requirements for web servers and HTTP proxy servers.

Internet barrier and transparent proxy modes can be used only on GNU/Linux.

Compatibility with security subsystems

By default, Dr.Web Gateway Security Suite does not support the SELinux security subsystem. Moreover, Dr.Web Gateway Security Suite operates by default in a reduced functionality mode on the GNU/Linux systems that use mandatory access models (for example, on the systems distributed with the PARSEC mandatory access subsystem, which assigns different privilege levels, so-called mandatory levels, to users and files).

In case of installing Dr.Web Gateway Security Suite on systems with SELinux (as well as on systems that use mandatory access models), it is necessary to additionally configure the security subsystem, so that Dr.Web Gateway Security Suite would operate in a full functionality mode. For details, refer to the Configuring Security Subsystems section.

5. Licensing

The rights to use Dr.Web Gateway Security Suite are granted by a license purchased from the Doctor Web company or from its partners. License parameters determining user rights are set in accordance with the License agreement, which the user accepts during Dr.Web Gateway Security Suite installation. The license contains information on the user and the vendor as well as usage parameters of the purchased product, including:

- a list of components licensed to the user;
- a Dr.Web Gateway Security Suite license period;
- other restrictions (for example, a number of computers on which the purchased copy of Dr.Web Gateway Security Suite is allowed for use).

Each Doctor Web product license has a unique serial number associated with a special file stored on the local computer. This file regulates operation of the Dr.Web Gateway Security Suite components in accordance with the license parameters and is called a *license key file*.

You can activate a *demo period* for evaluation purposes. Upon activating the demo period, you gain the right to use the installed copy of Dr.Web Gateway Security Suite with full functionality for this entire period if its activation requirements are observed. At that, a special key file named a *demo* key file is automatically generated.

If there is no valid license or a demo period is not activated (including cases when a license purchased earlier or a demo period is expired), anti-virus functions of Dr.Web Gateway Security Suite are blocked. Furthermore, the service for receiving updates for Dr.Web virus databases from Doctor Web update servers becomes unavailable. However, you can activate the Dr.Web Gateway Security Suite by connecting it to a centralized protection server as a part of an anti-virus network administered by an enterprise or an internet service provider. In this case, anti-virus functions and updates of the product instance installed on the computer included in the anti-virus network are managed by the centralized protection server.

6. Installing and Uninstalling

This section describes how to install and uninstall Dr.Web Gateway Security Suite, as well as how to obtain current updates and upgrade to a new version, if an earlier version of Dr.Web Gateway Security Suite is already installed on your computer.

Moreover, this section describes the procedure of custom installation and uninstallation of the Dr.Web Gateway Security Suite components (for example, to resolve errors that occurred during the operation of Dr.Web Gateway Security Suite or to install the product with a limited function set) and configuration of advanced security subsystems (such as SELinux) that could be necessary for installation or operation of Dr.Web Gateway Security Suite.

To perform these procedures, superuser (usually the *root* user) privileges are required. To gain superuser privileges, use the `su` command to change the current user or the `sudo` command to run the specified command as another user.



Compatibility of Dr.Web Gateway Security Suite with anti-virus products of other developers is not guaranteed. Due to the fact that the installation of two anti-viruses on one computer can cause errors of the operating system and a loss of important data, it is strongly recommended that you uninstall anti-virus products of other developers before the installation of Dr.Web Gateway Security Suite.

If your computer already has another Dr.Web anti-virus product installed from the universal package (`.run`), and you want to install one more Dr.Web anti-virus product (for example, you have Dr.Web Desktop Security Suite installed from the universal package, and you want to install Dr.Web Gateway Security Suite in addition to it), it is necessary to make sure that the version of the already installed product is the same as the version of Dr.Web Gateway Security Suite to be installed. If the version of the product to be installed is newer than the installed product version, it is necessary, before installation, to upgrade the installed Dr.Web Gateway Security Suite to the version of the product you want to install additionally.

Dr.Web Gateway Security Suite can be installed on FreeBSD only from a universal package.

Additional information:

- Installing Dr.Web Gateway Security Suite
- Upgrading Dr.Web Gateway Security Suite
- Uninstalling Dr.Web Gateway Security Suite
- Configuring Security Subsystems
- Dr.Web Gateway Security Suite Packages and Files
- Custom Installation and Uninstallation of Components

6.1. Installing Dr.Web Gateway Security Suite

To install Dr.Web Gateway Security Suite, do one of the following:

1. Download the installation file, which is a universal package for Unix systems, from the Doctor Web official website. The package is supplied with a console installer; its operation in the graphical desktop mode requires a terminal emulator.
2. Install Dr.Web Gateway Security Suite as a set of native packages (to do that, connect to the corresponding package repository of Doctor Web).



Dr.Web Gateway Security Suite can be installed on FreeBSD only from a universal package.

The correct operation of Dr.Web Gateway Security Suite on FreeBSD 13, 14 and 15 requires that the `compat12x` compatibility package is installed.

It is recommended to install Dr.Web Gateway Security Suite from a universal package on ALT 8 SP and other OSes that use outdated versions of a package manager.

After installing Dr.Web Gateway Security Suite, the IMA/EVM subsystem of ALT 8 SP 11100-01 can issue a warning about potential integrity violation. To avoid this warning, run the command after installing Dr.Web Gateway Security Suite:

```
# integalert fix
```

ALT 8 SP 11100-02 and ALT 8 SP 11100-03 OSes are not subjected to this issue.



After installing Dr.Web Gateway Security Suite using any of the methods provided in this manual, you will need to activate the license or install the key file. In addition, you can connect Dr.Web Gateway Security Suite to a centralized protection server (for details, refer to the Licensing section). Until that, *anti-virus protection is disabled*. In addition, in some cases it is necessary to configure the basic functionality of Dr.Web Gateway Security Suite, as described in the Getting Started section.

During the installation (using either a `.run` universal package or native packages by means of a package manager), messages about Dr.Web Gateway Security Suite installation results are sent at the local email address `root@localhost`.

Dr.Web Gateway Security Suite installed using any of the methods provided in this section can be uninstalled or updated if fixes for its components are available or a new version of Dr.Web Gateway Security Suite is released. If required, you can also configure GNU/Linux security subsystems for correct operation of Dr.Web Gateway Security Suite. If an issue with individual components occurs, you can perform their custom installation and uninstallation without uninstalling Dr.Web Gateway Security Suite.

6.1.1. Installing the Universal Package

The Dr.Web Gateway Security Suite universal package is distributed as an installation file named `drweb-<version>-av-igw-<OS>-<platform>.run`, where <*OS*> is a Unix-like OS, <*platform*> is the platform for which Dr.Web Gateway Security Suite is intended (`x86` for 32-bit platforms, `amd64`, `arm64`, `e2s` and `ppc64el` for 64-bit platforms), for example:

```
drweb-11.1.0-av-igw-linux-amd64.run
```



The name of the installation file corresponding to the above-mentioned format is referred to as <*.run file name*>, and a path to this file—as <*.run file path*> below in this section.

The version of Dr.Web Gateway Security Suite is defined by the first two dot-separated figures in the name of the universal package.

To install Dr.Web Gateway Security Suite components

1. If you do not have the installation file (universal package), download it from the official website of the Doctor Web company.
2. Save the installation file to the hard disk drive of the computer to any writeable directory (for example, `/home/`<*username*>, where <*username*> is the current user name).
3. Allow the file to start, for example, by running the command:

   ```
   # chmod +x <.run file path>
   ```

4. Start it by running the command:

   ```
   # <.run file path>
   ```

   or use a standard file manager of your graphical shell for both changing the file properties (permissions) and running the file.



If you install Dr.Web Gateway Security Suite on the Astra Linux SE OS of versions 1.6 and 1.7 operating in *closed software environment* (CSE) mode, the installer can fail to start because the Doctor Web public key is not on the list of trusted keys. In this case, configure the CSE mode (see Starting in CSE Mode (Astra Linux SE 1.6 and 1.7)) and start the installer again.

First, an integrity check of the archive is run, after which the archived files are unpacked to a temporary directory and the installer is started. If the installer is started without root privileges, it attempts to elevate its privileges asking you for the root password (the `sudo` utility is used). If the attempt fails, the installation process aborts.

If the file system partition containing the temporary directory has not enough free space for the unpacked files, the installation process is aborted and a corresponding message is displayed. In this case, change the value of the `TMPDIR` system environment variable so that it points to a directory located on a partition with enough free space and restart the installation. You can also use the `--target` option to set the target directory (for more details, see the Custom Installation and Uninstallation of Components section).

An installer for a command-line mode runs (to run it in a graphical desktop environment, you need a terminal emulator).

5. Follow the installer instructions.

You can also start the installer in silent mode by running the command:

```
# <.run file path> -- --non-interactive
```

In this case the installer is started in silent mode without displaying program dialogs of the command-line mode.

Using this option means that you accept the terms of the Dr.Web License Agreement. After installing Dr.Web Gateway Security Suite, you can find the License Agreement text in the file `/opt/drweb.com/share/doc/LICENSE` for GNU/Linux or `/usr/local/libexec/drweb.com/share/doc/LICENSE` for FreeBSD. The file extension indicates the language of the License Agreement. The `LICENSE` file (without any extension) stores the Dr.Web License Agreement in English. If you do not accept the terms of the License Agreement, you must uninstall Dr.Web Gateway Security Suite after its installation.

Superuser (usually the *root* user) privileges are required to start the installer in silent mode. To elevate your privileges, use the `su` or `sudo` command.

If the GNU/Linux distribution you use features the SELinux security subsystem, it can interrupt the installation process. If such situation occurs, temporarily set SELinux to the *Permissive* mode. To do this, run the command:

```
# setenforce 0
```

Restart the installer afterwards. When the installation completes, configure SELinux security policies to enable correct operation of the product components.

All unpacked installation files are deleted once the installation process completes.

It is recommended that you save the `.run` file used for the installation to reinstall Dr.Web Gateway Security Suite or its components without the need to update its version.

6.1.1.1. Installing from the Command Line

Once the installation program for the command line starts, the command prompt is displayed on the screen.

1. To start installation, enter `Yes` or `Y` in response to the "Do you want to continue?" question. To exit the installer, enter `No` or `N`. In this case, the installation will be canceled.
2. After that, you need to view the terms of the Doctor Web License Agreement displayed on the screen. Press ENTER to scroll the text down one line or SPACEBAR to scroll down one screen.

   There are no options to scroll the License Agreement up.

3. Once you have read the License agreement, you are prompted to accept its terms. Enter `Yes` or `Y` if you accept the License agreement. If you refuse to accept it, enter `No` or `N`. In the latter case, the installer exits.
4. After you accept the terms of the License Agreement, the installation of the Dr.Web Gateway Security Suite components automatically starts. During the procedure, the information about the installation process (installation log), including the list of installed components, is displayed on the screen.
5. If installation completes successfully, a message is displayed that informs you on how to manage Dr.Web Gateway Security Suite operation.

If an error occurs, a message describing the error is displayed on the screen, and then the installer exits. When the installation process fails due to an error, resolve the issues that caused this error and start the installation again.

6.1.2. Installing from the Repository

Native packages of Dr.Web Gateway Security Suite are stored in the Dr.Web official repository. Once you have added the Dr.Web repository to the list of those used by your operating system package manager, you can install the product from native packages the same way you install any other programs from the operating system repositories. Required dependencies are automatically resolved. Futhermore, in case of installing Dr.Web Gateway Security Suite from the connected repository your OS package manager detects updates for all Dr.Web components and suggests installing them.

To access the Dr.Web repository, internet access is required.

All the commands mentioned below, which are used to add repositories, import digital signature keys, install and uninstall packages, must be run with superuser (usually the *root* user) privileges. To elevate your privileges, use the `su` command to change the current user or the `sudo` command to run the specified command as another user.

Dr.Web Gateway Security Suite can be installed on FreeBSD only from a universal package.

Steps below are for the following OSes (package managers):

- Astra Linux, Debian, Mint, Ubuntu (apt);
- ALT (apt-rpm);
- Mageia, OpenMandriva Lx (urpmi);
- Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf);
- SUSE Linux (zypper).

Astra Linux, Debian, Mint, Ubuntu (apt)

1. The repository for these operating systems is protected with the digital signature of Doctor Web. To access the repository, import and add the digital signature key to the package manager storage by running the command:

```
# wget https://repo.drweb.com/drweb/drweb.gpg -O /etc/apt/trusted.gpg.d/drweb.gpg
```

The `apt-key` utility was used earlier to install the digital signature key from Doctor Web. This utility is deprecated and not recommended for usage. Furthermore, OS developers recommend using the `/etc/apt/trusted.gpg.d` directory instead of the `/etc/apt/trusted.gpg` file. For this reason, when you install Dr.Web Gateway Security Suite from the repository, if the signature key is installed to the `/etc/apt/trusted.gpg` directory, the `apt-get update` command (see below) displays the following warning: `Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.` At that, the installation of Dr.Web Gateway Security Suite continues as usual and the product functionality is not affected. To avoid this warning while installing or reinstalling Doctor Web products, install the signature key as indicated above.

2. To add the repository, run the command:

```
# echo "deb https://repo.drweb.com/drweb/debian 11.1 non-free" > /etc/apt/sources.list.d/drweb.list
```

You can complete steps 1 and 2 by downloading and installing a dedicated DEB package.

3. To install Dr.Web Gateway Security Suite from the repository, run the commands:

```
# apt-get update
# apt-get install drweb-internet-gateways
```

You can also use alternative package managers (for example, Synaptic or aptitude) to install the product. Furthermore, it is recommended to use alternative managers, such as aptitude, to solve a package conflict if it occurs.

ALT (apt-rpm)

1. To add the repository, add the following line to the file `/etc/apt/sources.list.d/drweb-apt-rpm-<arch>.list`:

```
rpm http://repo.drweb.com/drweb/altlinux 11.1/<arch> drweb
```

where <*arch*> represents the package architecture in use:

- for the 32-bit version: `i386`;
- for the AMD64 architecture: `x86_64`;
- for the ARM64 architecture: `aarch64`;
- for the E2K architecture: `e2s`;
- for the IBM POWER (ppc64el) architecture: `ppc64le`.

2. Prior to installing Dr.Web Gateway Security Suite on a computer of the E2K architecture running ALT, add the following lines to the `/etc/rpmrc` file:

```
arch_compat: e2kv4: e2s
arch_compat: e2k: e2s
arch_compat: e2s: noarch
```

3. To install Dr.Web Gateway Security Suite from the repository, run the commands:

```
# apt-get update
# apt-get install drweb-internet-gateways
```

You can also use alternative package managers (for example, Synaptic or aptitude) to install the product.

Mageia, OpenMandriva Lx (urpmi)

1. Add the repository by running the command:

```
# urpmi.addmedia drweb https://repo.drweb.com/drweb/linux/11.1/<arch>/
```

where <*arch*> represents the package architecture in use:

- for the 32-bit version: `i386`;
- for the 64-bit version: `x86_64`.

2. To install Dr.Web Gateway Security Suite from the repository, run the command:

```
# urpmi drweb-internet-gateways
```

You can also use alternative package managers (for example, rpmdrake) to install the product.

Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf)

1. Add the `drweb.repo` file with the content provided below to the `/etc/yum.repos.d` directory:

```
[drweb]
name=DrWeb - 11.1
baseurl=https://repo.drweb.com/drweb/linux/11.1/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://repo.drweb.com/drweb/drweb.key
```



If you plan to write the contents indicated above to a file by using such commands as `echo` with redirecting of an output, the `$` character must be escaped: `\$`.

You can complete step 1 by downloading and installing a dedicated RPM package.

2. To install Dr.Web Gateway Security Suite from the repository, run the command:

```
# yum install drweb-internet-gateways
```

On Fedora 22 and later, it is recommended to use the `dnf` manager instead of the `yum` manager, for example:

```
# dnf install drweb-internet-gateways
```

You can also use alternative package managers (for example, PackageKit or Yumex) to install the product.

SUSE Linux (zypper)

1. To add the repository, run the command:

```
# zypper ar https://repo.drweb.com/drweb/linux/11.1/\$basearch/ drweb
```

2. To install Dr.Web Gateway Security Suite from the repository, run the commands:

```
# zypper refresh
# zypper install drweb-internet-gateways
```

You can also use alternative package managers (for example, YaST) to install the product.

6.2. Upgrading Dr.Web Gateway Security Suite

Dr.Web Gateway Security Suite has two update modes:

1. Getting current updates for packages and components of the up-to-date Dr.Web Gateway Security Suite version (usually such updates comprise bug fixes and minor improvements in component operation).
2. Upgrading to a newer version. This updating method is used if Doctor Web has released a new version of Dr.Web Gateway Security Suite with new features.



Dr.Web Gateway Security Suite allows to update virus databases and the anti-virus engine even if the protected server does not have access to the internet.

6.2.1. Getting Current Updates

After the installation of Dr.Web Gateway Security Suite using any method described in the corresponding section, the package manager automatically connects to the Dr.Web package repository:

- If installation was performed from a universal package (a `.run` file) and the system uses DEB packages (for example, Astra Linux, Debian, Mint or Ubuntu), or the system does not have a package manager (FreeBSD), a separate version of the `zypper` package manager is used for operation with Dr.Web packages. It is automatically installed during Dr.Web Gateway Security Suite installation.

  To get and install updated Dr.Web packages with this manager, go to the directory `/opt/drweb.com/bin` for GNU/Linux or `/usr/local/libexec/drweb.com/bin` for FreeBSD and run the commands:

  ```
  # ./zypper refresh
  # ./zypper update
  ```

  On FreeBSD 11.x for the `amd64` architecture, a repository update error can occur while updating using the `zypper` manager. In this case, install the `compat10x-amd64` compatibility package and try again.

  To install the package, run the command:

  ```
  # pkg install compat10x-amd64
  ```

- In all other cases use updating commands of the package manager of your OS, for example:
  - on Red Hat Enterprise Linux or CentOS, use the `yum` command;
  - on Fedora, use the `yum` or `dnf` command;
  - on SUSE Linux, use the `zypper` command;
  - on Mageia or OpenMandriva Lx, use the `urpmi` command;

- on ALT, Astra Linux, Debian, Mint or Ubuntu, use the `apt-get` command.

You can also use alternative package managers developed for your operating system. If necessary, refer to the reference guide for your package manager.

If a new Dr.Web Gateway Security Suite version is released, packages with its components are put into the section of the Dr.Web repository corresponding to the new product version. In this case, updating requires switching the package manager to the new Dr.Web repository section (refer to Upgrading to a Newer Version).

6.2.2. Upgrading to a Newer Version

In this section

- Introductory remarks
- Upgrading previous versions
  - Upgrading by installing a universal package
  - Upgrading from the repository
  - Key file transfer
  - Restoring connection to a centralized protection server

Introductory remarks

Before you upgrade to a new version, make sure that your server meets the system requirements of the new version.

The upgrade procedure for Dr.Web Gateway Security Suite of earlier versions to a newer version is supported. Migrate to the newer version of Dr.Web Gateway Security Suite in the same way the earlier version of Dr.Web Gateway Security Suite was installed:

- If the current Dr.Web Gateway Security Suite version was installed from the repository, an upgrade requires updating program packages from the repository.
- If the current Dr.Web Gateway Security Suite version was installed from the universal package, to upgrade the product, you need to install another universal package that contains a newer version.

To determine how the version of Dr.Web Gateway Security Suite to be updated was installed, check whether the Dr.Web Gateway Security Suite executable directory contains the `uninst.sh` program uninstallation script. If so, the current version of Dr.Web Gateway Security Suite was installed from the universal package; otherwise, it was installed from the repository.

Dr.Web Gateway Security Suite can be installed on FreeBSD only from a universal package.

If you cannot update Dr.Web Gateway Security Suite the same way you installed it initially, uninstall your current version, and then install a new version using any convenient method. Installation and uninstallation procedures for previous versions of Dr.Web Gateway Security Suite are the same as installation and uninstallation methods described in the current manual. For additional information, refer to the Administrator manual for your current version of Dr.Web Gateway Security Suite.

If the version of Dr.Web Gateway Security Suite to be updated is controlled by a centralized protection server, it is recommended that you save the address of this server. For example, to determine the address of the centralized protection server to which Dr.Web Gateway Security Suite of the version later than 6.0.2 is connected, run the command:

```
$ drweb-ctl appinfo
```

In the output provided by this command, from the line like

```
ESAgent; <PID>; RUNNING 1; Connected <address>, on-line
```

save the <*address*> part (which can look like `tcp://`<*IP address*>`:`<*port*>, for example, `tcp://10.20.30.40:1234`). In addition, it is recommended that you save the server certificate file.

In case of any difficulties with finding out the parameters of the current connection, refer to the Administrator Manual for the installed version of Dr.Web Gateway Security Suite and to the administrator of your anti-virus network.

Upgrading previous versions

Upgrading by installing a universal package

Install the new version of Dr.Web Gateway Security Suite from the universal package. If necessary, during the installation you will be prompted to automatically uninstall installed components of the older version of Dr.Web Gateway Security Suite.



If you need to remove the installed version of Dr.Web Gateway Security Suite during the update process, and there are multiple Dr.Web server products installed on your server (for example, Dr.Web Server Security Suite, Dr.Web Mail Security Suite and Dr.Web Gateway Security Suite), you need to mark for removal only the packages listed below in order to keep products that are not to be updated functional (Dr.Web Server Security Suite and Dr.Web Mail Security Suite):

- `drweb-internet-gateways-doc`,
- `drweb-icapd`.

Upgrading from the repository



You cannot upgrade Dr.Web for UNIX Internet Gateways 6.0.2 to a new version from the repository if multiple Dr.Web server products of version 6.0.2 are installed on your server (for example, products for file servers, for mail servers and for internet gateways). In this case, install the new version of Dr.Web Gateway Security Suite on another machine.

To upgrade your current version of Dr.Web Gateway Security Suite installed from the repository of Doctor Web

Depending on the type of packages in use, do the following:

- **In case of using RPM packages (yum, dnf):**
  1. Change the repository in use (switch from the package repository of your current version to the package repository of the new version).

     You can find the name of the repository storing the packages of the new version in the Installing from the Repository section. For details on how to switch between repositories, refer to help guides of your distribution.

  2. Install the new version of Dr.Web Gateway Security Suite from the repository by running the command:

     ```
     # yum update
     ```

     or, if the dnf manager is used (such as on Fedora 22 and later):

     ```
     # dnf update
     ```

     If an error has occurred while updating the packages, uninstall and reinstall Dr.Web Gateway Security Suite. If necessary, see sections Uninstallation of Dr.Web Gateway Security Suite Installed from the Repository and Installing from the Repository (paragraphs related to your OS and package manager).

- **In case of using DEB packages (apt-get)**:
  1. Change the repository in use (switch from the package repository of your current version to the package repository of the new version).
  2. Update the packages of Dr.Web Gateway Security Suite by running the commands:

     ```
     # apt-get update
     # apt-get dist-upgrade
     ```

Key file transfer

Regardless of the selected method to upgrade Dr.Web Gateway Security Suite, your current license key file will be automatically transferred to a proper location required by the new version of Dr.Web Gateway Security Suite.

If any issue occurs during automatic installation of the key file, you can install it manually. The license key file of Dr.Web Gateway Security Suite is stored in the directory `/etc/opt/drweb.com/` for GNU/Linux or `/usr/local/etc/drweb.com/` for FreeBSD. If the valid license key file is lost, contact the Doctor Web technical support.

Restoring connection to a centralized protection server

If possible, your connection to a centralized protection server will be restored automatically after the upgrade (if the version to be upgraded was connected to the centralized protection server). In case the connection has not been automatically restored, to reestablish the connection of the upgraded version of Dr.Web Gateway Security Suite to the anti-virus network, run the command:

```
$ drweb-ctl esconnect <server address> --Certificate <path to server certificate file>
```

In case of any issues with the connection process, contact the administrator of your anti-virus network.

6.2.3. Updating Offline

In highly secure environments where an internet connection is blocked or limited, it is possible to update virus bases and the scanning engine offline. You need to download updates to a computer connected to the internet, copy them to a USB drive or a network drive and then install them on another computer, which is not connected to the internet. The update procedure must be run from the command line.

To get updates

1. Run the command on a computer connected to the internet:

   ```
   $ drweb-ctl update --Path <path to directory to store downloaded updates>
   ```

2. Copy the downloaded updates to a USB drive or a network drive.
3. Mount the network drive or the removable drive on the computer to be updated. If the updates are from the USB drive, run the commands:

   ```
   # mkdir /mnt/usb
   # mount <path to device> /mnt/usb
   ```

4. Install the updates by running the command:

   ```
   $ drweb-ctl update --From /mnt/usb
   ```

6.3. Uninstalling Dr.Web Gateway Security Suite

Depending on the method that you used to install Dr.Web Gateway Security Suite, you can uninstall the product using one of two methods:

- start the uninstaller to uninstall the universal package in command-line mode;
- uninstall the packages installed from the Doctor Web repository using a system package manager.

6.3.1. Uninstalling the Universal Package

Dr.Web Gateway Security Suite that was installed from the universal package can be uninstalled from the command line (if you are using a graphical desktop environment, you will need a terminal emulator).



The uninstaller uninstalls not only Dr.Web Gateway Security Suite, but also *all other* Dr.Web products installed on your computer.

If any other Dr.Web products, besides Dr.Web Gateway Security Suite, are installed on your computer, follow a custom component installation and uninstallation procedure to uninstall Dr.Web Gateway Security Suite only instead of starting the uninstaller.

Uninstalling Dr.Web Gateway Security Suite from the command line

To run the uninstaller from the command line, run the command:

```
# /opt/drweb.com/bin/uninst.sh
```

for GNU/Linux or

```
# /usr/local/libexec/drweb.com/bin/uninst.sh
```

for FreeBSD. The uninstallation procedure of Dr.Web Gateway Security Suite is described in the Uninstalling from the Command Line section.

To start the uninstaller in silent mode without displaying the user interface (including uninstaller dialogs in command-line mode), run the command:

```
# env DRWEB_NON_INTERACTIVE=yes /opt/drweb.com/bin/uninst.sh
```

On ALT 8 SP and OSes using outdated versions of the package manager, you may see the following messages in the process of uninstalling:

```
/etc/init.d/drweb-configd: No such file or directory
```

These messages do not affect the system functioning. The uninstallation procedure is being performed correctly.

6.3.1.1. Uninstalling from the Command Line

Once the command-line uninstaller runs, a prompt to uninstall the product is displayed on the command line.

1. To start uninstallation, enter `Yes` or `Y` in response to the "Do you want to continue?" request. To cancel uninstalling Dr.Web products from your computer, enter `No` or `N`. In this case, the uninstaller will exit.
2. An automatic uninstallation procedure of all installed Dr.Web packages will run after you confirm it. During this procedure, the information about the uninstallation progress will be displayed and logged.
3. Once the process is completed, the uninstaller will exit automatically.

6.3.2. Uninstallation of Dr.Web Gateway Security Suite Installed from the Repository

Steps below are for the following OSes (package managers):

- Astra Linux, Debian, Mint, Ubuntu (apt);
- ALT (apt-rpm);
- Mageia, OpenMandriva Lx (urpme);
- Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf);
- SUSE Linux (zypper).

All commands mentioned below for package uninstallation require superuser (usually the *root* user) privileges. To elevate your privileges, use the `su` command (to change the current user) or the `sudo` command (to run a command as another user).

Astra Linux, Debian, Mint, Ubuntu (apt)

To uninstall the root meta-package of Dr.Web Gateway Security Suite, run the command:

```
# apt-get remove drweb-internet-gateways
```

In this case, other packages that were automatically installed with the root meta-package to resolve its dependencies remain in the system.

To uninstall the root meta-package together with all dependencies, run the command:

```
# apt-get remove drweb-internet-gateways --autoremove
```

In this case, all packages (not only those of Dr.Web Gateway Security Suite) that were automatically installed to resolve dependencies of other packages but are no longer used are uninstalled.

If the root meta-package was already uninstalled, to automatically uninstall all unused packages, run the command:

```
# apt-get autoremove
```

You can also use alternative package managers (for example, Synaptic or aptitude) to uninstall Dr.Web Gateway Security Suite packages.

ALT (apt-rpm)

To uninstall Dr.Web Gateway Security Suite, use the commands for Debian and Ubuntu OSes (see above).

You can also use alternative package managers (for example, Synaptic or aptitude) to uninstall Dr.Web Gateway Security Suite packages.



On ALT 8 SP, the following messages can be displayed in the console during uninstallation:

```
/etc/init.d/drweb-configd: No such file or directory
```

These messages do not affect the functioning of the system. The uninstallation procedure is being performed correctly.

Mageia, OpenMandriva Lx (urpme)

To uninstall the root meta-package of Dr.Web Gateway Security Suite, run the command:

```
# urpme drweb-internet-gateways
```

In this case, other packages that were automatically installed with the root meta-package to resolve its dependencies remain in the system.

To uninstall the root meta-package together with all dependencies, run the command:

```
# urpme --auto-orphans drweb-internet-gateways
```

In this case, all packages (not only those of Dr.Web Gateway Security Suite) that were automatically installed to resolve dependencies of other packages but are no longer used are uninstalled.

You can also use alternative package managers (for example, `rpmdrake`) to uninstall Dr.Web Gateway Security Suite packages.

Red Hat Enterprise Linux, Fedora, CentOS (yum, dnf)

To uninstall all installed Dr.Web packages, run the command (the `*` character must be escaped on some OSes: `\*`):

```
# yum remove drweb*
```

On Fedora 22 and later, it is recommended to use the `dnf` manager instead of the `yum` manager, for example:

```
# dnf remove drweb*
```



These commands uninstall all packages that name starts with `drweb` (the standard name prefix of Dr.Web products). All packages with this prefix will be uninstalled, not only those of Dr.Web Gateway Security Suite.

You can also use alternative package managers (for example, PackageKit or Yumex) to uninstall Dr.Web Gateway Security Suite packages.

SUSE Linux (zypper)

To uninstall the root meta-package of Dr.Web Gateway Security Suite, run the command:

```
# zypper remove drweb-internet-gateways
```

In this case, other packages that were automatically installed with the root meta-package to resolve its dependencies remain in the system.

To uninstall all installed Dr.Web packages, run the command (the `*` character must be escaped on some OSes: `\*`):

```
# zypper remove drweb*
```



This command uninstalls all packages whose name starts with `drweb` (the standard name prefix for Dr.Web products). All packages with this prefix will be uninstalled, not only those of Dr.Web Gateway Security Suite.

You can also use alternative package managers (for example, YaST) to uninstall Dr.Web Gateway Security Suite packages.

6.4. Additional Information

6.4.1. Dr.Web Gateway Security Suite Packages and Files

In this section

- Packages
- Files

Packages

Dr.Web Gateway Security Suite consists of the following packages:

Package	Contents
`drweb-bases`	Virus database files
`drweb-boost`	Boost libraries
`drweb-clamd`	Dr.Web ClamD component files
`drweb-cloudd`	Dr.Web CloudD component files
`drweb-common`	The `drweb.ini` unified configuration file, main libraries, documentation, hierarchy of Dr.Web Gateway Security Suite directories, and a tool for collecting information on product configuration and system environment. During the installation of this package, a user named `drweb` and a group named `drweb` are created
`drweb-configd`	Dr.Web ConfigD component files
`drweb-ctl`	Dr.Web Ctl component files
`drweb-documentation`	Documentation files for Dr.Web for Unix products in the HTML format
`drweb-dws`	Files of a database of web resource categories
`drweb-engine`	Dr.Web Virus-Finding Engine files
`drweb-esagent`	Dr.Web ES Agent component files
`drweb-filecheck`	Dr.Web File Checker component files
`drweb-internet-gateways-doc`	Documentation in the PDF format
`drweb-internet-gateways`	Root meta-package of Dr.Web Gateway Security Suite

Package	Contents
drweb-gated	SpIDer Gate component files
drweb-firewall	Dr.Web Firewall for Linux component files
drweb-httpd	Files of the Dr.Web HTTPD component and of the management web interface (a meta-package)
drweb-httpd-bin	Dr.Web HTTPD component files
drweb-httpd-webconsole	Files of the management web interface
drweb-icu	Libraries for Unicode support and internationalization
drweb-icapd	Dr.Web ICAPD component files
drweb-libs	Files of main libraries
drweb-lookupd	Dr.Web LookupD component files
drweb-lua	Files of the Lua interpreter used by Dr.Web Gateway Security Suite components designed for monitoring network connections
drweb-netcheck	Dr.Web Network Checker component files
drweb-openssl	OpenSSL libraries
drweb-protobuf	Google Protobuf libraries
drweb-se	Dr.Web Scanning Engine component files
drweb-snmpd	Dr.Web SNMPD component files
drweb-update	Dr.Web Updater component files

The Custom Installation and Uninstallation of Components section describes typical sets of the components for custom installation that implement typical tasks of Dr.Web Gateway Security Suite.

Files

Dr.Web Gateway Security Suite files are stored in `/opt`, `/etc` and `/var` directories of the file system tree.

Structure of the directories in use:

Directory	Contents
• For GNU/Linux: `/etc/init.d/` • For FreeBSD: `/usr/local/etc/rc.d/`	The `drweb-configd` management script for the Dr.Web ConfigD configuration management daemon
• For GNU/Linux: `/etc/opt/drweb.com/` • For FreeBSD: `/usr/local/etc/drweb.com/`	The `drweb.ini` configuration file and the `drweb32.key` key file. In addition, it contains:
`certs/`	– files of certificates in use.
• For GNU/Linux: `/opt/drweb.com/` • For FreeBSD: `/usr/local/libexec/drweb.com/`	Main directory of Dr.Web Gateway Security Suite. It contains:
`bin/`	– executable files of all components (except Dr.Web Virus-Finding Engine);
`include/`	– header files of libraries in use;
`lib/`	– libraries in use;
`man/`	– system help files: `man`;
`share/`	– auxiliary product files, including:
`doc/`	▫ Dr.Web Gateway Security Suite documentation (License Agreement and Administrator manual, if the documentation packages are installed),
`drweb-bases/`	▫ files of Dr.Web virus databases (original files supplied during installation),
`scripts/`	▫ auxiliary script files.
• For GNU/Linux: `/var/opt/drweb.com/` • For FreeBSD: `/var/drweb.com/`	Auxiliary and temporary files, including:
`bases/`	– files of Dr.Web virus databases (the relevant updated version);
`cache/`	– cache of updates;
`drl/`	– lists of update servers in use;

Directory	Contents
`dws/`	– files of a database of web resource categories;
`lib/`	– Dr.Web Virus-Finding Engine as a dynamic-link library (`drweb32.dll`) and the settings for operation in the centralized protection mode distribution, is also stored here;
`update/`	– directory for temporarily storing updates during their download.

6.4.2. Custom Installation and Uninstallation of Components

In this section

- Typical component sets for custom installation
- Installation and uninstallation of Dr.Web Gateway Security Suite components installed from the repository
- Installation and uninstallation of Dr.Web Gateway Security Suite components installed from the universal package
  - Unpacking the installation file
  - Custom installation of components
  - Custom uninstallation of components

If necessary, you can choose to install or uninstall only certain Dr.Web Gateway Security Suite components by installing or uninstalling the respective packages. Perform custom installation or uninstallation the same way Dr.Web Gateway Security Suite was installed.

To reinstall a component, you can uninstall it first and then install again.

Typical component sets for custom installation

If you need to install Dr.Web Gateway Security Suite with limited functionality, you can install only component packages that provide the necessary functionality instead of installing a root meta-package from the repository or from the universal package. The packages required to resolve dependencies will be automatically installed. The table below displays component sets designed to resolve typical Dr.Web Gateway Security Suite tasks. The **Packages for Installation** column lists packages required for installation to obtain the specified component set.

Custom component set	Packages for installation	To be installed
Minimal set for console scanning	• `drweb-filecheck`, • `drweb-se`	• Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web File Checker,

Custom component set	Packages for installation	To be installed
		• Dr.Web Scanning Engine, • Dr.Web Updater, • Virus databases
Set for ClamAV emulation (`clamd`)	• `drweb-clamd`, • `drweb-se`	• Dr.Web ClamD, • Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web File Checker, • Dr.Web Network Checker, • Dr.Web Scanning Engine, • Dr.Web Updater, • Virus databases
Set for checking access to websites using a proxy server via the ICAP (without scanning traffic for threats)	• `drweb-icapd`	• Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web ICAPD, • Dr.Web Updater, • Dr.Web URL Checker, • Database of web resource categories
Set for checking access to websites using a proxy server via the ICAP (with scanning traffic for threats)  If anti-virus scanning is to be performed on another server, which will receive data for scanning via Dr.Web Network Checker, the `drweb-se` package is optional.	• `drweb-icapd`, • `drweb-netcheck`, • `drweb-se`	• Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web ICAPD, • Dr.Web Network Checker, • Dr.Web Scanning Engine, * • Dr.Web Updater, • Dr.Web URL Checker, • Database of web resource categories, • Virus databases *
Set for local scanning of HTTP connections  If anti-virus scanning of connections is not required, installation of `drweb-netcheck` and `drweb-se` packages is	• `drweb-dws`, • `drweb-gated`, • `drweb-firewall`, • `drweb-netcheck`, • `drweb-se`	• Dr.Web ConfigD, • Dr.Web Ctl, • Dr.Web Firewall for Linux, • Dr.Web Network Checker, • Dr.Web Scanning Engine, * • Dr.Web Updater, *** • Dr.Web URL Checker, • SpIDer Gate,

Custom component set	Packages for installation	To be installed
optional. If anti-virus scanning is to be performed on another server, which will receive data for scanning via Dr.Web Network Checker, the `drweb-se` package is optional. If there is no need to check whether a URL belongs to a category of unwanted web resources, the `drweb-dws` package is optional.		• Database of web resource categories, ** • Virus databases *

* The component will not be installed if the package `drweb-se` is not installed.
** The component will not be installed if the package `drweb-dws` is not installed.
*** The Dr.Web Updater component will be installed only if virus databases, the database of web resource categories, or the spam filter are installed.

Installation and uninstallation of Dr.Web Gateway Security Suite components installed from the repository

If Dr.Web Gateway Security Suite is installed from a repository, to install or uninstall a component, use a respective command of the package manager of your OS, for example:

1. To remove the Dr.Web Network Checker component (the `drweb-netcheck` package) from Dr.Web Gateway Security Suite installed on CentOS, run the command:

   ```
   # yum remove drweb-netcheck
   ```

2. To add the Dr.Web Network Checker component (the `drweb-netcheck` package) to Dr.Web Gateway Security Suite installed on Ubuntu, run the command:

   ```
   # apt-get install drweb-netcheck
   ```

If necessary, use help on the package manager of your OS.

Installation and uninstallation of Dr.Web Gateway Security Suite components installed from the universal package

If Dr.Web Gateway Security Suite is installed from a universal package and you want to additionally install or reinstall a package of a component, you will need an installation file (with the `.run` extension) that was used to install Dr.Web Gateway Security Suite. If you do not have this file anymore, download it from the Doctor Web company website.

Unpacking the installation file

When you start the `.run` file, you can specify the following command-line parameters:

`--noexec`—unpack Dr.Web Gateway Security Suite installation files instead of starting the installation process;

`--keep`—do not delete Dr.Web Gateway Security Suite installation files and the installation log after the installation;

`--target` <*directory*>—unpack Dr.Web Gateway Security Suite installation files to the specified <*directory*>.

To get a full list of command-line parameters that can be used with the installation file, run the command:

```
$ ./<.run file name> --help
```

For custom installation of Dr.Web Gateway Security Suite components, unpack the contents of the installation file. To do that, run the command:

```
$ ./<.run file name> --noexec --target <directory>
```

Custom installation of components

The `.run` installation file contains packages of all Dr.Web Gateway Security Suite components (in the RPM format) and auxiliary files. Package files of each component are named as follows:

```
<component_name>_<version>~linux_<platform>.rpm
```

where <*version*> is a string that contains the version and date of a package release, and <*platform*> is a platform to run Dr.Web Gateway Security Suite. Names of all Dr.Web Gateway Security Suite component packages start with the `drweb` prefix.

The installation kit includes the `zypper` package manager intended for the installation of packages. For custom installation, use the `installpkg.sh` service script. To do that, firstly unpack the contents of the installation package to any writeable directory.



To install packages, superuser (usually the *root* user) privileges are required. To gain superuser privileges, use the `su` command to change the current user or the `sudo` command to run the specified command as another user.

To install a component package, go to the directory containing the unpacked installation file and run the command from the command line or from a terminal emulator:

```
# ./scripts/installpkg.sh <package_name>
```

For example:

```
# ./scripts/installpkg.sh drweb-netcheck
```

If it is necessary to install Dr.Web Gateway Security Suite in full, start the installation script by running the command:

```
$ ./install.sh
```

Furthermore, you can install all Dr.Web Gateway Security Suite packages (among other things, to install missing or accidentally removed components) by starting the installation of the root meta-package:

```
# ./scripts/installpkg.sh drweb-internet-gateways
```

Custom uninstallation of components

If your OS uses RPM packages, to uninstall a package of a component, run the corresponding uninstallation command of the package manager of your OS:

- on Red Hat Enterprise Linux and CentOS, run the `yum remove` <*package_name*> command;
- on Fedora, run the `yum remove` <*package_name*> or `dnf remove` <*package_name*> command;
- on SUSE Linux, run the `zypper remove` <*package_name*> command;
- on Mageia or OpenMandriva Lx, run the `urpme` <*package_name*> command;
- on ALT, run the `apt-get remove` <*package_name*> command.

For example, on Red Hat Enterprise Linux:

```
# yum remove drweb-netcheck
```

If your OS uses DEB packages, or there is no package manager in your system (FreeBSD), for the custom uninstallation, use the `zypper` package manager supplied with Dr.Web Gateway Security Suite. To do that, go to the directory `/opt/drweb.com/bin` for GNU/Linux or `/usr/local/libexec/drweb.com/bin` for FreeBSD and run the command:

```
# ./zypper remove <package_name>
```

For example:

```
# ./zypper remove drweb-netcheck
```

If you want to uninstall Dr.Web Gateway Security Suite, start the uninstallation script by running the command:

```
# ./uninst.sh
```

To reinstall a component, you can uninstall it first and then install again by starting custom or full installation.

6.5. Configuring Security Subsystems

Presence of the SELinux enhanced security subsystem in the OS as well as the use of mandatory access control systems, such as PARSEC—as opposed to the classical discretionary model used by Unix—causes issues in the operation of Dr.Web Gateway Security Suite when its default settings are used. To ensure correct operation of Dr.Web Gateway Security Suite in this case, it is necessary to make additional changes to the settings of the security subsystem and/or to the settings of Dr.Web Gateway Security Suite.



Configuring the permissions of the PARSEC mandatory access control system for Dr.Web Gateway Security Suite allows its components to bypass the restrictions of the set security policies and to get access to the files that belong to different privilege levels.

Even if you have not configured the permissions of the PARSEC mandatory access control system for Dr.Web Gateway Security Suite, you still will be able to start file scanning directly from the command line. To do this, run the `drweb-ctl` command in standalone mode by indicating the `--Autonomous` parameter at startup.

In standalone mode, it will be possible to scan files that require a level of privileges not higher than the level of the user who started the scanning session. This mode has the following aspects:

- To start in standalone instance mode, you will need a valid key file, operation in centralized protection mode is not supported (it is possible to install the key file exported from a centralized protection server). In this case, even if Dr.Web Gateway Security Suite is connected to the centralized protection server, the standalone instance *does not notify* the centralized protection server of the threats detected in standalone instance mode.
- All supplementary components that support the functioning of the standalone instance will be started on behalf of the current user and will work with a specifically generated configuration file.
- All temporary files and Unix sockets used for interaction of components are created only in a directory with an unique name. This directory is created by the started standalone instance in the directory for temporary files (specified by the `TMPDIR` environment variable).
- Paths to virus databases, anti-virus engine and executable files of service components are set to default values.
- The number of the standalone instances working simultaneously is not limited.
- When the standalone instance is shut down, the set of components maintaining it is also shut down.

6.5.1. Configuring SELinux Security Policies

In this section

- Universal package installation issues

- Dr.Web Gateway Security Suite operation issues

If your GNU/Linux distribution features the SELinux (*Security-Enhanced Linux*) security subsystem, you may need to adjust SELinux security policies to enable correct operation of Dr.Web Gateway Security Suite service components (such as the scanning engine) after their installation.

Universal package installation issues

If SELinux is enabled, the installation of the Dr.Web Gateway Security Suite universal package from the installation file (`.run`) can fail because an attempt to create the *drweb* special user, as which Dr.Web Gateway Security Suite components run, will be blocked.

If installation of Dr.Web Gateway Security Suite from the installation file (`.run`) fails due to inability to create the *drweb* user, check the SELinux operation mode by running the `getenforce` command. The command outputs the current protection mode:

- `Enforcing`—protection is active and a restrictive strategy is used: actions that violate security policies are blocked and registered in the audit log;
- `Permissive`—protection is active but a permissive strategy is used: actions that violate the security policy are only registered in an audit log but not blocked;
- `Disabled`—SELinux is installed but not active.

If SELinux is operating in *Enforcing* mode, temporarily (during the installation of Dr.Web Gateway Security Suite) change its mode to *Permissive*. For that purpose, run the command:

```
# setenforce 0
```

This command (until the next reboot) enables *Permissive* mode for SELinux.



Regardless of the operation mode enabled by running the `setenforce` command, after the restart of the operating system, SELinux returns to the safe operation mode specified in its settings (the file with SELinux settings is usually stored in the `/etc/selinux` directory).

After Dr.Web Gateway Security Suite is successfully installed from the installation file, enable the *Enforcing* mode again before starting and activating the product. For that purpose, run the command:

```
# setenforce 1
```

Dr.Web Gateway Security Suite operation issues

In certain cases when SELinux is running, several Dr.Web Gateway Security Suite components (such as `drweb-se` and `drweb-filecheck`) cannot start, thereby hindering object scanning and file system monitoring. A failure to start these components causes the occurrence of 119 and 120 error messages in the system log managed by the `syslog` service (this log is typically located in the `/var/log/` directory).

When the SELinux security system blocks access, such an event is also output to an audit system log. In general, when the audit daemon is used in the system, the audit log is stored in the `/var/log/audit/audit.log` file. Otherwise, messages about blocked operations are written to the general log file (`/var/log/messages` or `/var/log/syslog`).

If the scanning components of the product do not operate because they are blocked by SELinux, compile special *security policies* for them.



Certain GNU/Linux distributions do not feature the utilities mentioned below. If so, you may need to additionally install them.

To configure SELinux security policies

1. Create a new file with the SELinux policy source code (a `.te` file). This file defines restrictions related to the described module. The policy source code file can be created in one of the following ways:
   1) Using the `audit2allow` utility, which is the simplest method. The utility generates permissive rules from messages on access denial in system log files. You can set to search messages automatically or specify a path to the log file manually.



You can use this method only if Dr.Web Gateway Security Suite components have violated SELinux security policies and these events have been registered in the audit system log. If not, wait for such an incident triggered by Dr.Web Gateway Security Suite to occur or force-create permissive policies by using the `policygentool` utility (see below).

The `audit2allow` utility is contained either in the `policycoreutils-python` package or in the `policycoreutils-devel` package (for Red Hat Enterprise Linux, CentOS, Fedora operating systems, depending on the version) or in the `python-sepolgen` package (for Debian and Ubuntu operating systems).

Example of using `audit2allow`:

```
# grep drweb-se.real /var/log/audit/audit.log | audit2allow -M drweb-se
```

In the given example, the `audit2allow` utility performs a search in the `/var/log/audit/audit.log` file to find access denial messages for the `drweb-se` component.

The utility creates two files: the `drweb-se.te` policy source file and the `drweb-se.pp` policy module ready to install.

If no corresponding incidents are found in the system audit log, the utility returns an error message.

In most cases, you do not need to modify the policy file created by the `audit2allow` utility; thus, it is recommended to go to step 4 for the installation of the `drweb-se.pp` policy module.



The `audit2allow` utility outputs the `semodule` command with all arguments. By copying it to the command line and running it, you complete step 4. Go to step 2 only if you want to modify the security policies that were automatically generated for Dr.Web Gateway Security Suite components.

2) Using the `policygentool` utility. For that purpose, specify the name of the component that you want to be treated differently and the full path to its executable file.



The `policygentool` utility included in the `selinux-policy` package for Red Hat Enterprise Linux and CentOS may not operate correctly. If so, use the `audit2allow` utility.

Example of policy creation using `policygentool`:

- for the `drweb-se` component:

```
# policygentool drweb-se /opt/drweb.com/bin/drweb-se.real
```

- for the `drweb-filecheck` component:

```
# policygentool drweb-filecheck /opt/drweb.com/bin/drweb-filecheck.real
```

You will be prompted to specify several general properties for created the domain. After that, three files that determine the policy will be created (for each of the components):

<module_name>`.te`, *<module_name>*`.fc` and *<module_name>*`.if`.

2. If required, edit the generated policy source file *<module_name>.te*, then use the `checkmodule` utility to create a binary representation (a `.mod` file) of this source file of the local policy.



This command requires the `checkpolicy` package to be installed in the system.

Usage example:

```
# checkmodule -M -m -o drweb-se.mod drweb-se.te
```

3. Create a policy module for installation (a `.pp` file) with the help of the `semodule_package` utility.

   Example:

```
# semodule_package -o drweb-se.pp -m drweb-se.mod
```

4. To install the created policy module, use the `semodule` utility.

   Example:

```
# semodule -i drweb-se.pp
```

For details on SELinux operating principles and configuration, refer to documentation on your GNU/Linux distribution.

6.5.2. Starting in CSE Mode (Astra Linux SE 1.6 and 1.7)

The Astra Linux SE OS supports a special *closed software environment* (CSE) mode. In this mode, applications can be started only if their executable files are signed with a digital signature of a developer whose public key is added to the OS list of trusted keys.

Starting with Astra Linux SE 1.6, the signing mechanism has been changed. You must configure Astra Linux SE 1.6 and 1.7 prior to starting Dr.Web Gateway Security Suite in CSE mode.

To configure Astra Linux SE 1.6 and 1.7 to start Dr.Web Gateway Security Suite in CSE mode

1. Install the `astra-digsig-oldkeys` package, if not installed, using an OS installation disk.
2. Add the public key of the Doctor Web company to the directory `/etc/digsig/keys/legacy/keys` (if the directory is absent, create it):

   ```
   # cp /opt/drweb.com/share/doc/digsig.gost.gpg /etc/digsig/keys/legacy/keys
   ```

3. Run the command:

   ```
   # update-initramfs -k all -u
   ```

4. Restart the operating system.

7. Getting Started

1. Activate Dr.Web Gateway Security Suite.
2. Ensure its proper operation.
3. Integrate Dr.Web Gateway Security Suite with your HTTP proxy server (see the instruction for a Squid proxy server).
4. To protect a local web server from threats from the external network, change the settings of the SpIDer Gate monitor.
5. If you are not using an external proxy server, configure the proxy mode for SpIDer Gate.
6. Check what components are running and enable additional components, which are disabled by default, if you need them for the protection of your server (for example, Dr.Web ClamD or Dr.Web SNMPD).



You may also need to perform other actions apart from enabling the additional components, for example, you may need to adjust their default configuration.

To view the list of installed and running components and their settings, use one of the following:

- Dr.Web Ctl command-line management tool (use the `drweb-ctl appinfo`, `drweb-ctl cfshow` and `drweb-ctl cfset` commands);
- Dr.Web Gateway Security Suite management web interface (by default, you can access it via a web browser at `https://127.0.0.1:4443`).

7.1. Registration and Activation

In this section

- Purchasing and registering license
- Dr.Web Gateway Security Suite activation
  - Obtaining demo license
  - Key file installation
- Repeated registration

Purchasing and registering license

After a license is purchased, updates to product components and virus databases can be regularly downloaded from Doctor Web update servers. Moreover, standard technical support provided by Doctor Web and its partners becomes available.

You can purchase any Dr.Web product as well as obtain a product serial number either from our partners or our online store. For details on license options, visit the official website of the Doctor Web company.

License registration is required to prove that you are a legal user of Dr.Web Gateway Security Suite and activate its anti-virus functions, including regular updates of virus databases. We recommend that you register the product and activate the license once the installation is completed.

Dr.Web Gateway Security Suite activation

A license can be activated on the official website of the Doctor Web company.

To activate or renew the license, you need to enter the serial number. The serial number is supplied with Dr.Web Gateway Security Suite or via email when purchasing or renewing the license online.

Obtaining demo license

You can use the `license` command of the Dr.Web Ctl command-line utility (`drweb-ctl`), which allows you to obtain a demo key file or the license key file corresponding to the serial number of the registered license. A demo period for your copy of Dr.Web Gateway Security Suite can be obtained on the official website of the Doctor Web company. After you select the product and fill the registration form, you will receive an email with a serial number or key file for demo period activation.



Another demo period of Dr.Web Gateway Security Suite for the same computer can be obtained only after a certain time period.

Key file installation

If you have a key file corresponding to a valid license for the product (for example, you have obtained the key file from a vendor by email after registration or Dr.Web Gateway Security Suite is being migrated to another server), you can activate Dr.Web Gateway Security Suite by specifying a path to the key file. You can do that as follows:

- Manually. For that:
  1. Unpack the key file if archived.
  2. Next, do one of the following.
     - Copy the key file to the directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD and rename the file to `drweb32.key`.

- In the Dr.Web Gateway Security Suite configuration file specify the key file path as the `KeyPath` parameter value. In this case the key file name can be different from `drweb32.key`.

3. Reload the Dr.Web Gateway Security Suite configuration by running the command:

```
# drweb-ctl reload
```

to apply changes.

Alternatively, run the command:

```
# drweb-ctl cfset Root.KeyPath <path to key file>
```

In the latter case, it is not required to additionally reload the Dr.Web Gateway Security Suite configuration. The key file will not be copied to the directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD and will remain at its original location.



If the key file is not copied to the directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD, the user becomes responsible for ensuring that the file is protected from corruption or deletion. This installation method is not recommended as the key file can be accidentally deleted (for example, if the directory, where the key file is located, is automatically cleaned up by the system). Remember that if a key file is lost, you can request a new one, but the number of such requests is limited.

Repeated registration

If a key file is lost but the license is not expired, you must register again by providing the personal data you specified during the first registration. You can use a different email address. In this case, the license key file will be sent at the new address.

A license key file can be obtained using the license management command a limited number of times. If that amount has been exceeded, you can confirm the registration of your serial number on the website of the Doctor Web company to receive the key file. The key file is sent to the email that was specified during the first registration.

After the key file is delivered to you by email, you need to install it.

7.1.1. Key File

A key file, which is a special file stored on the local computer, corresponds to the purchased license or activated demo period for Dr.Web Gateway Security Suite. The file regulates product usage rights in accordance with the purchased license or activated demo period.

The key file has `.key` extension and is valid if satisfies the following criteria:

- License or demo period is not expired.
- Demo period or license applies to all anti-virus components required by the product.

- Integrity of the key file is not violated.

If any of the conditions are violated, the license key file becomes invalid.



During Dr.Web Gateway Security Suite operation, the key file must be located in the default directory `/etc/opt/drweb.com` for GNU/Linux or `/usr/local/etc/drweb.com` for FreeBSD under the name `drweb32.key`.

Components of Dr.Web Gateway Security Suite regularly check whether the key file is available and valid. The key file is digitally signed to prevent its editing. So, the edited key file becomes invalid. It is not recommended to open your key file in text editors in order to avoid its accidental corruption.

If no valid key file (license or demo) is found, or if the license is expired, operation of the anti-virus components is blocked until a valid key file is installed.

It is recommended that you keep the license key file until it expires. In this case, there is no need to register the serial number again upon reinstalling Dr.Web Gateway Security Suite or installing it on another server, and you can use the license key file obtained during the initial registration process.



Dr.Web key files are usually packed in a `.zip` archive if sent via email. The archive with a key file to activate Dr.Web Gateway Security Suite is usually named `agent.zip`. If there are *several* archives in the email message, you should use only `agent.zip`. Before installing the key file, you must unpack the archive using any suitable tool and extract the key file to any available directory (for example, to your home directory or to a USB flash drive).

7.2. Testing Product Operation

The *EICAR (European Institute for Computer Anti-Virus Research)* test helps testing operation of anti-virus programs that detect viruses using signatures. This test was designed specifically so that users could test reaction of an installed anti-virus to a threat without putting their computers at risk.

Although the *EICAR* test program is not actually malware, it is treated by the majority of anti-viruses as a virus. Dr.Web anti-virus products report the following upon detection of this “virus”: *EICAR Test File (NOT a Virus!)*. Other anti-viruses alert users in a similar way. The EICAR test program is a 68-byte `.com` file for MS-DOS/Windows that outputs the following message to the console or to a terminal emulator screen when running:

```
EICAR-STANDARD-ANTIVIRUS-TEST-FILE!
```

The test program body contains only text characters that form the following string:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

If you create a text file consisting of the string provided above, the resulting file will be the "virus" program.

If Dr.Web Gateway Security Suite operates correctly, this file must be detected during a file system scan regardless of the scan type and the user must be notified of the detected threat: EICAR Test File (NOT a Virus!).

An example of a command to test operation of Dr.Web Gateway Security Suite using the EICAR test program:

```
$ echo 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' > testfile
&& drweb-ctl scan testfile && rm testfile
```

This command writes the string that represents the body of the EICAR test program to a file named `testfile` created in the current directory, scans the resulting file and removes this file afterwards.



The abovementioned test requires write access to the current directory. In addition, make sure that it does not contain a file named `testfile` (if necessary, change the file name in the command).

If the test "virus" is detected, the following message is displayed:

```
<path to the current directory>/testfile - infected with EICAR Test File (NOT a Virus!)
```

If an error occurs during the test, refer to the description of known errors.

Incoming HTTP traffic can be scanned for threats using two methods.

Via a web browser

1. Open a web browser and go to proxy server settings.
2. Enter the appropriate proxy server settings for ICAP.
3. Visit the webpage. Notification of an infected file will appear in the browser window.

Via console

Make the following request:

```
$ curl -x 127.0.0.1:3128 https://secure.eicar.org/eicar.com
```

In response, a notification of blocking the requested resource will appear.

7.3. Integration with a Squid Proxy Server

In this section

- Configuring Dr.Web ICAPD

- Configuring Squid
  - For Squid 3.2 and later
  - For Squid 3.1
  - For Squid 3.0
- Advanced Squid settings: data size restrictions

Configuring Dr.Web ICAPD

To integrate Dr.Web ICAPD with a Squid HTTP proxy server, you will need to review the current values of parameters in the Dr.Web ICAPD settings section (the `[ICAPD]` section) and change them if necessary:

- In the `ListenAddress` parameter, specify the address of the network socket (<*IP address*>:<*port*>) that will be listened to by Dr.Web ICAPD waiting for connections from an HTTP proxy server (by default, socket `127.0.0.1:1344` is used).
- In the `Block*` settings, enable or disable categories of websites and threat types that Dr.Web ICAPD should block or allow.
- If required, specify the list of websites to be blocked as the value of the `BlackList` parameter. The value of the `WhiteList` parameter specifies the list of websites that should not be blocked.



The `BlackList` parameter takes precedence over `WhiteList`. If the same domain is specified by both parameters, it will be blocked.

- To configure access to websites in depth (on the basis of various conditions), edit the traffic monitoring rules.



The default values of the `UsePreview`, `Use204` and `AllowEarlyResponse` parameters in the Dr.Web ICAPD settings section allow the component to use the corresponding features of the Internet Content Adaptation Protocol (ICAP) (i.e. allow it to use the *ICAP preview* mode, to return the 204 status code not only in *ICAP preview* mode, and to start sending an "early" response before the entire request has been received from the proxy server). It is recommended that you do not change the default values if no issues with HTTP request processing occur.

After the settings are adjusted, reload the Dr.Web Gateway Security Suite configuration by running the command:

```
# drweb-ctl reload
```

You can also restart Dr.Web Gateway Security Suite by restarting the Dr.Web ConfigD configuration management daemon by running the command:

```
# service drweb-configd restart
```

Configuring Squid

To enable the interaction between Squid and Dr.Web ICAPD, edit the `squid.conf` configuration file (usually located in the `/etc/squid/` directory) to enable ICAP. To integrate ICAP with Dr.Web Gateway Security Suite, proceed as follows:

1. Enable the ICAP in the settings of Squid.
2. Register Dr.Web ICAPD as the ICAP service for Squid.
3. Enable the *ICAP preview* mode (optionally).
4. Allow transferring the client data (i.e. the IP address and the user name of a user who has passed authentication at the proxy server) for using in the rules of Dr.Web ICAPD (optionally).
5. Enable the support of persistent connections between Dr.Web ICAPD and Squid (optional; though enabling persistent connections is not necessary, it increases the performance of Squid operating together with Dr.Web ICAPD).



- To make Squid check HTTP requests (*REQMOD*) and HTTP responses (*RESPMOD*) via ICAP, add two ICAP services of the corresponding types.
- To make Squid use Dr.Web ICAPD as an ICAP service, the address and the port specified in `icap_service` must match the address and the port specified by the `ListenAddress` parameter in the Dr.Web ICAPD settings.
- Dr.Web ICAPD will not work with Squid if the `icap_preview_size` parameter value is not `0`.
- The IP address and the username are generated by Squid itself and sent to Dr.Web ICAPD in an ICAP request in `X-Client-Username` and `X-Client-IP` headers. The values of these headers must be encoded by the methods used in Squid by default. The settings of Squid that affect the encoding methods (`icap_client_username_encode` and `icap_client_username_header`) should not be modified.



To integrate with Dr.Web Gateway Security Suite, Squid should be built with ICAP support (that is, compiled with the `--enable-icap-client` option). Otherwise, it is not possible to establish the connection between Squid and Dr.Web ICAPD.

To use HTTPS, Squid must be built with the support of SSL certificates (that is, compiled with the `--with-openssl` and `--enable-ssl-crtd` options). Squid settings must have SSL bumping enabled.

Settings for different Squid versions can differ. This section provides three configuration variants for the following Squid versions: 3.2 and earlier, 3.1 and 3.0. Edit your Squid configuration file according to the examples.
If the lines from the examples below are commented out in your configuration file, uncomment them. If the lines from the examples are absent, add them to your configuration file.

Only the parameters from sections #1 and #2 are obligatory for configuring the interaction between Dr.Web ICAPD and Squid. The parameters from sections #3 and #4 are optional.

For Squid 3.2 and later

```
#1
icap_enable on

#2
icap_service i_req reqmod_precache bypass=0 icap://127.0.0.1:1344/reqmod
icap_service i_res respmod_precache bypass=0 icap://127.0.0.1:1344/respmod

adaptation_access i_req allow all
adaptation_access i_res allow all

#3
icap_preview_enable on
icap_preview_size 0

#4 (in Squid 3.2, the icap_send_client_ip and icap_send_client_username parameters have
been renamed)
adaptation_send_client_ip on
adaptation_send_username on

#5
icap_persistent_connections on
```

For Squid 3.1

```
#1
icap_enable on

#2 (in Squid 3.1, the format used to configure a service has been changed and the
icap_access parameter has been renamed)
icap_service i_req reqmod_precache bypass=0 icap://127.0.0.1:1344/reqmod
icap_service i_res respmod_precache bypass=0 icap://127.0.0.1:1344/respmod

adaptation_access i_req allow all
adaptation_access i_res allow all

#3
icap_preview_enable on
icap_preview_size 0

#4
icap_send_client_ip on
icap_send_client_username on

#5
icap_persistent_connections on
```

For Squid 3.0

```
#1
icap_enable on

#2
icap_service i_req reqmod_precache 0 icap://127.0.0.1:1344/reqmod
icap_service i_res respmod_precache 0 icap://127.0.0.1:1344/respmod

icap_class icapd_class_req i_req
icap_class icapd_class_resp i_res

icap_access icapd_class_req allow all
icap_access icapd_class_resp allow all

#3
icap_preview_enable on
icap_preview_size 0

#4
icap_send_client_ip on
icap_send_client_username on

#5
icap_persistent_connections on
```

Restart Squid after editing the configuration file and saving the changes.

Advanced Squid settings: data size restrictions

If necessary, you can limit the size of data that Squid sends for scanning via ICAP. To do this, specify a data size in bytes or a regular expression in the configuration file the value of the `Content-Length` header, for example:

```
acl <name> rep_header Content-Length ^[0-9]{7,}$
```

In this example the condition <*name*> is observed if the `Content-Length` header value is greater than 999999.

The condition from the example above can be used to allow (`allow`) or deny (`deny`) scanning of the server response via ICAP (replace `all` with the condition name <*name*> in Squid connection parameters).
The following example shows the settings used to *deny* scanning of responses for which the condition <*name*> is observed:

```
#Squid 3.1 and later
adaptation_access i_res deny <name>

#Squid 3.0 and later
icap_access icapd_class_resp deny <name>
```



The `Content-Length` header can be absent in the web server response. In this case, the settings provided above will not be applied.

The detailed information on configuring the restrictions for web traffic monitoring in Squid can be found in the official documentation.

Restart Squid to apply changes.

7.4. Protecting a Local Web Server

In this section

- Configuring redirection of connections
- Configuring scanning

This feature is available only for distributions designed for OSes of the GNU/Linux family.

To protect a web server running on the same host on which Dr.Web Gateway Security Suite is installed, you need to enable scanning of all the incoming traffic received by the web server with Dr.Web Firewall for Linux in the SpIDer Gate monitor.

Configuring redirection of connections

To configure web server protection, specify the following values for `[LinuxFirewall]` section parameters in the configuration file:

Parameter	Required value
`InspectHttp`	`On`
`AutoconfigureIptables`	`Yes`
`AutoconfigureRouting`	`Yes`
`LocalDeliveryMark`	`Auto`
`ClientPacketsMark`	`Auto`
`ServerPacketsMark`	`Auto`
`TproxyListenAddress`	`127.0.0.1:0` If a custom IP address or port is used in Dr.Web Firewall for Linux operation, specify them here
`InputDivertEnable`	`Yes`
`InputDivertNfqueueNumber`	`Auto`

Parameter	Required value
`InputDivertConnectTransparently`	`Yes`

To view or edit Dr.Web Firewall for Linux settings, use:

- Dr.Web Ctl command-line management tool (use the `drweb-ctl cfshow` and `drweb-ctl cfset` commands).

  For example, the command:

  ```
  # drweb-ctl cfset LinuxFirewall.InputDivertEnable Yes
  ```

  configures Dr.Web Firewall for Linux so that the incoming data is scanned by the SpIDer Gate monitor if HTTP is used and the `InspectHttp` parameter value is set to `On`.
- Dr.Web Gateway Security Suite management web interface (by default, you can access it via a web browser at `https://127.0.0.1:4443`).

To scan data transmitted via the secure protocol, HTTPS:

- Enable scanning of the traffic transmitted via SSL/TLS by running the command:

  ```
  # drweb-ctl cfset LinuxFirewall.UnwrapSsl Yes
  ```

  It is recommended to use the `cfset` command of the `drweb-ctl` tool or the management web interface, because in this case the scanning rules depending on this parameter will change automatically.
- Export a certificate to be used by Dr.Web Gateway Security Suite for embedding in secure SSL/TLS channels by running the command:

  ```
  $ drweb-ctl certificate > <cert_name>.pem
  ```

  It is necessary to indicate a name of the file to store the certificate in the PEM format.
- Add the certificate to the system list of trusted certificates and specify it as the trusted certificate for web clients (browsers) and the web server (for details, see the Appendix E. Generating SSL Certificates section).

Configuring scanning

Specify values of the following parameters in the Dr.Web Firewall for Linux section (`[LinuxFirewall]`) of the configuration file:

1. Parameters for scanning transmitted data (`ScanTimeout`, `HeuristicAnalysis`, `PackerMaxLevel`, `ArchiveMaxLevel`, `MailMaxLevel`, `ContainerMaxLevel` and `MaxCompressionRatio`) that limit scanning duration and resource consumption. If detailed configuration is not required, keep default values.
2. `Block*` parameters for blocking unwanted URLs and content.
3. `BlockUnchecked` parameter to define SpIDer Gate reactions in case the received data cannot be scanned.

For more detailed configuration of HTTP message filtering rules (depending on conditions), edit the Lua procedure or the `RuleSet` rules.

After the settings are adjusted, reload the Dr.Web Gateway Security Suite configuration by running the command:

```
# drweb-ctl reload
```

You can also restart Dr.Web Gateway Security Suite by restarting the Dr.Web ConfigD configuration management daemon by running the command:

```
# service drweb-configd restart
```

7.5. Using SpIDer Gate in Proxy Mode

In this section

- Configuring the proxy mode
- Configuring scanning



This feature is available only for distributions designed for OSes of the GNU/Linux family.

If the internet gateway does not have an HTTP proxy server that could communicate with Dr.Web Gateway Security Suite via ICAP or ClamAV protocols (using the Dr.Web ClamD component directly), to protect the local network from threats spreading over the internet, configure Dr.Web Firewall for Linux so that the data passed to the internet gateway with Dr.Web Gateway Security Suite installed on it is scanned by the SpIDer Gate monitor (a transparent proxy mode).

Configuring the proxy mode

To configure web server protection, specify the following values for `[LinuxFirewall]` section parameters in the configuration file:

Parameter	Required value
`InspectHttp`	`On`
`AutoconfigureIptables`	`Yes`
`AutoconfigureRouting`	`Yes`
`LocalDeliveryMark`	`Auto`
`ClientPacketsMark`	`Auto`
`ServerPacketsMark`	`Auto`

Parameter	Required value
TproxyListenAddress	127.0.0.1:0 If a custom IP address or port is used in Dr.Web Firewall for Linux operation, specify them here
ForwardDivertEnable	Yes
ForwardDivertNfqueueNumber	Auto
ForwardDivertConnectTransparently	Yes

To view and change the settings of Dr.Web Firewall for Linux, use the following:

- Dr.Web Ctl command-line management tool (use the `drweb-ctl cfshow` and `drweb-ctl cfset` commands).

  For example, the command:

  ```
  # drweb-ctl cfset LinuxFirewall.ForwardDivertEnable Yes
  ```

  configures Dr.Web Firewall for Linux such that the data passing the host is scanned by SpIDer Gate, if HTTP is used and the `InspectHttp` parameter is set to `On`.
- Dr.Web Gateway Security Suite management web interface (by default, you can access it via a web browser at `https://127.0.0.1:4443`).

To scan data transferred via HTTPS:

- Enable scanning of SSL/TLS traffic:

  ```
  # drweb-ctl cfset LinuxFirewall.UnwrapSsl Yes
  ```

  It is recommended to use the `cfset` command of the `drweb-ctl` tool or the management web interface, because in this case the scanning rules depending on this parameter will change automatically.
- Export a certificate to be used by Dr.Web Gateway Security Suite for embedding in secure SSL/TLS channels by running the command:

  ```
  $ drweb-ctl certificate > <cert_name>.pem
  ```

  It is necessary to indicate a name of the file to store the certificate in the PEM format.
- Add the certificate to the system list of trusted certificates and specify it as a trusted certificate for web clients (browsers) and the web server (for details, see the Appendix E. Generating SSL Certificates section).

Configuring scanning

Specify the following parameters in the Dr.Web Firewall for Linux section (`[LinuxFirewall]`) of the configuration file:

1. Parameters for scanning transferred data (`ScanTimeout`, `HeuristicAnalysis`, `PackerMaxLevel`, `ArchiveMaxLevel`, `MailMaxLevel`, `ContainerMaxLevel` and

`MaxCompressionRatio`), which limit scanning duration and resource consumption. If detailed configuration is not required, keep default values.

2. `Block*` parameters for blocking unwanted URLs and content.
3. `BlockUnchecked` parameter to define SpIDer Gate reactions in case the received data cannot be scanned.

For more detailed configuration of HTTP message filtering rules (depending on conditions), edit the Lua procedure or the `RuleSet` rules.

After the settings are adjusted, reload the Dr.Web Gateway Security Suite configuration by running the command:

```
# drweb-ctl reload
```

You can also restart Dr.Web Gateway Security Suite by restarting the Dr.Web ConfigD configuration management daemon by running the command:

```
# service drweb-configd restart
```

8. Brief Instructions

In this section

- Working with HTTP proxies and web servers:
  - How to connect Dr.Web Gateway Security Suite to Squid
  - How to protect a web server
  - How to configure the proxy mode for SpIDer Gate
- General operation of Dr.Web Gateway Security Suite:
  - How to restart Dr.Web Gateway Security Suite
  - How to connect to a centralized protection server
  - How to disconnect from a centralized protection server
  - How to activate Dr.Web Gateway Security Suite
  - How to upgrade Dr.Web Gateway Security Suite
  - How to add or remove a component of Dr.Web Gateway Security Suite
  - How to manage Dr.Web Gateway Security Suite component operation
  - How to view the log of Dr.Web Gateway Security Suite

How to connect Dr.Web Gateway Security Suite to Squid

Follow the instructions provided in the Integration with a Squid Proxy Server section.

How to protect a web server

Follow the instructions provided in the Protecting a Local Web Server section.

How to configure the proxy mode for SpIDer Gate

Follow the instructions provided in the Using SpIDer Gate in Proxy Mode section.

How to restart Dr.Web Gateway Security Suite

To restart already running Dr.Web Gateway Security Suite, you can use the script that controls the Dr.Web ConfigD configuration management daemon. Starting, stopping, or restarting the daemon will respectively start, stop or restart Dr.Web Gateway Security Suite.

The script that controls the operation of Dr.Web ConfigD is stored in the standard OS directory (`/etc/init.d/` for GNU/Linux or `/usr/local/etc/rc.d/` for FreeBSD) and is named `drweb-configd`. The script has the following parameters:

Parameter	Description
`start`	Start Dr.Web ConfigD if it is not running. Upon starting, Dr.Web ConfigD starts all the required components of Dr.Web Gateway Security Suite.
`stop`	Shut down Dr.Web ConfigD if it is running. Upon shutting down, Dr.Web ConfigD shuts down all the components of Dr.Web Gateway Security Suite.
`restart`	Restart (shut down and start) Dr.Web ConfigD that will shut down and start all Dr.Web Gateway Security Suite components. If Dr.Web ConfigD is not running, the parameter has the same effect as `start`.
`condrestart`	Restart Dr.Web ConfigD only if it is running.
`reload`	Send a HUP signal to Dr.Web ConfigD if it is running. Dr.Web ConfigD forwards this signal to all the components of Dr.Web Gateway Security Suite. The parameter is used to make all Dr.Web Gateway Security Suite components reread their configuration.
`status`	Output the current state of Dr.Web ConfigD to the console.

For example, to restart Dr.Web Gateway Security Suite (or start it, if it is not running) on an OS of the GNU/Linux family, run the command:

```
# /etc/init.d/drweb-configd restart
```

How to connect to a centralized protection server

1. Obtain a centralized protection server address and certificate file from your anti-virus network administrator. You may also need additional parameters such as an identifier and a password for your station or identifiers of the main group and the tariff group.
2. Use the `esconnect` command of the Dr.Web Ctl command-line tool included in Dr.Web Gateway Security Suite.

   To establish a connection, you must use the `--Certificate` parameter by specifying a path to the certificate file of the server. You can additionally enter an identifier of your host ("station" in the terms of the centralized protection server) and a password for authentication on the server, if you know them, by using the `--Login` and `--Password` parameters. If these parameters are specified, connection to the server will be established only if you specify a correct identifier-password pair. If the parameters are not specified, connection to the server will be established only if it is approved for the server (automatically or by the administrator of the anti-virus network, depending on the server settings).

   Moreover, you can use the `--Newbie` parameter (connect as a new user). If this mode is allowed on the server, after this connection is approved, the server automatically generates

a unique identifier-password pair for this host, which will be further used to connect it to the server.

> In this mode the centralized protection server generates a new account for the host even if this host already has another account on the server.

A standard example of the command to connect Dr.Web Gateway Security Suite to the centralized protection server:

```
# drweb-ctl esconnect <server address> --Certificate <path to the server certificate file>
```

After establishing a connection to the centralized protection server, Dr.Web Gateway Security Suite will operate in the centralized protection mode or in the mobile mode, depending on permissions set on the server and the value of the `MobileMode` configuration parameter of the Dr.Web ES Agent component. To force the mobile mode, set this parameter to `On`. For operation in the centralized protection mode, set the parameter to `Off`.

A standard example of the command to switch Dr.Web Gateway Security Suite, which is connected to the centralized protection server, to the mobile mode is as follows:

```
# drweb-ctl cfset ESAgent.MobileMode On
```

> If the centralized protection server in use does not support or does not allow the mobile mode, changing the `MobileMode` parameter value will not switch Dr.Web Gateway Security Suite to the mobile mode.

How to disconnect from a centralized protection server

To disconnect Dr.Web Gateway Security Suite from the centralized protection server and switch to the standalone mode, use the `esdisconnect` command of the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line:

```
# drweb-ctl esdisconnect
```

To use Dr.Web Gateway Security Suite in standalone mode, a valid license key file is required. Otherwise, anti-virus functions of Dr.Web Gateway Security Suite *will be blocked* after switching to the standalone mode.

How to activate Dr.Web Gateway Security Suite

1. Register on the website of the Doctor Web company.
2. You will receive an archive containing a valid license key file at the email address that you specified during the registration (you can also download this archive directly from the website after you have finished the registration).
3. Install the key file.

How to upgrade Dr.Web Gateway Security Suite

Update component versions or upgrade to a new version.



You may be asked to remove the current version of Dr.Web Gateway Security Suite.

How to add or remove a component of Dr.Web Gateway Security Suite

Follow the custom component installation and uninstallation procedure.



When installing and uninstalling a component, other Dr.Web Gateway Security Suite components can be additionally installed or uninstalled to resolve dependencies.

How to manage Dr.Web Gateway Security Suite component operation

To view the status of Dr.Web Gateway Security Suite components and to manage their operation, use:

- The Dr.Web Ctl command-line management tool. Use the `drweb-ctl appinfo`, `drweb-ctl cfshow` and `drweb-ctl cfset` commands. To view the list of available management commands, run the `drweb-ctl --help` command.
- Dr.Web Gateway Security Suite management web interface. By default, you can access it via a web browser at `https://127.0.0.1:4443`.

How to view the log of Dr.Web Gateway Security Suite

By default, the unified log of all Dr.Web Gateway Security Suite components is output to `syslog` (a file used by the system component `syslog` to log messages; the file depends on the system and is located in the `/var/log` directory). Unified log settings are defined in the configuration file in the `[Root]` section (`Log` and `DefaultLogLevel` parameters). `Log` and `LogLevel` parameters are provided for each component in its settings section. They set the log storage location and the logging level of messages output by the component to the log.

Alternatively, run the `drweb-ctl log` command.

To change the logging settings, use the Dr.Web Ctl command-line management tool or the Dr.Web Gateway Security Suite management web interface.

To identify errors, configure the output of the unified log of all components to be stored in a separate file and enable the output of detailed debug information. For that, run the commands:

```
# drweb-ctl cfset Root.Log <log path>
# drweb-ctl cfset Root.DefaultLogLevel DEBUG
```

To reset the unified log settings for all components, run the commands:

```
# drweb-ctl cfset Root.Log -r
# drweb-ctl cfset Root.DefaultLogLevel -r
```

9. Dr.Web Gateway Security Suite Components

This section contains a description of the Dr.Web Gateway Security Suite components. For each of them, you can find information about its functions, operation principles and parameters stored in the configuration file.

9.1. Dr.Web ConfigD

The Dr.Web ConfigD configuration management daemon is the main control component of Dr.Web Gateway Security Suite. It provides centralized storage for settings of all Dr.Web Gateway Security Suite components, manages the operation of all components and organizes trusted data exchange among them.

Dr.Web ConfigD performs the following functions:

- starting and stopping the components of Dr.Web Gateway Security Suite depending on settings;
- restarting the components automatically in case of failures;
- starting the components upon the request of other components;
- informing the components of changed settings;
- enabling the centralized management of configuration parameters;
- passing the information from the key file in use to the components;
- receiving the license information from the components;
- receiving the new license information from the specialized components;
- informing the running components of license changes.

9.1.1. Operating Principles

The Dr.Web ConfigD configuration management daemon always runs with superuser privileges (*root*). It starts other Dr.Web Gateway Security Suite components and communicates with them via a preliminarily open socket. The configuration daemon accepts connections from other Dr.Web Gateway Security Suite components via an information socket (publicly accessible) and an administrative socket (accessible only to the components running with superuser privileges). The daemon loads configuration parameters and license information from files or receives this information from a centralized protection server via Dr.Web ES Agent and sets valid default values of the configuration parameters. By the time any component starts or receives the SIGHUP signal, the configuration management daemon has a comprehensive and consistent set of configuration parameters for all Dr.Web Gateway Security Suite components.

Upon receiving the SIGHUP signal, Dr.Web ConfigD reloads the configuration parameters and license information. If required, the daemon also instructs all components to reload their configuration parameters.

Upon receiving the SIGTERM signal, Dr.Web ConfigD shuts down all components and then finishes its own operation. Dr.Web ConfigD removes all temporary files of the components after they are shut down.

Principles of interaction with other components

1. All components receive the configuration parameters and license information from Dr.Web ConfigD at startup. Only these settings are used by the components in their further operation.
2. Dr.Web ConfigD forwards messages from all components started with it to a unified log. All messages output to *stderr* by the components are gathered by Dr.Web ConfigD and stored in the unified log of Dr.Web Gateway Security Suite with a mark indicating the component that reported an error and time of its occurrence.
3. Upon shutting down, all controlled components return an exit code. If the code differs from `101`, `102` and `103`, the component will be restarted and the corresponding message from *stderr* will be output to the Dr.Web Gateway Security Suite log.
   - Code 101 is returned when the component cannot operate with the current license. The component will be restarted only after the license parameters are modified.
   - Code 102 is returned when the component cannot operate with the current configuration parameters. If some configuration parameters were modified, Dr.Web ConfigD will try to restart the component.
   - Code `103` is returned in case the components started by Dr.Web ConfigD upon request (Dr.Web Scanning Engine and Dr.Web File Checker) have been idle for a long time. The timeout after which the component is shut down with error code `103` is specified in the settings of the corresponding component (the `IdleTimeLimit` parameter).
   - If new configuration parameters received from Dr.Web ConfigD by the component cannot be applied on the fly, the component exits with code `0` so that Dr.Web ConfigD can restart it.
   - If the component cannot connect to Dr.Web ConfigD or a communication protocol error occurs, the component outputs a corresponding message to *stderr* and exits with code `1`.
4. Signal exchange is maintained.
   - Dr.Web ConfigD sends the SIGHUP signal to the component instructing it to apply the modified configuration parameters.
   - Dr.Web ConfigD sends the SIGTERM signal to the component instructing it to shut down. After receiving the signal, the component should shut down within 30 seconds.
   - If the component does not shut down within 30 seconds, Dr.Web ConfigD sends the SIGKILL signal to forcibly shut down the component.

9.1.2. Command-Line Arguments

To start the Dr.Web ConfigD configuration management daemon from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-configd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-configd [<parameters>]
```

Dr.Web ConfigD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None
`--config`	Function: Use the specified configuration file for further operation. Short form: `-c` Arguments: <*path to file*>—path to the configuration file in use.
`--daemonize`	Function: Run the component as a daemon; that is, without access to the terminal. Short form: `-d` Arguments: None
`--pid-file`	Function: Use the specified PID file for further operation. Short form: `-p` Arguments: <*path to file*>—path to the file to store the process identifier (PID).

Example:

```
$ /opt/drweb.com/bin/drweb-configd -d -c /etc/opt/drweb.com/drweb.ini
```

This command runs Dr.Web ConfigD as a daemon, which forces the component to use the configuration file `/etc/opt/drweb.com/drweb.ini`.

Startup notes

To enable the operation of Dr.Web Gateway Security Suite, the component must run as a daemon. Under normal conditions, Dr.Web ConfigD starts automatically at the startup of the

operating system; for this purpose the component is supplied with the `drweb-configd` management script located in a system directory (`/etc/init.d` for GNU/Linux or `/usr/local/etc/rc.d` for FreeBSD). To manage the operation of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line (the tool is started by running the `drweb-ctl` command).



To get documentation on this component from the command line, run the `man 1 drweb-configd` command.

9.1.3. Configuration Parameters

The Dr.Web ConfigD configuration management daemon uses configuration parameters specified in the `[Root]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the Dr.Web ConfigD component. Default value: `Notice`
`Log` *{log type}*	Logging method of the configuration daemon and of those components for which another value of this parameter is not specified.  Upon its initial startup, before the configuration file is read, the configuration daemon uses the following values of the parameter: • as a daemon (if run with the `-d` option)—`SYSLOG:Daemon`; • `Stderr` otherwise. If a component is operating in the background (was started with the `-d` option from the command line), the `Stderr` value *cannot* be used for this parameter. Default value: `SYSLOG:Daemon`
`PublicSocketPath` *{path to file}*	Path to the socket used for interaction by all Dr.Web Gateway Security Suite components. Default value: `/var/run/.com.drweb.public`
`AdminSocketPath` *{path to file}*	Path to the socket used for interaction by Dr.Web Gateway Security Suite components with elevated (administrative) privileges. Default value: `/var/run/.com.drweb.admin`

Parameter	Description
`DebugIpc` *{boolean}*	Log or do not log detailed IPC messages at the debug level (with `LogLevel = DEBUG`). These messages reflect interaction of the configuration management daemon with other components. Default value: `No`
`UseCloud` *{boolean}*	Use or do not use the Dr.Web Cloud service to receive information about whether files and URLs are malicious or not. Default value: `No`
`UseMesh` *{boolean}*	Use or do not use the Dr.Web MeshD component for scanning. If this parameter is disabled, Dr.Web Scanning Engine is used for scanning instead of Dr.Web MeshD. Allowed values: • `On`, `Yes`, `True`—use Dr.Web MeshD; • `Off`, `No`, `False`—do not use Dr.Web MeshD. If the Dr.Web MeshD component is not installed, the Dr.Web Scanning Engine component is used irrespective of the set value. If none of the components is installed, the scan ends with error x119. Default value: `No`
`UseVxcube` *{boolean}*	Use or do not use Dr.Web vxCube to analyze email attachments as an external filter connected to the MTA. Default value: `No`
`KeyPath` *{path to file}*	Path to the key file (license or demo). Default value: • for GNU/Linux: `/etc/opt/drweb.com/drweb32.key` • for FreeBSD: `/usr/local/etc/drweb.com/drweb32.key`
`CoreEnginePath` *{path to file}*	Path to the dynamic library of Dr.Web Virus-Finding Engine. Default value: • for GNU/Linux: `/var/opt/drweb.com/lib/drweb32.dll` • for FreeBSD: `/var/drweb.com/lib/drweb32.dll`
`VirusBaseDir` *{path to directory}*	Path to the directory with virus database files. Default value: • for GNU/Linux: `/var/opt/drweb.com/bases` • for FreeBSD: `/var/drweb.com/bases`
`DwsDir` *{path to directory}*	Path to the directory that contains files of an automatically updated database of web resource categories. Default value: • for GNU/Linux: `/var/opt/drweb.com/dws`

Parameter	Description
	• for FreeBSD: `/var/drweb.com/dws`
`VersionDir` *{path to directory}*	Path to a directory, where the information about the current versions of Dr.Web Gateway Security Suite components is stored. Default value: • for GNU/Linux: `/var/opt/drweb.com/version` • for FreeBSD: `/var/drweb.com/version`
`CacheDir` *{path to directory}*	Path to the cache directory (being used to store cache for updates as well as cache for information about scanned files). Default value: • for GNU/Linux: `/var/opt/drweb.com/cache` • for FreeBSD: `/var/drweb.com/cache`
`TempDir` *{path to directory}*	Path to the directory with temporary files. Default value: *Path from the system environment variable* `TMPDIR`, `TMP`, `TEMP` or `TEMPDIR` (the environment variables are searched in this particular order). Otherwise `/tmp`, if none of them was found.
`RunDir` *{path to directory}*	Path to the directory with all PID files of running components and sockets used for interaction by Dr.Web Gateway Security Suite components. Default value: `/var/run`
`VarLibDir` *{path to directory}*	Path to the directory with libraries used by Dr.Web Gateway Security Suite components. Default value: • for GNU/Linux: `/var/opt/drweb.com/lib` • for FreeBSD: `/var/drweb.com/lib`
`AdminGroup` *{group name | GID}*	Group of users with administrative privileges for Dr.Web Gateway Security Suite management. These users, in addition to the superuser (the *root* user), are allowed to elevate privileges of Dr.Web Gateway Security Suite components to superuser privileges. Default value: *Defined automatically* during Dr.Web Gateway Security Suite installation
`TrustedGroup` *{group name | GID}*	Group of trusted users. The parameter is used by the SpIDer Gate network traffic monitoring component. Network traffic of such users is skipped by SpIDer Gate. Default value: `drweb`
`VersionNotification` *{boolean}*	Notify or do not notify the user of updates for the current version of Dr.Web Gateway Security Suite. Default value: `Yes`

Parameter	Description
`DefaultLogLevel` *{logging level}*	Default event logging level *for all* Dr.Web Gateway Security Suite components. The value of this parameter is used if a custom logging level is not specified for some component. Default value: `Notice`
`VxcubeApiAddress` *{string}*	Domain name (FQDN) or IP address of a host running the Dr.Web vxCube API server to be connected to. Default value: *(not specified)*
`VxcubeApiKey` *{string}*	Dr.Web vxCube API key. Default value: *(not specified)*
`VxcubeProxyUrl` *{connection address}*	Address of the proxy server used for connecting to Dr.Web vxCube. Only HTTP proxies without authorization are supported. Possible values: <*connection address*>—proxy server connection parameters in the following format: `http://`<*host*>`:`<*port*>, where: • <*host*> is the host address of the proxy server (an IP address or a domain name, i.e. FQDN); • <*port*> is the port to be used. Default value: *(not specified)*

9.2. Dr.Web Ctl

In this section

- General information
- Remote host scanning

General information

You can manage operation of Dr.Web Gateway Security Suite from the command line of the operating system. For that, you can use the special Dr.Web Ctl utility (`drweb-ctl`). You can use it to perform the following operations:

- start scanning files, disk boot records and executables of active processes;
- start scanning files on remote network hosts (see the note);
- start updating anti-virus components (virus databases, the scanning engine and so on depending on the distribution);
- view and change parameters of the Dr.Web Gateway Security Suite configuration;

- view the status of the Dr.Web Gateway Security Suite components and statistics on detected threats;
- view quarantine and manage quarantined objects (via the Dr.Web File Checker component);
- connect to a centralized protection server or disconnect from it.

User commands to control Dr.Web Gateway Security Suite will only take effect if the Dr.Web ConfigD configuration daemon is running (by default, it is automatically run on system startup).

Note that some control commands require superuser privileges.

To elevate privileges, use the `su` command (change the current user) or the `sudo` command (run the specified command as another user).

The `drweb-ctl` tool supports auto-completion of commands for managing Dr.Web Gateway Security Suite operation if this option is enabled your command shell. If the command shell does not allow auto-completion, you can configure this option. For that purpose, refer to the instruction manual for the used OS distribution.

When shutting down, the tool returns the exit code according to convention for the POSIX compliant systems: 0 (zero)—if an operation is successfully completed, non-zero—if otherwise.

The tool only returns a non-null exit code in the case of an internal error (for example, the tool could not connect to a component, the requested operation could not be executed, and so on). If the tool detects and possibly neutralizes a threat, it returns the null exit code, because the requested operation (such as `scan` and so on) is successfully completed. If you need to define the list of the detected threats and applied actions, analyze the messages displayed on the console.

Codes of all errors are listed in the Appendix F. Known Errors section.

Remote host scanning

Dr.Web Gateway Security Suite allows you to scan files located on remote network hosts for threats. Such hosts can be not only fully-featured computing machines, such as workstations and servers, but also routers, set-top boxes, and other smart devices of the Internet of Things. To perform the remote scanning, the remote host has to provide a remote terminal access via *SSH* (*Secure Shell*) or *Telnet*. To access the device, you need to know an IP address and a domain name of the remote host, as well as the credentials of the user that can remotely access the system via *SSH* or *Telnet*. This user must have access rights to the scanned files (at least the reading rights).

This function can be used only for detection of malicious and suspicious files on a remote host. Elimination of threats (i.e. isolation in the quarantine, removal, and cure of malicious objects) using remote scanning is impossible. To eliminate the detected threats on the remote host, use administration tools provided directly by this host. For example, for routers and other smart

devices, update the firmware; for computing machines, establish a connection (in a remote terminal mode, as one of the options) and perform the respective operations in the file system (remove or move files, etc.), or run the anti-virus software installed on them.

Remote scanning is performed only with the `drweb-ctl` command-line tool (using the `remotescan` command).

9.2.1. Command-Line Call Format

The call format for the command-line tool which manages Dr.Web Gateway Security Suite operation is as follows:

```
$ drweb-ctl [<general options> | <command> [<argument>] [<command options>]]
```

where:

- *<general options>*—options that can be applied on startup when the command is not specified or can be applied for any command. Not mandatory for startup.
- *<command>*—command to be run by Dr.Web Gateway Security Suite (for example, start scanning, output the list of quarantined objects and so on).
- *<argument>*—command argument. Depends on the specified command. Some commands do not accept arguments.
- *<command options>*—options for managing the operation of the specified command. Depend on the command. Some commands do not accept options.

9.2.2. General Options

The following general options are available:

Option	Description
`-h, --help`	Show general help information and exit. To display the help information on any command, use the call: `$ drweb-ctl <command> -h`
`-v, --version`	Show the module version and exit
`-d, --debug`	Show debug information when running the specified command. It cannot be run if a command is not specified. Use the call: `$ drweb-ctl <command> -d`

9.2.3. Commands

In this section

- Anti-Virus Scanning Commands

- Commands to Manage Detected Threats and Quarantine
- Configuration Management Commands
- Advanced Commands
  - Commands to Manage Updates and Operation in Centralized Protection Mode
  - Information Commands

9.2.3.1. Anti-Virus Scanning Commands

The following commands to manage anti-virus scanning are available:

Command	Description
`scan <path>`	**Purpose:** Initiate scanning the specified file or directory by the file scanning component Dr.Web File Checker. **Arguments** <*path*>—path (can be relative) to the file or directory to be scanned. This argument may be omitted if you use the `--stdin` or the `--stdin0` option. To specify several files that satisfy a certain criterion, use the `find` utility (see Usage Examples) and the `--stdin` or `--stdin0` option. **Options** `-a [--Autonomous]`—run standalone instances of Dr.Web Scanning Engine and Dr.Web File Checker to perform the specified scan, shutting them down after it is completed. ⚠ Threats detected during standalone scanning will not be added to the common list of threats detected displayed by the `threats` command, and a centralized protection server will not be notified of them, if Dr.Web Gateway Security Suite is controlled by it. `--Report <type>`—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout <time interval>`—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--FollowSymlinks`—resolve symlinks automatically.

Command	Description
	`--PackerMaxLevel` <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ArchiveMaxLevel` <*number*>—maximum nesting level while scanning archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MailMaxLevel` <*number*>—maximum nesting level while scanning files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ContainerMaxLevel` <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MaxCompressionRatio` <*ratio*>—maximum compression ratio of scanned objects. The value must be no less than `2`. Default value: `3000`. `--MaxSizeToExtract` <*number*>—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*. `--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Exclude` <*path*>—excluded path. The path can be relative and contain a file mask (with the following wildcards: `?` and `*`, as well as character classes `[ ]`, `[! ]` and `[^ ]`).

Command	Description
	Optional parameter; can be set more than once. `--OnKnownVirus` <*action*>—action to perform upon detection of a known threat by using the signature-based analysis. Allowed actions: `REPORT`, `CURE`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnIncurable` <*action*>—action to perform upon detection an incurable threat or when the curing action (`CURE`) has failed. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnSuspicious` <*action*>—action to perform upon detection of a suspicious object using the heuristic analysis. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnAdware` <*action*>—action to perform upon detection of adware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnDialers` <*action*>—action to perform upon detection of a dialer. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnJokes` <*action*>—action to perform upon detection of joke software. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnRiskware` <*action*>—action to perform upon detection of riskware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnHacktools` <*action*>—action to perform upon detection of a hacktool. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`.  If a threat is detected in a file inside a container (an archive, an email message and so on), the container is quarantined (`QUARANTINE`) and not deleted (`DELETE`). `--stdin`—get the list of paths to be scanned from the standard input stream (*stdin*). Paths in the list must be separated with the new line character (`\n`). `--stdin0`—get the list of paths to scan from the standard input string (*stdin*). Paths in the list must be separated by the zero character NUL (`\0`).

Command	Description
	When using `--stdin` and `--stdin0` options, the paths on the list should not contain patterns or regular expressions for a search. We recommend that you use the `--stdin` and `--stdin0` options to process a path list generated by an external utility, for example, `find` in the `scan` command (see Usage Examples).
`bootscan` *<device>* `|` `ALL`	**Purpose:** Start scanning boot records on specified disks using the file scan component Dr.Web File Checker. Both MBR and VBR records are scanned. **Arguments** *<disk drive>*—path to the block file of a disk device whose boot record you want to scan. You can specify several disk devices separated by spaces. The argument is mandatory. If `ALL` is specified instead of the device file, all boot records on all available disk devices will be checked. **Options** `-a [--Autonomous]`—run standalone instances of Dr.Web Scanning Engine and Dr.Web File Checker to perform the specified scan, shutting them down after it is completed. Threats detected during standalone scanning will not be added to the common list of threats detected displayed by the `threats` command, and a centralized protection server will not be notified of them, if Dr.Web Gateway Security Suite is controlled by it. `--Report` *<type>*—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` *<time interval>*—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--HeuristicAnalysis` *<On|Off>*—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Cure` *<Yes|No>*—attempt or do not attempt to cure detected threats. If the value is set to `No`, only a notification about a detected threat is displayed.

Command	Description
	Default value: `No`. `--ShellTrace`—display additional debug information when scanning a boot record
`procscan`	**Purpose:** Initiate scanning of executables containing the code of currently running system processes with the Dr.Web File Checker component. If a malicious executable file is detected, it is neutralized, and all processes run by this file are forced to terminate. **Arguments**: None. ⚠ This command is intended only for Dr.Web Gateway Security Suite distributions running on GNU/Linux OSes. **Options** `-a [--Autonomous]`—run standalone instances of Dr.Web Scanning Engine and Dr.Web File Checker to perform the specified scan, shutting them down after it is completed. ⚠ Threats detected during standalone scanning will not be added to the common list of threats detected displayed by the `threats` command, and a centralized protection server will not be notified of them, if Dr.Web Gateway Security Suite is controlled by it. `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--PackerMaxLevel` <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`.

Command	Description
	`--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--ContainerMaxLevel` <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--Exclude` <*path*>—path to be excluded from scanning. The path can contain a file mask with the following allowed symbols: `?` and `*`, as well as the symbol classes `[ ]`, `[! ]`, `[^ ]`. The path (including the path with the file mask) must be absolute. Optional parameter; can be set more than once. `--OnKnownVirus` <*action*>—action to perform upon detection of a known threat by using the signature-based analysis. Allowed actions: `REPORT`, `CURE`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnIncurable` <*action*>—action to perform upon detection an incurable threat or when the curing action (`CURE`) has failed. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnSuspicious` <*action*>—action to perform upon detection of a suspicious object using the heuristic analysis. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnAdware` <*action*>—action to perform upon detection of adware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnDialers` <*action*>—action to perform upon detection of a dialer. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnJokes` <*action*>—action to perform upon detection of joke software. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. `--OnRiskware` <*action*>—action to perform upon detection of riskware. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`.

Command	Description
	`--OnHacktools` <*action*>—action to perform upon detection of a hacktool. Allowed actions: `REPORT`, `QUARANTINE`, `DELETE`. Default value: `REPORT`. If a threat is detected in an executable file, Dr.Web Gateway Security Suite terminates all processes started by the file.
`memscan --Cure` <*boolean*>	**Purpose**: Start process scan. Unlike the `procscan` command, scanning occurs only in random-access memory. **Arguments**: None. **Options** `--Cure` <*boolean*>—cure a threat upon detection. Mandatory parameter. Allowed values: • `On`, `Yes`, `True`—cure a threat upon detection; • `Off`, `No`, `False`—do not cure a threat upon detection. `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one process in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `-d [--Debug]`—display debug information while running the specified command.
`netscan [`<*path*>`]`	**Purpose:** Start distributed scanning of the specified file or directory using the Dr.Web Network Checker agent for network data scanning. If there are no configured connections to other hosts that are running Dr.Web for Unix, then the scanning will be done only using the locally available scan engine (similar to the `scan` command). **Arguments** <*path*>—path to the file or directory to be scanned. If this argument is omitted, data from the stdin input stream will be scanned.

Command	Description
	Options `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--PackerMaxLevel` <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ArchiveMaxLevel` <*number*>—maximum nesting level while scanning archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MailMaxLevel` <*number*>—maximum nesting level while scanning files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ContainerMaxLevel` <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MaxCompressionRatio` <*ratio*>—maximum compression ratio of scanned objects.

Command	Description
	The value must be no less than `2`. Default value: `3000`. `--MaxSizeToExtract` <*number*>—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*. `--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Cure` <*Yes*|*No*>—attempt or do not attempt to cure detected threats. If the value is set to `No`, only a notification about a detected threat is displayed. Default value: `No`
`rawscan` <*path*>	**Purpose**: Start "raw" scanning of the specified file or directory with Dr.Web Scanning Engine directly, without the use of Dr.Web File Checker. ⚠ Threats detected during "raw" scanning are not included in the list of detected threats that can be displayed using the `threats` command. It is recommended that you use this command only to debug the functioning of Dr.Web Scanning Engine. Note that the command outputs the "cured" status, if at least *one* threat is neutralized of those threats that are detected in a file (not *all* threats might be neutralized). Thus, it is *not recommended* to use this command if you need thorough file scanning. In the latter case it is recommended to use the `scan` command. **Arguments** <*path*>—path to the file or directory to be scanned. **Options** `--ScanEngine` <*path*>—path to the Unix socket of Dr.Web Scanning Engine. If not specified, a standalone instance of the scan engine will be started (which will be shut down once the scanning is complete). `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format.

Command	Description
	Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--PackerMaxLevel` <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ArchiveMaxLevel` <*number*>—maximum nesting level while scanning archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MailMaxLevel` <*number*>—maximum nesting level while scanning files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ContainerMaxLevel` <*number*>—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MaxCompressionRatio` <*ratio*>—maximum compression ratio of scanned objects. Must be no less than `2`. Default value: `3000`. `--MaxSizeToExtract` <*number*>—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*.

Command	Description
	`--HeuristicAnalysis` <*On*|*Off*>—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Cure` <*Yes*|*No*>—attempt or do not attempt to cure detected threats. If the value is set to `No`, only a notification about a detected threat is displayed. Default value: `No`. `--ListCleanItem`—output the list of clean (non-infected) files found inside the container that was scanned. `--ShellTrace`—enable display of additional debug information when scanning a file. `--Output` <*path to file*>—duplicate the output of the command to the specified file
`remotescan` <*host*> <*path*>	**Purpose**: Start scanning the specified file or directory at the specified remote host having connected to it using *SSH* or *Telnet*. ⚠ Threats detected during remote scanning are not neutralized and are also not added to the list of detected threats displayed using the `threats` command. This function can be used only for detection of malicious and suspicious files on a remote host. To eliminate detected threats on the remote host, it is necessary to use administration tools provided directly by this host. For example, for routers, set-top boxes, and other "smart" devices, a mechanism for a firmware update can be used; for computing machines, it can be done by connecting to them (as an option, using a remote terminal mode) and by performing corresponding operations in their file system (file removal or moving and so on), or by running an anti-virus software installed on them. **Arguments** • <*host*>—IP address or a domain name of the remote host to be connected to for scanning. • <*path*>—path to the file or directory to be scanned (the path must be absolute). **Options** `-l [--Login]` <*name*>—login (user name) used for authorization on the remote host via the selected protocol. If a user name is not specified, an attempt is made to connect to a remote host as the user who started the command.

Command	Description
	`-i [--Identity]` <*path to file*>—private key file used for authentication of the specified user via the selected protocol. The `remotescan` command is designed to scan SOHO (Small Office / Home Office) routers. A key in the format used by default by the `ssh-keygen` utility is not supported. To scan a remote host via SSH, a key in the PEM format is required. To generate the key, use the command: `$ ssh-keygen -t rsa -m PEM -f rsa_pem` `-m [--Method]` <*SSH*|*Telnet*>—remote host connection method (protocol). If the method is not specified, SSH is used. `-p [--Port]` <*number*>—number of the port on the remote host for connecting via the selected protocol. Default value: Default port for the selected protocol (`22` for SSH, `23` for Telnet). `--UseChannels` <*number*>—number of data transfer channels. Default value: `5`. `--Password` <*password*>—password used for authentication of a user via the selected protocol. Password is passed as plain text. `--Report` <*type*>—type of the scan report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. Default value: `BRIEF`. `--ScanTimeout` <*time interval*>—timeout for scanning one file in milliseconds. If the value is set to `0`, scanning time is not limited. Default value: `0`. `--PackerMaxLevel` <*number*>—maximum nesting level while scanning packed objects. A packed object is executable code compressed with specialized software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned.

Command	Description
	If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ArchiveMaxLevel <number>`—maximum nesting level while scanning archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MailMaxLevel <number>`—maximum nesting level while scanning files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--ContainerMaxLevel <number>`—maximum nesting level while scanning other types of objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. If the value is set to `0`, nested objects are skipped. Default value: `8`. `--MaxCompressionRatio <ratio>`—maximum compression ratio of scanned objects. Must be no less than `2`. Default value: `3000`. `--MaxSizeToExtract <number>`—maximum size for files enclosed in archives. Files which size is greater than the value of this parameter will be skipped when scanning. The size is specified as a number with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Default value: *(not specified)*. `--HeuristicAnalysis <On|Off>`—enable or disable the heuristic analysis during a scan. Default value: `On`. `--Exclude <path>`—path to be excluded from scanning. The path can contain a file mask with the following allowed symbols: `?` and `*`, as well as the symbol classes `[ ]`, `[! ]`, `[^ ]`. The path (including the path with the file mask) must be absolute. Optional parameter; can be set more than once. `--TransferListenAddress <address>`—address for receiving files transferred from the remote device for scanning. Optional parameter. If not indicated, an arbitrary address is used.

Command	Description
	`--TransferListenPort` <*port*>—port for receiving files transferred from the remote device for scanning. Optional parameter. If not indicated, an arbitrary port is used. `--TransferExternalAddress` <*address*>—address for the remote device to send files for scanning. Optional parameter. If not indicated, the `--TransferListenAddress` option value or the outgoing address of the already established session is used. `--TransferExternalPort` <*port*>—port to transfer files for scanning, specified for the remote device. Optional parameter. If not indicated, an automatically determined port is used. `--ForceInteractive`—use the SSH interactive session (only for SSH connections). *Optional parameter.*

9.2.3.2. Commands to Manage Detected Threats and Quarantine

The following commands for managing threats and quarantine are available:

Command	Description
`threats [`<*action*> <*object*>`]`	**Purpose**: Apply the specified action to earlier detected threats according to their identifiers. A type of the action is specified by the command option. If the action is not specified, displays information about detected but not neutralized threats. The information about threats is displayed according the format, specified using the non-mandatory `--Format` option. If the `--Format` option is not specified, the following information is displayed for each threat: • an identifier assigned to the threat (its ordinal number); • the full path to the infected file; • information about the threat (its name and type according to the classification of the Doctor Web company); • information about the file: its size, owner, time of last modification; • history of operations applied to an infected file: detection, applied actions and so on. **Arguments**: None. **Options** `--Format "`<*format string*>`"`—output information about threats in the specified format. The description of the format string is below.

Command	Description
	If this option is specified together with any action option, it is ignored. `-f [--Follow]`—wait for new messages about new threats and display them once they are received (CTRL+C interrupts the waiting). If this option is specified together with any action option, it is ignored. `--Cure` <*threat list*>—attempt to cure the listed threats (threat identifiers are comma-separated); `--Quarantine` <*threat list*>—quarantine the listed threats (threat identifiers are comma-separated); `--Delete` <*threat list*>—delete the listed threats (threat identifiers are comma-separated); `--Ignore` <*threat list*>—ignore the listed threats (threat identifiers are comma-separated). If you need to apply the action to all detected threats, specify `All` instead of <*threat list*>. For example, the command: `$ drweb-ctl threats --Quarantine All` quarantines all detected malicious objects. `--Directory` <*list of directories*>—output only threats detected in files in directories from <*list of directories*>. If this option is specified together with any action option, it is ignored.
`quarantine [`<*action*> <*object*>`]`	**Purpose:** Apply an action to the specified object in quarantine. If the action is not specified, information about quarantined objects and their identifiers together with brief information about original files put in quarantine is displayed. Information about isolated objects is output according to a format specified with the optional `--Format` parameter. If the `--Format` parameter is not specified, the following information is output for every isolated (quarantined) object: • an identifier assigned to a quarantined object; • the original path to the file that was moved to quarantine; • the date of putting the file in quarantine; • information about the file: its size, owner, time of last modification; • information about the threat (name of the threat, threat type according to the classification used by the Doctor Web company). **Arguments**: None.

Command	Description
	Options `--Format "<format string>"`—display information about quarantined objects in the specified format. The description of format string is below.  If this option is specified together with any action option, it is ignored. `-f [--Follow]`—wait for new messages about new threats and display them once they are received (CTRL+C interrupts the waiting).  If this option is specified together with any action option, it is ignored. `-a [--Autonomous]`—run a standalone instance of the Dr.Web File Checker file scanning component to perform the specified quarantine action and shut down the component upon completion. This option can be used together with any options mentioned below. `--Discovery [<list of directories>,]` searches for quarantine directories in the specified list of directories and add them to the consolidated quarantine upon detecting a threat. If the *<list of directories>* is not specified, search for quarantine directories in the common locations of the file system (volume mounting points and user home directories). This option can be specified not only with the `-a` (`--Autonomous`) option (see above), but also with any options/actions listed below. Moreover, if the `quarantine` command is run in standalone instance mode, that is, with the `-a` (`--Autonomous`) option but without the `--Discovery` option, then it has the same effect as calling: `quarantine --Autonomous --Discovery` `--Delete <object>`—delete the specified quarantined object.  Quarantined objects are deleted permanently—this action is irreversible. `--Restore <object>`—restore the specified object from the quarantine to its original location.  This command may require to run `drweb-ctl` with superuser (the *root* user) privileges. You can restore the file from quarantine even if it is infected. `--Cure <object>`—attempt to cure the specified object in the quarantine.

Command	Description
	Even if the object was successfully cured, it will remain in quarantine. To restore the cured object from quarantine, use the `--Restore` option. `--TargetPath` <*path*>—restore an object from quarantine to the specified location: either as a file with the specified name (if <*path*> is a path to a file), or to the specified directory (if <*path*> is a path to a directory). A path can be absolute or relative (with regard to the current directory). This option can only be used in combination with the `--Restore` option. As an <*object*>, specify the object identifier in quarantine. To apply the action to all quarantined objects, specify `All` instead of <*object*>. For example, the command `$ drweb-ctl quarantine --Restore All --TargetPath test` restores all quarantined objects and puts them in the `test` subdirectory located in the current directory from which the `drweb-ctl` command was run. If the `--Restore All` variant is indicated together with the additional option `--TargetPath`, this option must set a path to a directory, not to a file.

Formatted output for threats and quarantine commands

The output format is defined using the format string specified as the optional argument `--Format`. The format string must be put in quotes. The format string can include common symbols (displayed "as is"), as well as special markers which will be replaced with corresponding information at the output. The following markers are available:

1. Common for `threats` and `quarantine` commands:

Marker	Description
`%{n}`	New line
`%{t}`	Tabulation
`%{threat_name}`	The name of the detected threat according to the classification of the Doctor Web company
`%{threat_type}`	Threat type ("known virus" and so on) according to Doctor Web classification

Marker	Description
`%{size}`	Original file size
`%{origin}`	The full name of the original file with path
`%{path}`	Synonym of `%{origin}`
`%{ctime}`	Modification date/time of the original file in "*%Y-%b-%d %H:%M:%S*" format (for example, "`2018-Jul-20 15:58:01`")
`%{timestamp}`	Similar to `%{ctime}`, but in the *Unix timestamp* format
`%{owner}`	The original file owner
`%{rowner}`	The remote owner of the original file (if not applicable or value is unknown it is replaced with `?`)

2. Specific for `threats` command:

Marker	Description
`%{hid}`	The identifier of the threat record in the history of events associated with the threat
`%{tid}`	Threat identifier
`%{htime}`	Date/time of the event related to the threat
`%{app}`	The identifier of the Dr.Web Gateway Security Suite component which processed a threat
`%{event}`	The latest event related to a threat: • `FOUND`—threat was detected; • `CURE`—threat was cured; • `QUARANTINE`—file with a threat was quarantined; • `DELETE`—file with a threat was deleted; • `IGNORE`—threat was ignored; • `RECAPTURED`—threat was detected by another component
`%{err}`	Error message text (if no error has occurred, the text is replaced with an empty string)

3. Specific for `quarantine` command:

Marker	Description
`%{qid}`	The identifier of the quarantined object
`%{qtime}`	Date/time of moving the object to quarantine

Marker	Description
`%{curetime}`	Date/time of curing attempt of the quarantined object (if not applicable or the value is unknown, it is replaced with `?`)
`%{cureres}`	The result of the quarantined object curing attempt: • `cured`—threat was cured; • `not cured`—threat was not cured or no curing attempts were made

Example

```
$ drweb-ctl quarantine --Format "{%{n} %{origin}: %{threat_name} - %{qtime}%{n}}"
```

This command displays quarantine contents as records of the following type:

```
{
 <path to file>: <threat name> - <date of putting in quarantine>
}
...
```

9.2.3.3. Configuration Management Commands

The following commands to manage configuration are available:

Command	Description
`cfset` *<section>*`.`*<parameter>* *<value>*	**Purpose**: Change the active value of the specified parameter in the current configuration of Dr.Web Gateway Security Suite. **Arguments** • *<section>*—name of the configuration file section which provides the parameter. This argument is mandatory. • *<parameter>*—name of the parameter to be changed. This argument is mandatory. • *<value>*—new parameter value. This argument is mandatory.

Command	Description
	 To specify a parameter value, the format <*section*>.<*parameter*> <*value*> is always used, the assignment character = is not used. If you want to indicate several parameter values, you need to repeatedly call the `cfset` command, as many times as the number of parameter values you want to add. To add a new value to the list of parameter values, you need to use the `-a` option (see below). You cannot specify the string <*parameter*> <*value 1*>, <*value 2*> as an argument, because the string "<*value 1*>, <*value 2*>" will be considered one value of <*parameter*>. For description of the configuration file, refer to the section Appendix D. Dr.Web Gateway Security Suite Configuration File, as well as the documentation displayed upon running `man 5 drweb.ini`. **Options** `-a [--Add]`—do not substitute the current parameter value but add the specified value to the list (allowed only for parameters that can accept a list of values). This option should also be used for adding new parameter groups with a tag. `-e [--Erase]`—do not substitute the current parameter value but remove the specified value from the list (allowed only for parameters that can have several values, specified as a list). `-r [--Reset]`—reset the parameter value to the default. At that, <*value*> is not required in the command and is ignored if specified. Options are not mandatory. If they are not specified, then the current parameter value (including a list of values) are substituted with the specified value. If you use the `-r` option for sections that contain individualized parameter settings for the Dr.Web ClamD component connection points, the parameter value in the individualized settings section will be changed to the value of its corresponding “parent” parameter in the component settings section. If you need to add a new connection point <*point*> for Dr.Web ClamD, run the command `cfset ClamD.Endpoint.<point> -a` for example:

Command	Description
	`cfset ClamD.Endpoint.point1 -a` This command requires `drweb-ctl` to be started with superuser (the *root* user) privileges. If necessary, use the `su` or `sudo` commands.
`cfshow [<section>[.<parameter>]]`	**Purpose**: Display parameters of the current configuration of Dr.Web Gateway Security Suite. The command to display parameters is specified as follows: <*section*>.<*parameter*> = <*value*>. Sections and parameters of non-installed components are not displayed by default. **Arguments** • <*section*>—name of the configuration file section parameters of which are to be displayed. The argument is optional. If not specified, parameters of all configuration file sections are displayed. • <*parameter*>—name of the displayed parameter. Optional argument. If not specified, all parameters of the section are displayed. Otherwise, only this parameter is displayed. If a parameter is specified without the section name, all parameters with this name from all of the configuration file sections are displayed. **Options** `--Uncut`—display all configuration parameters, and not only those used with the currently installed set of components. If the option is not specified, only parameters used by the installed components are displayed. `--Changed`—display only those parameters whose values differ from the default ones. `--Ini`—display parameter values in the `.ini` file format: at first, the section name is specified in square brackets, then the section parameters listed as <*parameter*> = <*value*> pairs (one pair per line). `--Value`—output only the value of the specified parameter. The <*parameter*> argument is mandatory in this case.
`reload`	**Purpose**: Reload the configuration of Dr.Web Gateway Security Suite. For that purpose, the Dr.Web ConfigD configuration management daemon performs the following actions: • rereads the configuration and notifies all Dr.Web Gateway Security Suite components about its changes;

Command	Description
	• reopens the Dr.Web Gateway Security Suite log; • starts the components that use virus databases (including the scanning engine); • attempts to start those components that were shut down abnormally. **Arguments**: None. **Options**: None

9.2.3.4. Advanced Commands

In this section

- Commands to Manage Updates and Operation in Centralized Protection Mode
- Information Commands

9.2.3.4.1. Commands to Manage Updates and Operation in Centralized Protection Mode

The following commands for managing updates are available, as well as commands for operation in the centralized protection mode:

Command	Description
`update`	**Purpose**: Start the updating process of anti-virus components (virus databases, the scan engine and so on, depending on the distribution) from Doctor Web update servers, terminate the updating process if it is already running, or perform rollback of the latest update to the previous versions of updated files. The command has no effect if Dr.Web Gateway Security Suite is connected to the centralized protection server. **Arguments**: None. **Options** `--Stop`—terminate the running update process. `--Rollback`—roll back the last update and restore the previous version of the files that have been updated during the last update. `--From` <*path*>—apply updates offline from a specified directory. `--Path` <*path*>—store files for updating offline in a specified directory. If this directory already has files, then they will be updated.

Command	Description
`esconnect` *<server>*`[:`*<port>*`]`	**Purpose**: Connect Dr.Web Gateway Security Suite to the specified centralized protection server. For details on the operation modes, refer to the Operation Modes section. **Arguments** • *<server>*—IP address or network name of the host on which the centralized protection server is operating. This argument is mandatory. • *<port>*—port number used by the centralized protection server. The argument is optional and should be specified only if the centralized protection server uses a non-standard port. **Options** `--Certificate` *<path>*—file path to a certificate of the centralized protection server, the connection to which will be established. `--Login` *<ID>*—login (workstation identifier) used for connection to the centralized protection server. `--Password` *<password>*—password for connection to the centralized protection server. `--Compress` *<On|Off>*—enable (`On`) or disable (`Off`) forced compression of transmitted data. If not specified, usage of compression is determined by the server. `--Encrypt` *<On|Off>*—enable (`On`) or disable (`Off`) forced encryption of transmitted data. If not specified, usage of encryption is determined by the server. `--Newbie`—connect as a "newbie" (get a new account on the server). `--Group` *<ID>*—identifier of the group to which the workstation is added on connection. Must be specified together with the `--Password` option. `--Rate` *<ID>*—identifier of the tariff group applied to your workstation when it is included in one of the centralized protection server groups. Must be specified together with the `--Group` option. `--CfgFile` *<path>*—connect to the centralized protection server using the configuration file with connection settings.  This command requires `drweb-ctl` to be started with superuser (the *root* user) privileges. If necessary, use the `su` or `sudo` commands.
`esdisconnect`	**Purpose**: Disconnect Dr.Web Gateway Security Suite from the centralized protection server and switch it to a standalone mode.  The command has no effect if Dr.Web Gateway Security Suite already operates in standalone mode. **Arguments**: None.

Command	Description
	Options: None.  This command requires `drweb-ctl` to be started with superuser (the *root* user) privileges. If necessary, use the `su` or `sudo` commands.

9.2.3.4.2. Information Commands

The following information commands are available:

Command	Description
`appinfo`	**Purpose**: Output information about active Dr.Web Gateway Security Suite components. The following information is output for each running component: • internally used name; • GNU/Linux process identifier (PID); • state (running, stopped and so on); • error code, if the component has been terminated owing to an error; • additional information (optional); For the configuration daemon (`drweb-configd`), the following is output as additional information: • the list of installed components—*Installed*; • the list of components which must be run by the configuration daemon—*Should run*. **Arguments**: None. **Options** `-f [--Follow]`—wait for new messages on module status change and display them once such a message is received (CTRL+C interrupts waiting)
`baseinfo`	**Purpose**: Display the information on the current version of the scan engine and status of virus databases. The following information is displayed: • version of the scan engine; • release date and time of the virus databases being used; • the number of available threat records; • the time of the last successful update of the virus databases and of the scan engine;

Command	Description
	• the time of the next scheduled automatic update. **Arguments**: None. **Options** `-l [--List]`—display the full list of loaded files of virus databases and a number of threat records in each file
`certificate`	**Purpose**: Display contents of the trusted Dr.Web certificate used by Dr.Web Gateway Security Suite to scan protected connections if this option is enabled in settings. To save the certificate in the <*cert_name*>`.pem` file, run the command: `$ drweb-ctl certificate > <cert_name>.pem` **Arguments**: None. **Options**: None
`events`	**Purpose**: Display Dr.Web Gateway Security Suite events. In addition, this command allows you to manage the events (mark them as read or delete them). **Arguments**: None. **Options** `--Report` <*type*>—type of the event report. Allowed values: • `BRIEF`—brief report; • `DEBUG`—detailed report; • `JSON`—serialized report in the JSON format. `-f [--Follow]`—wait for new events and display them upon their occurrence (CTRL+C interrupts waiting). `--ShowSeen`—display already read events as well; `-s [--Since]` <*date, time*>—show the events that occurred before the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`). `-u [--Until]` <*date, time*>—show the events that occurred no later than the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`). `-t [--Types]` <*type list*>—show the events of the specified types only (types are comma-separated). The following event type is available: `UnexpectedAppTermination`—unexpected component shutdown. To view all types of events, use `All`.

Command	Description
	`--Show` <*list of events*>—display the listed events (event identifiers are comma-separated); `--Delete` <*list of events*>—remove the listed events (event identifiers are comma-separated); `--MarkAsSeen` <*list of events*>—mark the listed events as read (event identifiers are comma-separated). If you want to mark as "read" or delete all events, specify `All` instead of <*events list*>. For example, the command `$ drweb-ctl events --MarkAsSeen All` will mark all existing events as "read"
`report` <*type*>	**Purpose**: Create a report on Dr.Web Gateway Security Suite events in the HTML format (the page body is output to the specified file). **Arguments** <*type*>—event type that required reporting (indicate one type). See allowed values in the `--Types` option description of the `events` command. A mandatory argument. **Options** `-o [--Output]` <*path to file*>—save the report to the specified file. The option is mandatory. `-s [--Since]` <*date, time*>—report events that occurred no earlier than the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`). `-u [--Until]` <*date, time*>—report events that occurred no later than the specified timestamp (<*date, time*> is specified as `"YYYY-MM-DD hh:mm:ss"`). `--TemplateDir` <*path to directory*>—path to the directory that contains HTML report templates. Options `-s`, `-u`, and `--TemplateDir` are not mandatory.
`license`	**Purpose**: Display the information about the currently active license, get a demo-version license, or get the key file for a license that has already been registered (for example, that has been registered on the company website). If no options are specified, then the following information is output (if you are using a license for the standalone mode): • a license number, • date and time when the license expires. If you are using a license provided to you by a centralized protection server (for the use of the product in the centralized protection mode or mobile mode), the corresponding message is output.

Command	Description
	Arguments: None. **Options** `--GetDemo`—request a demo key that is valid for one month and receive this key, if the conditions for the provision of a demo period have not been violated. `--GetRegistered` <*serial number*>—get a license key file for the specified serial number, if the conditions for the provision of a new key file have not been breached (for example, breached by using the product not in centralized protection mode, when the license is managed by a centralized protection server). `--NetworkTimeout` <*time interval*>—timeout in milliseconds for network operations during the use of the `license` command. This parameter is used to continue activation when the connection is temporarily lost. If the connection is re-established before the timeout expires, the activation will be resumed. If `0` is specified, then there is no timeout. Default value: `0`. `--Proxy http://`<*username*>`:`<*password*>`@`<`server address`>`:<*port*>—get a license key via the proxy server (used only with one of the previously mentioned options—`--GetDemo` or `--GetRegistered`). *If the serial number is not the one provided for a demo period, you must first register this number at the company website.* For further information about licensing Dr.Web products, refer to the Licensing section. To register a serial number or to get a demo period, an internet connection is required.
`log`	**Purpose**: Display the latest log records of Dr.Web Gateway Security Suite in the console (the *stdout* stream, similar to the `tail` command). **Arguments**: None. **Options** `-s [--Size]` <*number*>—number of the latest log records to be displayed on the screen. `-c [--Components]` <*components list*>—list of component identifiers whose records are displayed. Identifiers are space-separated. If no argument is defined, all available records logged by all components are displayed.

Command	Description
	Actual identifiers of the installed components (e.g. internal component names displayed in the log) can be displayed using the `appinfo` command. `-f [--Follow]`—wait for new log records and display them once they are received (CTRL+C interrupts waiting).  This command requires `drweb-ctl` to be started with superuser (usually the *root* user) privileges. If necessary, use the `su` or `sudo` commands.
`stat`	**Purpose:** Display statistics about the operation of components that process files or about the operation of the network data scanning agent Dr.Web Network Checker (press CTRL+C or Q to interrupt displaying the statistics). The statistics output includes: • a name of the component that initiated file scanning; • component PID; • an average number of files processed per second during the last minute, 5 minutes, 15 minutes; • a percentage of using the cache of the scanned files; • an average number of scan errors per second. For the distributed scanning agent, the following information is displayed: • a list of local clients that initiated scanning; • a list of remote hosts that received files for scanning; • a list of remote hosts that sent files for scanning. For local clients of the distributed scanning agent, their PID and name are specified; for remote clients—an address and port of the host. For both clients—local and remote—the following information is displayed: • an average number of files scanned per second; • an average number of sent and received bytes per second; • an average number of errors per second. **Arguments**: None. **Options** `-n [--Netcheck]`—display statistics on operation of the network data scanning agent
`lookup` *<type>@<tag>[@<template>]*	**Purpose**: Verify data source connection settings. **Arguments**

Command	Description
	<*type*>—data source type. The following values are allowed: `ldap`, `ad`, `pq`, `sqlite`, `mysql`, `redis`, `allmatch`, `regex`, `cidr`, `mask`. These types are characterized in the description of Dr.Web LookupD settings. <*tag*>—data source identifier. For example, if the `[LookupD.LDAP.auth1]` section is created in the unified configuration file, `auth1` will be an identifier of a data source of the LDAP type. <*template*>—template (string) to search in a data source. Optional argument. If the template is set, a search for it is performed and the *true* value (found) or the *false* value (not found) is returned. **Options** `--User` <*user name*>—user name to connect to the data source; `--Domain` <*domain name*>—domain name to connect to the data source. ⚠ This option can only be used in combination with the `--User` option. `-d [--Debug]`—display debug information. **Examples** `$ drweb-ctl lookup --user user --domain d1 "mysql@test1"`

9.2.4. Usage Examples

In this section

- Scanning objects
  - Simple scanning commands
  - Scanning of files selected by criteria
  - Scanning of additional objects
- Configuration management
- Threat management
- Example of operation in standalone instance mode

Scanning objects

Simple scanning commands

1. Perform scanning of the `/home` directory with default parameters:

```
$ drweb-ctl scan /home
```

2. Scan file paths listed in the `daily_scan` file (one path per line):

```
$ drweb-ctl scan --stdin < daily_scan
```

3. Perform scanning of the boot record on the *sda* drive:

```
$ drweb-ctl bootscan /dev/sda
```

4. Perform scanning of the running processes:

```
$ drweb-ctl procscan
```

Scanning of files selected by criteria

Examples for file selection for scanning are listed below and use the result of the `find` utility operation. The obtained list of files is sent to the `drweb-ctl scan` command with the `--stdin` or `--stdin0` parameter.

1. Scan listed files returned by the `find` utility and separated with the NUL (`\0`) character:

```
$ find -print0 | drweb-ctl scan --stdin0
```

2. Scan all files in all directories, starting from the root directory, on one partition of the file system:

```
$ find / -xdev -type f | drweb-ctl scan --stdin
```

3. Scan all files in all directories, starting from the root directory, with the exception of the `/var/log/messages` and `/var/log/syslog` files:

```
$ find / -type f ! -path /var/log/messages ! -path /var/log/syslog | drweb-ctl scan --stdin
```

4. Scan all files of the *root* user in all directories, starting from the root directory:

```
$ find / -type f -user root | drweb-ctl scan --stdin
```

5. Scan all files of the *root* and *admin* users in all directories, starting from the root directory:

```
$ find / -type f \( -user root -o -user admin \) | drweb-ctl scan --stdin
```

6. Scan all files of the users with UID within the range of 1000–1005 in all directories, starting from the root directory:

```
$ find / -type f -uid +999 -uid -1006 | drweb-ctl scan --stdin
```

7. Scan files in all directories, starting from the root directory, with a nesting level of no more than five:

```
$ find / -maxdepth 5 -type f | drweb-ctl scan --stdin
```

8. Scan files in a root directory while ignoring files in subdirectories:

```
$ find / -maxdepth 1 -type f | drweb-ctl scan --stdin
```

9. Scan files in all directories, starting from the root directory, while following all symbolic links:

```
$ find -L / -type f | drweb-ctl scan --stdin
```

10. Scan files in all directories, starting from the root directory, without following symbolic links:

```
$ find -P / -type f | drweb-ctl scan --stdin
```

11. Scan files created no later than May 1, 2017 in all directories, starting from the root directory:

```
$ find / -type f -newermt 2017-05-01 | drweb-ctl scan --stdin
```

Scanning of additional objects

Scan objects located in the `/tmp` directory on the remote host *192.168.0.1* by connecting to it via SSH as the user *user* with the password *passw*:

```
$ drweb-ctl remotescan 192.168.0.1 /tmp --Login user --Password passw
```

Configuration management

1. Display information about the running components of Dr.Web Gateway Security Suite:

```
$ drweb-ctl appinfo
```

2. Display all parameters of the `[Root]` section:

```
$ drweb-ctl cfshow Root
```

3. Set `No` as the value of the `Start` parameter in the `[LinuxSpider]` section of the active configuration (this will disable SpIDer Guard):

```
# drweb-ctl cfset LinuxSpider.Start No
```

Superuser privileges are required to perform this action. To elevate the privileges, you can use the `sudo` command:

```
$ sudo drweb-ctl cfset LinuxSpider.Start No
```

4. Force update of anti-virus components of Dr.Web Gateway Security Suite:

```
$ drweb-ctl update
```

5. Restart the component configuration of Dr.Web Gateway Security Suite:

```
# drweb-ctl reload
```

Superuser privileges are required to perform this action. To elevate the privileges, you can use the `sudo` command:

```
$ sudo drweb-ctl reload
```

6. Connect Dr.Web Gateway Security Suite to a centralized protection server operating on host *192.168.0.1* if a server certificate is stored in the `/home/user/cscert.pem` file:

```
$ drweb-ctl esconnect 192.168.0.1 --Certificate /home/user/cscert.pem
```

7. Connect Dr.Web Gateway Security Suite to the centralized protection server using the `install.cfg` configuration file:

```
$ drweb-ctl esconnect --CfgFile <path to install.cfg>
```

8. Disconnect Dr.Web Gateway Security Suite from the centralized protection server:

```
# drweb-ctl esdisconnect
```

Superuser privileges are required to perform this action. To elevate the privileges, you can use the `sudo` command:

```
$ sudo drweb-ctl esdisconnect
```

9. View the last log records made by the `drweb-update` and `drweb-configd` components in the Dr.Web Gateway Security Suite log:

```
# drweb-ctl log -c Update ConfigD
```

Threat management

1. Display information on detected threats:

```
$ drweb-ctl threats
```

2. Quarantine all files containing non-neutralized threats:

```
$ drweb-ctl threats --Quarantine All
```

3. Display the list of quarantined files:

```
$ drweb-ctl quarantine
```

4. Restore all quarantined files:

```
$ drweb-ctl quarantine --Restore All
```

Example of operation in standalone instance mode

Scan files and process quarantine in standalone instance mode:

```
$ drweb-ctl scan /home/user -a --OnKnownVirus=QUARANTINE
$ drweb-ctl quarantine -a --Delete All
```

The first command will scan files in the `/home/user` directory in standalone instance mode. Files containing known threats will be quarantined. The second command will process quarantine content (in standalone instance mode as well) and remove all quarantined objects.

9.2.5. Configuration Parameters

The Dr.Web Ctl command-line management tool does not have its own parameter section in the unified configuration file of Dr.Web Gateway Security Suite. The tool uses the parameters specified in the `[Root]` section.

9.3. Dr.Web Management Web Interface

In this section

- Function
- System requirements of the web interface
- Accessing the web interface
- Main menu

Function

The Dr.Web Gateway Security Suite management web interface allows you to:

- view the current state of Dr.Web Gateway Security Suite components, start or stop some of them;
- view the status of updates and start an updating process manually, if necessary;
- view the status of the product license and load a license key, if necessary;
- view the list of detected threats and manage quarantined objects (only threats detected in the local file system with the Dr.Web File Checker component are displayed);
- edit the settings of Dr.Web Gateway Security Suite components;
- connect Dr.Web Gateway Security Suite to a centralized protection server or switch to the standalone mode;
- start an on-demand scanning of local files (including by dragging and dropping them to the page opened in your browser).

System requirements of the web interface

Correct operation of the web interface is guaranteed for the following web browsers:

- Microsoft Internet Explorer 11 and later;
- Mozilla Firefox 25 and later;
- Google Chrome 30 and later.

Accessing the web interface

To access the web interface, type the following address in the address bar of your browser:

```
https://<drweb_host>:<port>/
```

where <*drweb_host*> is the IP address or the name of the host on which Dr.Web Gateway Security Suite including the Dr.Web HTTPD web interface server operates, and <*port*> is the port (on this host) listened by Dr.Web HTTPD. To access components of Dr.Web Gateway

Security Suite running on the local host, use the IP address `127.0.0.1` or the name `localhost`. By default, the port is `4443`.

Thus, to access the web interface on the local host by default, use the following URL:

```
https://127.0.0.1:4443/
```

After the connection to the management server is established, the startup page opens and displays the authentication form. To access management functions, fill in the authentication form by specifying the login and password of a user who has administrative privileges on the host on which Dr.Web Gateway Security Suite operates.

If needed, you can provide authorization via the web interface using a personal user certificate. To do so:

1. Create a personal certificate signed by a certificate authority certificate.
2. Import the signed certificate as a user authorization certificate in the browser that is used to connect to the management web interface.
3. In the Dr.Web HTTPD settings (the `AdminSslCA` parameter), specify a path to the certificate authority certificate used to sign your personal certificate.

If you use a personal user certificate to authorize on the web interface, the authorization form does not appear, and the user is authorized as *root*.

If necessary, refer to the Appendix E. Generating SSL Certificates section.

Main menu

In the left pane of the web interface, which appears once you have successfully passed authentication, there is a main menu with the following items:

- **Main**—open the main page that displays the list of installed components of Dr.Web Gateway Security Suite and their status.
- **Threats**—open a page that displays all threats detected on the server. In this section, you can manage detected threats, for example, quarantine infected objects, rescan, cure or delete malicious objects.
- **Settings**—open a page with settings of Dr.Web Gateway Security Suite components installed on the server.
- **Information**—open a page that displays brief information about the version of this web interface and about the state of the virus databases.
- **Help**—open a new browser tab with the help information on Dr.Web for Unix products.
- **Scan File**—display a panel for quick file scanning, which will stay available on top of any opened page of the web interface until you close this panel.
- **Log Out**—end the current management web interface session (unavailable if the authentication with a user personal certificate is used).

9.3.1. Managing the Components

You can view the list of components included in Dr.Web Gateway Security Suite and manage their operation on the **Main** page.

The listed Dr.Web Gateway Security Suite components are separated into two groups: main components that monitor threats and service components that are responsible for the overall correct operation of Dr.Web Gateway Security Suite. The list of main components (depends on the distribution) is displayed as a table in the upper part of the page. The following information is provided for each component:

1. **Name**. Click the name to open the settings page for a component.
2. **State**. A state of the component is indicated by a switch and a text label reflecting the current state of the component. To start the component or to suspend its operation, click the switch. The possible states of the switch are:

	Component is disabled and not used.
	Component is enabled and operates correctly.
	Component is enabled but does not operate because of an error.

   If an error occurred in the operation of the component, an error message is displayed instead of a note about a component state. Click  to open a window with detailed information about the occurred error and with recommendations for resolving it.
3. **Load**. An average number of files processed by the component per second within the last minute, 5 minutes, 15 minutes respectively are specified (three numbers separated with a slash).
4. **Errors**. An average number of errors encountered in the operation of the component per second within the last minute, 5 minutes, 15 minutes respectively are specified (three numbers separated with a slash).

To display a tooltip, place the cursor over .

Below the table that provides the information about the main components, you will find Dr.Web Gateway Security Suite service components (such as Dr.Web Scanning Engine, Dr.Web File Checker and so on) listed as a set of tiles. Their states and operational statistics are also displayed. To open a settings page of a component, click the name of the required component. As a rule, these components are started and stopped automatically when necessary. If any of them can be started or stopped manually by the user, the tile of the service component displays a switch for starting and stopping the component besides its name and operational statistics.

The bottom of the page displays information about the virus databases and the license. To force updating the virus databases, click **Update**. By clicking **Renew** (or **Activate license**, depending on the current state of your license) you can renew or activate your license by uploading a relevant key file for Dr.Web Gateway Security Suite to the server.

9.3.2. Threat Management

You can view a list of detected threats and manage a reaction to them on the **Threats** page.

This page displays the full list of threats detected by the components of Dr.Web Gateway Security Suite that monitor and scan the file system. At the top of the page, you can see a menu which allows filtration of threats on the basis of their categories:

- **All**—display all detected threats (including both active and quarantined threats);
- **Active**—display only active threats, that is, those that were detected but not neutralized yet;
- **Blocked**—display those threats that were not neutralized, but access to files containing them was blocked for users;
- **Quarantined**—display quarantined threats;
- **Errors**—display threats whose processing has finished with an error.

To the right of the name of each category in the menu, a number of detected threats that fall into this category is displayed. To display threats of a required category, click its name in the menu.



Threats detected by the components that scan network traffic (SpIDer Gate, Dr.Web ICAPD), and also by Dr.Web ClamD are not displayed on the **Threats** page. To trace the threats detected by these components, you can monitor threat counters and receive notifications available via SNMP (the Dr.Web SNMPD component gives access to threat counters and notifications of incidents according to the Dr.Web MIB structure).

For each threat, the following information is listed:

- **File**—name of a file that contains a malicious object (a file path is not specified);
- **Owner**—name of a user who owns the file with a threat;
- **Component**—name of a component that detected the threat;
- **Threat**—name of the threat detected in the file, as classified by the Doctor Web company.

For an object selected in the list, the following detailed information is displayed to the right of the list:

- name of the threat (displayed as a link that opens a page of the Dr.Web virus information library with a threat description);
- file size, in bytes;
- name of the component that detected the threat;
- date and time of threat detection;
- file last modification date and time;
- name of the user who owns the infected file;
- name of the group that includes the file owner;

- identifier that was assigned to the infected file (if the file was quarantined);
- full path to the file original location (at the moment of threat detection).

To select an object, click the corresponding line on the list. To select multiple objects, select the check boxes for the corresponding objects. To select all objects or cancel the selection at once, toggle the check box in the **File** field in the threat list header.

To apply actions to the objects selected on the list of threats, click the corresponding button on the toolbar that is located directly above the threat list. The toolbar contains the following buttons:

	Delete the selected files.
	Restore the selected files from quarantine to their original location.
	Apply an additional action to the selected files (available actions are specified in a drop-down list). The following additional actions are available: • **Quarantine**—quarantine the files with threats; • **Cure**—attempt to cure the threats; • **Ignore**—ignore the threats detected in the selected files and remove the threats from the list.

 Some buttons can be unavailable depending on the type of the selected threats.

The **Threats** page also allows you to filter displayed threats based on a search query. To filter out unnecessary threats and display only those that correspond to the query, use the search box. The box is displayed on the right side of the toolbar and contains . To filter the threat list, enter a word in the search box. At that, all threats that do not have the entered word in their name or description will be hidden (the search is case-insensitive). To clear search results and display the unfiltered list, click  in the search box or erase the word.

9.3.3. Managing Settings

In this section

- Viewing and editing component settings in a tabular form
- Viewing and editing component settings in the .ini editor
- Additional information

You can view and change current configuration parameters of the components included in Dr.Web Gateway Security Suite and listed on the main page. For that, open the **Settings** page. On this page, you can also switch Dr.Web Gateway Security Suite to a *centralized protection* mode or a *standalone* mode (for further information about these modes, refer to the Operation Modes section).

On the left side of the page, a menu is displayed that contains the names of all Dr.Web Gateway Security Suite components whose settings can be viewed and adjusted. To view and adjust the settings of a component, select its name from the menu by clicking it. The name of the component whose settings you are currently viewing and editing is highlighted in this menu to the left.

- The **Centralized protection** item of the menu takes you to the page for managing the centralized protection mode.
- The **General Settings** item of the menu corresponds to the settings of Dr.Web ConfigD, which is responsible for the overall functioning of Dr.Web Gateway Security Suite.

If a component has additional, specific sections of settings apart from a section with its main settings (for example, such sections are available for the Dr.Web ClamD component, which emulates the interface of ClamAV® and uses these additional sections to store individual scanning parameters for clients that use a specific connection address), then an icon indicating that you can collapse or expand additional (dependent) sections is displayed on the menu to the left of the component name. If the icon looks like ▸, additional sections are hidden. If the icon looks like ▾, additional sections are displayed on the menu, one per line. To collapse or expand the list of additional sections for a component, click the collapse/expand icon next to the name of the required component.

- The additional sections of the component settings are indented to the right. To view or edit parameters of an additional section, click its name.
- To add an additional subsection with settings for a component, if possible, click +. This icon is positioned to the right of the component name and appears when you point to the component name. Then specify a unique name (tag) for a new subsection and click **OK**. To close the window without creating a subsection, click **Cancel**.
- To remove a subsection, click ×. This icon is positioned to the right of the subsection name (tag) and appears when you point to the component name. Then, confirm that you want to remove the subsection by clicking **Yes** or cancel by clicking **No**.

At the top of the settings page, you can see a menu that allows you to change a viewing mode. The following modes are available:

- **All**—display all component configuration parameters that can be viewed and adjusted in a table editor;
- **Changed**—display only those component configuration parameters that have values different from the defaults in the table editor;
- **Ini Editor**—display component configuration parameters that have values different from the defaults in an editor for parameters in the configuration file format (in `parameter = value` pairs).

The settings management page also contains a panel for filtering the list of displayed parameters based on a search query. To filter the list of parameters and keep only those parameters whose description contains the given string, use the search bar. The bar is located on the right side of the menu controlling the display mode and contains . To filter the parameter list, enter a word in the search bar. At that, all parameters whose description does

not contain the specified word will be hidden (the search is case-insensitive). To clear search results and return to the initial parameter list, click in the search bar or delete the search word.

Parameters can be filtered only when they are displayed in tabular form (in **All** mode or **Changed** mode).

Viewing and editing component settings in a tabular form

When viewing parameters in tabular form (the **All** mode or the **Changed** mode), each table row contains a name and a description of a parameter (on the left) and its current value (on the right). For boolean parameters (those that have only two available values: “Yes” and “No”), a check box is displayed instead of a value (the selected check box means “Yes” and the cleared one means “No”).



When you view all parameters (and not only those that were changed), the modified (non-default) values are provided on the list in bold.

The complete parameter list is split into sections, such as **General**, **Advanced** and so on. To collapse or expand a table section, click its title. When the section is collapsed and its parameters are not displayed in the table, › is displayed to the left of the section name. When the section is expanded and its parameters are displayed in the table, ⌄ is displayed to the left of the section name.

To adjust a parameter, click its current value on the table (for a boolean parameter—select or clear the corresponding check box). If the parameter has a set of predefined values, clicking the current value opens a drop-down list in which a required value can be set. If the parameter has a numerical value, clicking the current value opens an editing field directly in the table. In this case, specify the required value and press ENTER. In all these cases the changed parameter value is immediately stored in the component settings.

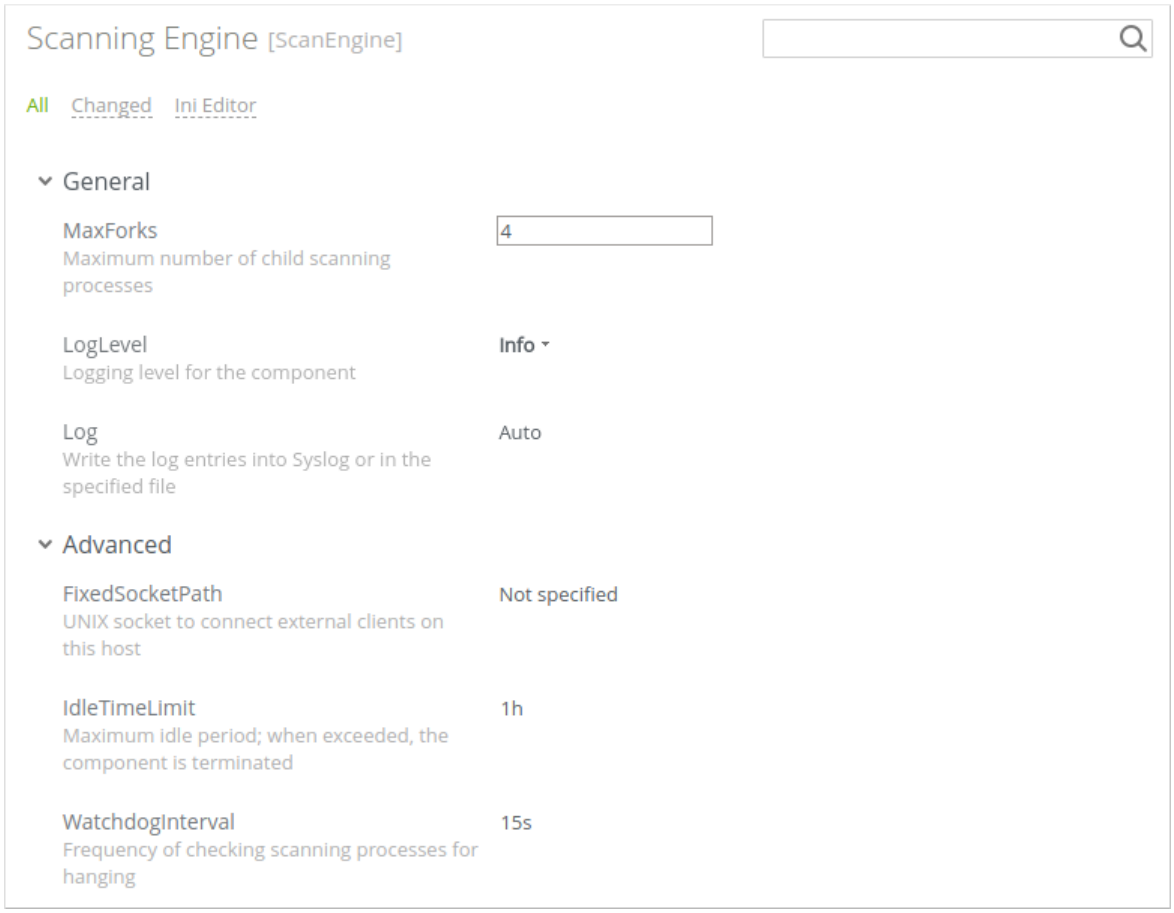


Figure 3. Component settings in tabular form

If the parameter accepts a string value or a list of arbitrary values, a pop-up window will appear once you click the parameter current value to edit it. If the parameter accepts a list of values, they are displayed in a multiline editing field (one value per line) as shown in the figure below. To edit the values of the list, change, delete or add required lines in the editing field.

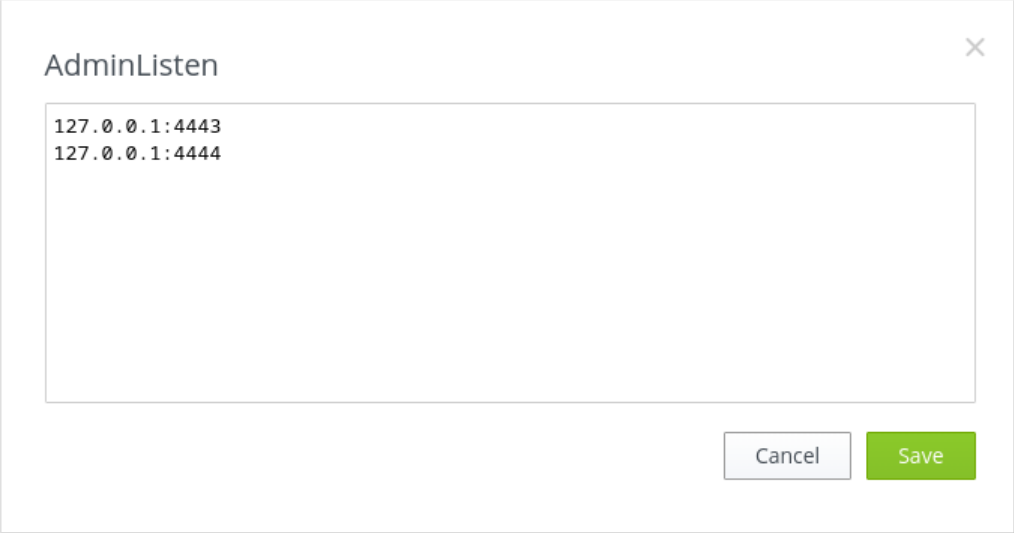


Figure 4. Editing a list of values

After editing a parameter value, click **Save** to save your changes and to close the window. To close the window without saving the changes, click **Cancel** or ✕ in the upper right corner of the pop-up window.

Viewing and editing component settings in the .ini editor

When viewing parameters in **Ini Editor** mode, they are displayed in an editor for parameters in the configuration file format (in `parameter = value` pairs), where `parameter` is a parameter name that is set directly in a component settings section of the configuration file. In this mode, only those parameters are displayed whose values differ from the defaults (that is, the parameters whose values are in bold in the **All** table). The figure below displays parameters in pairs.

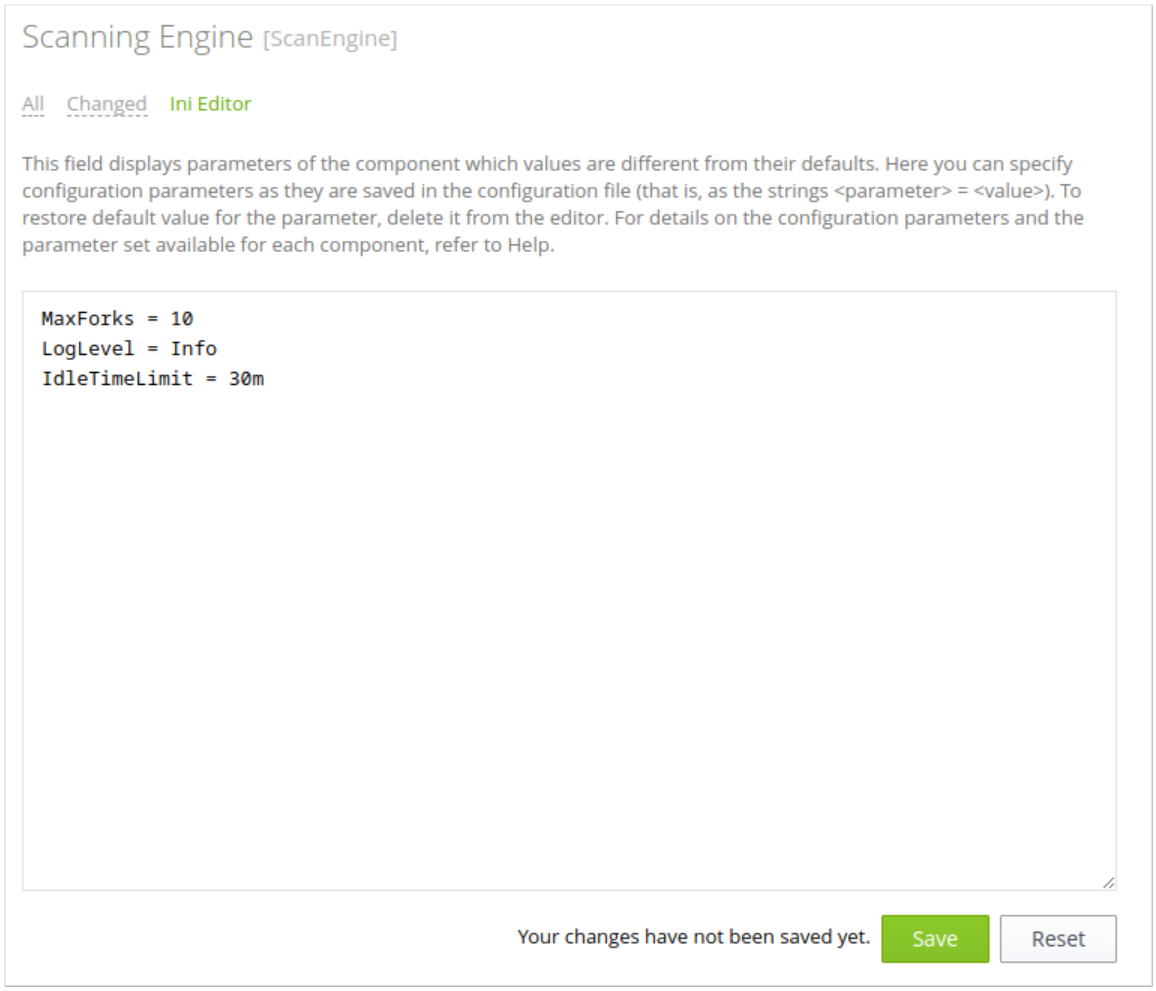


Figure 5. Editing settings in the configuration file format

To make changes, edit the text in accordance with the rules for editing the configuration file (this will modify only the section that contains the settings of the component selected to the left). If necessary, you can specify a new value for any parameter available for the component. In this case, the default value of this parameter changes to the value you enter in the editor. If you want to reset the parameter back to its default value, delete the line containing this parameter in this editor. If you do so, once you save the changes, the parameter is reset to its default value.

Once you have finished editing, click **Save** to save the changes. To discard the changes, click **Cancel**.

If you click **Save**, the text displayed in the editor is validated: the program checks whether all parameters are existent and their set values are valid. In case of an error, a corresponding message is displayed.

For details on the configuration file, its structure and aspects of specifying parameter values, refer to the Appendix D. Dr.Web Gateway Security Suite Configuration File section.

Additional information

- Configuration parameters of the Dr.Web ConfigD component (general settings).
- Configuration parameters of the Dr.Web ICAPD component.
- Configuration parameters of the SpIDer Gate component.
- Configuration parameters of the Dr.Web Firewall for Linux component.
- Configuration parameters of the Dr.Web ES Agent component.
- Configuration parameters of the Dr.Web Updater component.
- Configuration parameters of the Dr.Web ClamD component.
- Configuration parameters of the Dr.Web File Checker component.
- Configuration parameters of the Dr.Web Scanning Engine component.
- Configuration parameters of the Dr.Web Network Checker component.
- Configuration parameters of the Dr.Web SNMPD component.
- Configuration parameters of the Dr.Web CloudD component.
- Configuration parameters of the Dr.Web LookupD component.
- Configuration parameters of the Dr.Web StatD component.
- Managing Centralized Protection.

9.3.3.1. Managing Centralized Protection

You can connect Dr.Web Gateway Security Suite to a centralized protection server or disconnect from it, thereby switching the product to the standalone mode. To open a page where you can manage centralized protection, select **Centralized protection** from the settings menu on the **Settings** page.

To connect Dr.Web Gateway Security Suite to a centralized protection server or disconnect from it, use the corresponding check box on this page.

Connection to a centralized protection server

At an attempt to connect to a centralized protection server a pop-up window appears on the screen; in this window you need to specify the parameters for connecting to the centralized protection server.

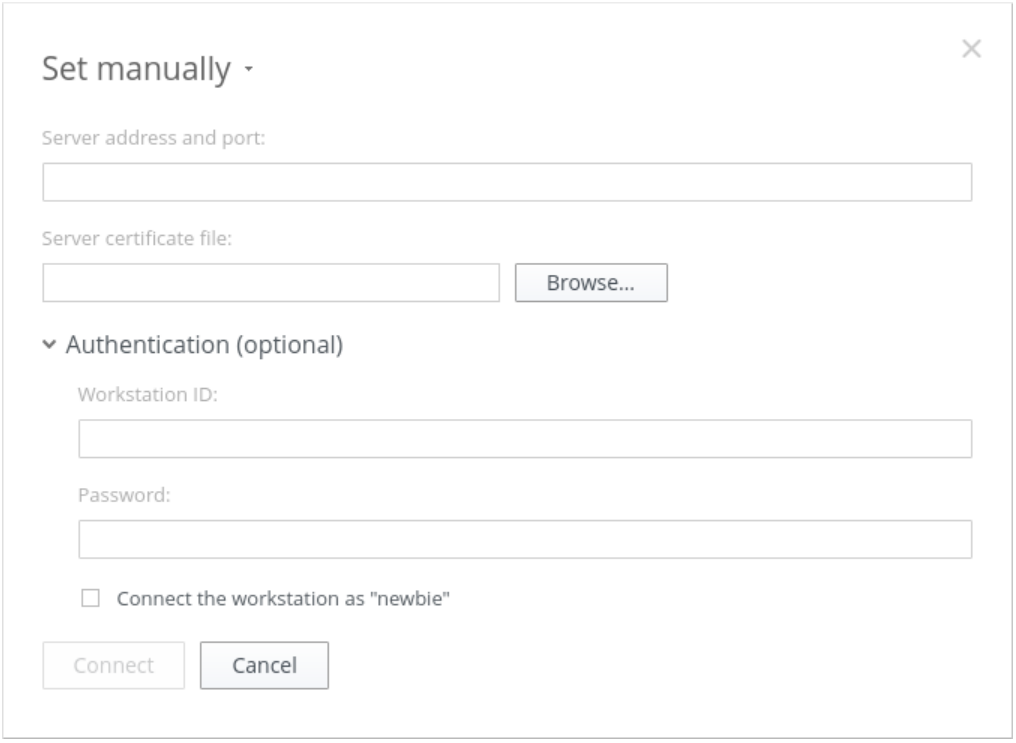


Figure 6. Connection to the centralized protection server

Select a method for connecting to the server from the drop-down list located at the top of the window. Three methods are available:

- *Load from file*,
- *Set manually*,
- *Detect automatically*.

If you select the *Load from file* option, specify a path to a file with connection settings in the corresponding field of the window. The file is provided by an antivirus network administrator. If you select the *Set manually* option, specify an address and a port to connect to the centralized protection server. In addition, for the *Set manually* or *Detect automatically* options, you can also specify a path to a file containing a server public key if you have one (usually, this file is provided by an antivirus network administrator or an internet service provider).

Furthermore, in the **Authentication (optional)** section you can specify a workstation identifier and a password for authentication on the server, if you know them. If these fields are filled in, your connection to the server will be established only if a correct identifier/password pair was provided. If you leave these fields empty, connection to the server is established only if it is approved by the server (either automatically or by the antivirus network administrator depending on the server settings).

Furthermore, you can select the **Connect the workstation as "newbie"** check box. If the newbie option is allowed on the server and the connection is approved, the server automatically generates a unique identifier/password pair to be further used for connecting your computer to this server.

If you connect as a newbie, the centralized protection server generates a new account for your computer even if it already had an account on this server.

Connection parameters must be specified in strict conformity with instructions provided by the administrator of your antivirus network or service provider.

To connect to the server, specify all the parameters, click **Connect** and wait for the connection to be established. To close the window without establishing the connection to the server, click **Cancel**.

Once you have connected Dr.Web Gateway Security Suite to the centralized protection server, its operation is managed by the centralized protection server until you switch back to the standalone mode. Connection to the server is established automatically every time Dr.Web Gateway Security Suite is started.

9.3.4. Scanning Local Files

In this section

- Opening a panel to scan local files and adjusting scanning parameters
- Starting a scan of local files

The web interface allows you to quickly scan files stored on your local computer used to access the web interface. The scanning engine included in Dr.Web Gateway Security Suite is used for scanning. The files selected for scanning are uploaded via HTTP to a server, but after the scanning, even if threats have been found, the files are not stored on the server and are not quarantined. The user who sent the files for scanning is only informed about its result.

Opening a panel to scan local files and adjusting scanning parameters

You can select and upload the files for scanning via a panel for scanning local files, which is displayed when you select the **Scan File** item on the main menu of the web interface. The activated panel is displayed in the bottom right corner of the web interface. The figure below shows what the panel for scanning local files looks like.

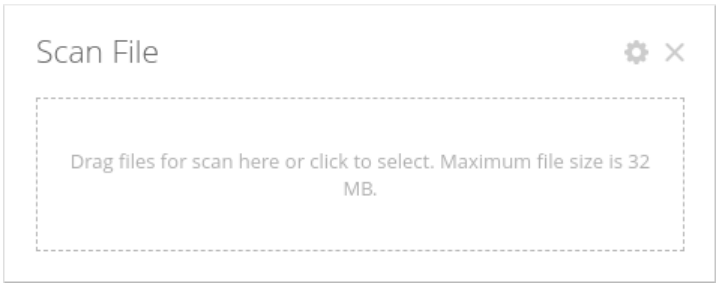


Figure 7. Panel for scanning local files

To close this panel, click ✕ in the top right corner of the panel. Click ⚙ to display the settings for the scanning of local files, where you can set the maximum time to scan a file (which does not include the time for uploading the file to the server from the local computer), enable or disable the heuristic analysis, and also set the maximum compression ratio for compressed objects and the maximum nesting level for objects packed into containers (such as archives).

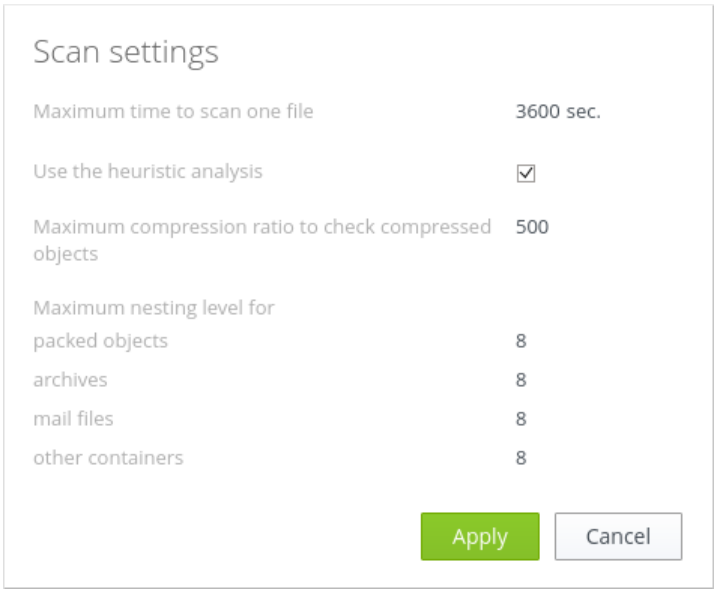


Figure 8. Adjusting the parameters for the scanning of local files

To apply the adjusted settings and to return to the file selection mode, click **Apply**. To go back to file selection without applying changes, click **Cancel**.

Starting a scan of local files

To start file scanning, click the target area **Drag files for scan here or click to select**; at that, a standard file chooser of your file manager opens. You can select multiple files at once for scanning.



You cannot select directories for scanning.

In addition, you can drag selected files with your mouse directly to the target area from the file manager window. Once the files to be scanned have been specified, they are uploaded to the server running Dr.Web Gateway Security Suite and scanned after uploading. During the uploading and scanning of the files, the file scanning panel displays the overall scanning progress.

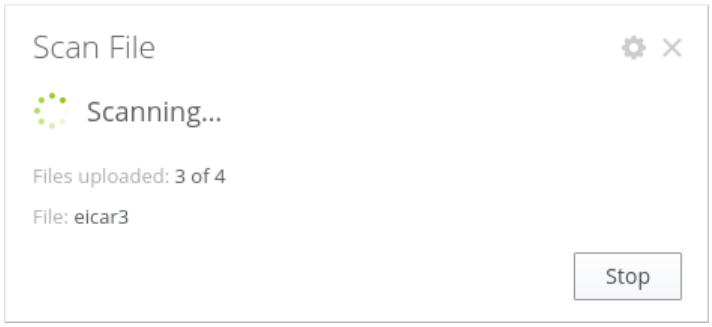


Figure 9. Scanning local files

If necessary, you can abort uploading and scanning by clicking **Stop**. Once the scanning is complete, a report about the scanning of the uploaded files is displayed on the panel.

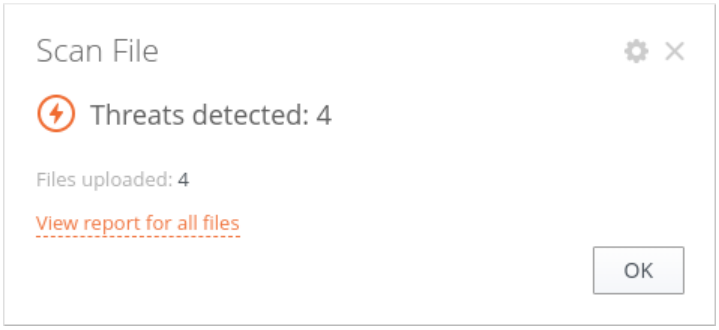


Figure 10. Result of the scanning of local files

If multiple files have been uploaded, an extended report about the scanning will be available. To view the extended report, click **View report for all files**.

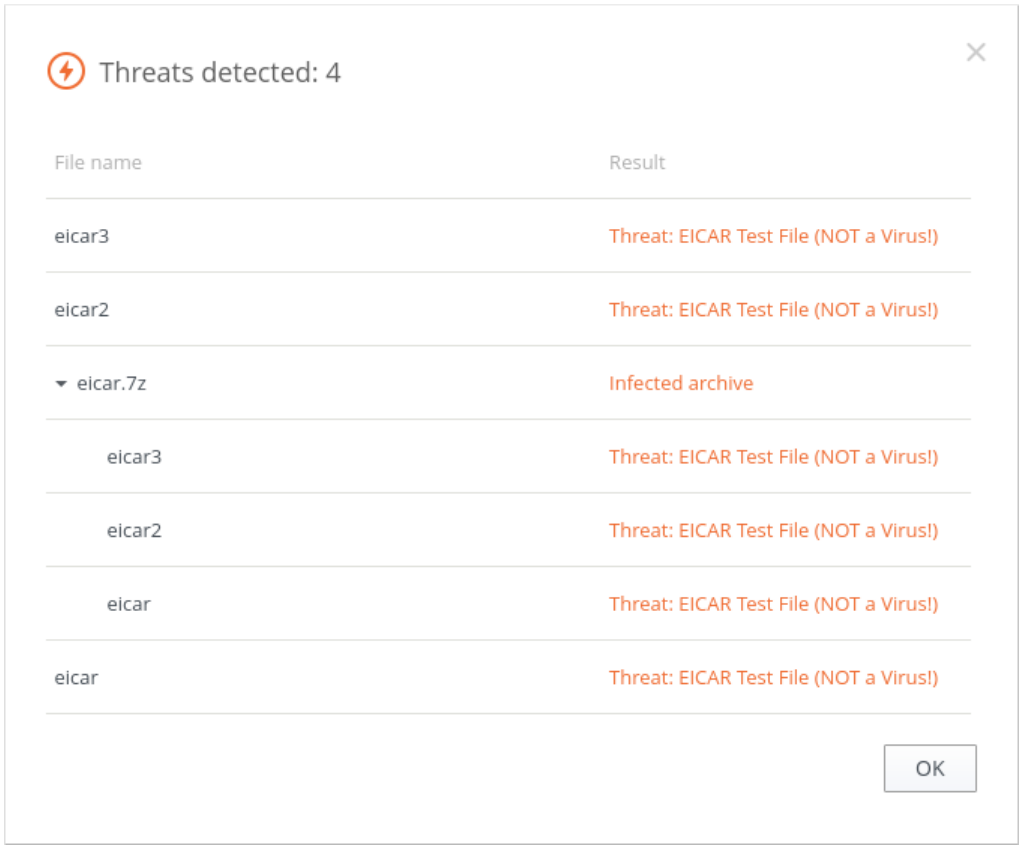


Figure 11. Extended report on scanned local files

To close the report and return to selecting new files for scanning, click **OK**.

You can start scanning local files with the current settings even if the file scanning panel is closed. To start uploading and scanning local files, just drag them from the file manager window to the web interface page opened in your browser.

9.4. Dr.Web ICAPD

The Dr.Web ICAPD component connects to an HTTP proxy server (such as Squid) via ICAP. The server is installed on a gateway that is used by LAN users to access the internet. The proxy server uses Dr.Web ICAPD as an external filter analyzing both user requests and server responses to these requests. If user access to any resource located on the external network must be blocked, or transmitted data (a user request or a server response) contains a threat or cannot be scanned because of an error, Dr.Web ICAPD instructs the proxy server to return a special HTML page generated by Dr.Web ICAPD from a template to the user.



If file scanning via HTTP is highly intensive, issues with scanning may arise when available file descriptors are depleted by the Dr.Web Network Checker component. In this case, it is necessary to increase the limit by the number of file descriptors available to Dr.Web Gateway Security Suite.

9.4.1. Operating Principles

The Dr.Web ICAPD component uses the ICAP protocol (the Internet Content Adaptation Protocol described in RFC 3507) to interact with a proxy server, which is external with respect to Dr.Web Gateway Security Suite and which handles HTTP/HTTPS connections of LAN hosts to web servers.

ICAP is a lightweight HTTP-like protocol. A client sends a request including headers and an encapsulated HTTP request to be scanned to an ICAP server. The ICAP server returns the modified encapsulated HTTP request and one or several headers in its response.

The following request types (methods) are used for interaction via ICAP:

- `REQMOD`—to verify and modify requests;
- `RESPMOD`—to verify and modify responses;
- `OPTIONS`—to obtain the information about ICAP server connection parameters.

`REQMOD` and `RESPMOD` requests use the following headers:

- `X-Client-IP`—IP address from which a verifiable HTTP request was sent;
- `X-Server-IP`—destination IP address at which a verifiable HTTP request was sent;
- `X-Client-Username`—name of the client authenticated on the proxy server (specified as `user` or `user@domain`).

`REQMOD` and `RESPMOD` responses use the following headers:

- `X-Response-Info` is included in the response if the requested web resource belongs to an unwanted or potentially dangerous category. Always has the `Blocked` value.
- `X-Infection-Found` contains information about the presence of viruses and other unwanted or potentially dangerous objects.

- `X-Virus-ID` contains the name of a detected threat.
- `X-Violations-Found` comprises the information about the errors that occurred during scanning.

The `OPTIONS` request may include the `DrWeb-Get-Scan-Status` header. If this header has the `Yes` value, the information about the version of Dr.Web Virus-Finding Engine, Dr.Web Scanning Engine and the virus bases will be returned in the response (see below).

The `OPTIONS` response includes the following headers:

- `X-Allow-Out`—list of header fields that Dr.Web ICAPD may include in responses;
- `X-Include`—values of the headers retrieved from a previously received request;
- `DrWeb-Core-Engine`—version of Dr.Web Virus-Finding Engine;
- `DrWeb-Scan-Engine`—version of Dr.Web Scanning Engine;
- `DrWeb-Scan-Status`—status of the current scanning operation;
- `DrWeb-Database-Timestamp`—virus base timestamp;
- `DrWeb-Virus-Records`—number of records in the base.

Furthermore, Dr.Web ICAPD can filter web content and block access to unwanted and potentially dangerous web resources. If the user requests such resource, a block page generated on the basis of the template is returned. The page contains the information on a reason of the blocking. The same page is returned when Dr.Web ICAPD detects a threat or an error occurs while scanning the data being transmitted.

To check whether a URL belongs to unwanted or potentially dangerous categories, Dr.Web ICAPD uses both the database of web resource categories, which is updated regularly from the Doctor Web update servers, and the Dr.Web Cloud service.

Potentially dangerous categories are the following:

- *InfectionSource*—websites containing malware (“infection sources”);
- *NotRecommended*—fraudulent websites (that use “social engineering”) visiting which is not recommended;
- *AdultContent*—websites that contain pornographic or erotic materials, dating websites and so on;
- *Violence*—websites that encourage violence or contain materials about various fatal accidents and so on;
- *Weapons*—websites dedicated to weapons and explosives or providing information on their manufacturing and so on;
- *Gambling*—websites that provide access to online games of chance, casinos, auctions, including websites for placing bets and so on;
- *Drugs*—websites that promote use, production or distribution of drugs and so on;
- *ObsceneLanguage*—websites that contain obscene language (in section titles, articles and so on);

- *Chats*—websites that offer real-time exchange of text messages;
- *Terrorism*—websites that contain aggressive and propaganda materials or description of terrorist attacks, and so on;
- *FreeEmail*—websites that offer a possibility of free registration of an email box;
- *SocialNetworks*—social networking services: general, professional, corporate, interest-based; thematic dating websites;
- *DueToCopyrightNotice*—websites links to which are provided by the copyright holders of some copyrighted work (movies, music, and so on);
- *OnlineGames*—websites that provide access to games using a permanent internet connection;
- *Anonymizers*—websites allowing the user to hide personal information and providing access to blocked websites;
- *CryptocurrencyMiningPool*—websites that provide access to services for cryptocurrency mining;
- *Jobs*—job search websites.

In the settings of the component, the system administrator can specify unwanted categories of web resources and create black and white lists. Requesting resources included in a black list will result in returning a block page. Access to a web resource included in a white list is always allowed, even if such resource belongs to an unwanted category.



The same website can belong to multiple categories. User access to such website will be blocked if it belongs to at least one unwanted category.

Even if a website is included in the white list, data (sent and downloaded from the website) is scanned for threats.

Due to aspects of the ICAP protocol, the scanning of large data chunks (`.iso` images, large archives, video files, and so on) can take a long time. It is recommended that you limit MIME types of data to be scanned, as well as limit the maximum size of data for scanning via the ICAP protocol in the HTTP proxy server settings (see an example for a Squid proxy server).

The same Dr.Web Updater component regularly and automatically updates databases of web resource categories and virus databases for Dr.Web Scanning Engine from Doctor Web servers. The Dr.Web Cloud service is mediated by the Dr.Web CloudD component (using the cloud service is configured in the general settings of Dr.Web Gateway Security Suite and can be disabled, if necessary). To scan data being transmitted, Dr.Web ICAPD uses a network scanning component, Dr.Web Network Checker, which initiates data scanning via Dr.Web Scanning Engine.

To block or pass HTTP requests and responses, the Dr.Web ICAPD component can use rules defined in the settings and a Lua script.



See the Integration with a Squid Proxy Server section for information about integrating Dr.Web ICAPD with an HTTP proxy server.

9.4.2. Command-Line Arguments

To start the Dr.Web ICAPD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-icapd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-icapd [<parameters>]
```

Dr.Web ICAPD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-icapd --help
```

This command outputs short help information about the Dr.Web ICAPD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-icapd` command.

9.4.3. Configuration Parameters

In this section

- Component parameters
- Traffic monitoring and access blocking rules
  - Viewing and editing the rules
  - Viewing the rules using the drweb-ctl cfshow command
  - Editing the rules using the drweb-ctl cfset command
  - Default set of rules
  - Examples of the traffic monitoring and access blocking rules

The component uses configuration parameters specified in the `[ICAPD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

Component parameters

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-icapd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-icapd`
`Start` *{boolean}*	The component is started by the Dr.Web ConfigD configuration management daemon. Setting this parameter to `Yes` instructs the configuration management daemon to start the component immediately, and setting this parameter to `No`—to shut down the component immediately. Default value: `No`

Parameter	Description
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`DebugDumpIcap` *{boolean}*	Log or do not log messages transmitted via ICAP at the debug level (with `LogLevel = DEBUG`). Default value: `No`
`ListenAddress` *{network socket}*	Network socket (IP address and port) listened by Dr.Web ICAPD for connections from HTTP proxy servers. Default value: `127.0.0.1:1344`
`TemplatesDir` *{path to directory}*	Path to a directory with the templates for the HTML notifications sent upon blocking a web resource. Default value: • for GNU/Linux: `/var/opt/drweb.com/templates/icapd` • for FreeBSD: `/var/drweb.com/templates/icapd`

`Whitelist`
{domain list}
	White list of domains. All domains included in this list will be accessible to users even if belonging to unwanted web resource categories. All subdomains of these domains will also be accessible.
	
	The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	
	Example: Add `example.com` and `example.net` domains to the list.
	
	1. Adding values to the configuration file.
	• Two values per line:
	`[ICAPD]`
	`Whitelist = "example.com", "example.net"`
	• Two lines (one value per line):
	`[ICAPD]`
	`Whitelist = example.com`
	`Whitelist = example.net`

Parameter	Description

	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset ICAPD.Whitelist -a example.com`
	`# drweb-ctl cfset ICAPD.Whitelist -a example.net`
	⚠ Actual usage of the domain list indicated in this parameter depends on *how* this list is used in the rules for controlling access to web resources defined for Dr.Web ICAPD.
	The list of default rules (see below) guarantees that access to domains (and their subdomains) from this list will be granted even if it contains domains from the list of blocked web resource categories. Furthermore, the default set of rules guarantees that data downloaded from the white list domains *will be scanned for threats*.
	Default value: *(not specified)*

`Blacklist`
{domain list}
	Black list of domains. All domains included in this list will not be accessible to users even if not belonging to unwanted web resource categories. All subdomains of these domains will also be blocked.
	The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	Example: Add `example.com` and `example.net` domains to the list.
	1. Adding values to the configuration file.
	• Two values per line:
	`[ICAPD]`
	`Blacklist = "example.com", "example.net"`
	• Two lines (one value per line):
	`[ICAPD]`
	`Blacklist = example.com`
	`Blacklist = example.net`

Parameter	Description

	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset ICAPD.Blacklist -a example.com`
	`# drweb-ctl cfset ICAPD.Blacklist -a example.net`
	
	Actual usage of the domain list indicated in this parameter depends on *how* this list is used in the rules for controlling access to web resources defined for Dr.Web ICAPD.
	The list of default rules (see below) guarantees that access to domains (and their subdomains) from this list will always be blocked. If a domain is added simultaneously to `Whitelist` and `Blacklist`, the default rules guarantee that user access to it will be blocked.
	Default value: *(not specified)*

`Adlist`
{list of strings}
	List of regular expressions covering websites. A URL that matches any of the regular expressions from this list is considered to be the advertising URL.
	The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	Example: Add the '`.*(?:\.|\/\/)ads\..+`' and '`.*/ad/.*\.gif$`' expressions to the list.
	1. Adding values to the configuration file.
	• Two values per line:
	`[ICAPD]`
	`Adlist = ".*(?:\.|\/\/)ads\..+", ".*/ad/.*\.gif$"`
	• Two lines (one value per line):
	`[ICAPD]`
	`Adlist = .*(?:\.|\/\/)ads\..+`
	`Adlist = .*/ad/.*\.gif$`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset ICAPD.Adlist -a '.*(?:\.|\/\/)ads\..+'`
	`# drweb-ctl cfset ICAPD.Adlist -a '.*/ad/.*\.gif$'`

Parameter	Description
	Regular expressions are specified using either the POSIX syntax (BRE or ERE) or the Perl syntax (PCRE or PCRE2). Actual usage of the list of the expressions indicated in this parameter depends on *how* this list is used in the rules for controlling access to web resources defined for Dr.Web ICAPD. The list of default rules (see below) guarantees that access to a URL from this list will be blocked only if the domains of these URLs are not included in `Whitelist`. Default value: *(not specified)*
`BlockInfectionSource` *{boolean}*	Block attempts to connect to websites containing malware (belonging to the *InfectionSource* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `Yes`
`BlockNotRecommended` *{boolean}*	Block attempts to connect to non-recommended websites (belonging to the *NotRecommended* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `Yes`
`BlockAdultContent` *{boolean}*	Block attempts to connect to websites containing adult content (belonging to the *AdultContent* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockViolence` *{boolean}*	Block attempts to connect to websites containing graphic violence (belonging to the *Violence* category).

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockWeapons` *{boolean}*	Block attempts to connect to websites dedicated to weapons (belonging to the *Weapons* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockGambling` *{boolean}*	Block attempts to connect to gambling websites (belonging to the *Gambling* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockDrugs` *{boolean}*	Block attempts to connect to websites dedicated to drugs (belonging to the *Drugs* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockObsceneLanguage` *{boolean}*	Block attempts to connect to websites with obscene language (belonging to the *ObsceneLanguage* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockChats` *{boolean}*	Block attempts to connect to chat websites (belonging to the *Chats* category).

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockTerrorism` *{boolean}*	Block attempts to connect to websites dedicated to terrorism (belonging to the *Terrorism* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockFreeEmail` *{boolean}*	Block attempts to connect to websites of free email services (belonging to the *FreeEmail* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockSocialNetworks` *{boolean}*	Block attempts to connect to social networking websites (belonging to the *SocialNetworks* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockDueToCopyrightNotice` *{boolean}*	Block attempts to connect to websites that were added at the request of a copyright holder (belonging to the *DueToCopyrightNotice* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `Yes`
`BlockOnlineGames` *{boolean}*	Block attempts to connect to online gaming websites (belonging to the *OnlineGames* category).

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockAnonymizers` *{boolean}*	Block attempts to connect to anonymizer websites (belonging to the *Anonymizers* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockCryptocurrencyMiningPools` *{boolean}*	Block attempts to connect to websites combining users to mine cryptocurrencies (belonging to the *CryptocurrencyMiningPool* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockJobs` *{boolean}*	Block attempts to connect to job search websites (belonging to the *Jobs* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "ICAPD.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockKnownVirus` *{boolean}*	Block incoming and outgoing data if it contains a known threat. To enable blocking, the settings must contain the following rule (see below): `threat_category in "ICAPD.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockSuspicious` *{boolean}*	Block incoming and outgoing data if it contains an unknown threat detected by the heuristic analyzer.

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `threat_category in "ICAPD.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockAdware` *{boolean}*	Block incoming and outgoing data if it contains adware. To enable blocking, the settings must contain the following rule (see below): `threat_category in "ICAPD.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockDialers` *{boolean}*	Block incoming and outgoing data if it contains a dialer. To enable blocking, the settings must contain the following rule (see below): `threat_category in "ICAPD.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockJokes` *{boolean}*	Block incoming and outgoing data if it contains a joke program. To enable blocking, the settings must contain the following rule (see below): `threat_category in "ICAPD.BlockThreat" : BLOCK as _match` Default value: `No`
`BlockRiskware` *{boolean}*	Block incoming and outgoing data if it contains riskware. To enable blocking, the settings must contain the following rule (see below): `threat_category in "ICAPD.BlockThreat" : BLOCK as _match` Default value: `No`
`BlockHacktools` *{boolean}*	Block incoming and outgoing data if it contains a hack tool. To enable blocking, the settings must contain the following rule (see below): `threat_category in "ICAPD.BlockThreat" : BLOCK as _match` Default value: `No`

Parameter	Description
`ScanTimeout` *{time interval}*	Timeout for scanning one file at the request of Dr.Web ICAPD. Allowed values: from 1 second (`1s`) to 1 hour (`1h`). Default value: `30s`
`HeuristicAnalysis` *{On | Off}*	Enable or disable the heuristic analysis for detection of unknown threats. The heuristic analysis provides higher detection reliability but increases the duration of scanning. Action applied to threats detected by the heuristic analyzer is specified by the `BlockSuspicious` parameter. Allowed values: • `On`—enable the heuristic analysis during a scan; • `Off`—disable the heuristic analysis. Default value: `On`
`PackerMaxLevel` *{integer}*	Maximum nesting level for packed objects. A packed object is executable code compressed with special software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects that may also include packed objects and so on. The value of this parameter specifies a nesting limit beyond which packed objects inside other packed objects are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`ArchiveMaxLevel` *{integer}*	Maximum nesting level for archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`MailMaxLevel` *{integer}*	Maximum nesting level for mailer files (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned.

Parameter	Description
	Default value: `8`
`ContainerMaxLevel` *{integer}*	Maximum nesting level while scanning other types of objects containing nested objects (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects will not be scanned. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`MaxCompressionRatio` *{integer}*	Maximum compression ratio of compressed/packed objects (a ratio between a compressed size and an uncompressed size). If the ratio for an object exceeds the limit, this object is skipped during the data scanning initiated by Dr.Web ICAPD. The compression ratio must be no less than `2`. Default value: `500`
`BlockUnchecked` *{boolean}*	Block incoming and outgoing data if it cannot be scanned. Default value: `No`
`UsePreview` *{boolean}*	Enable or disable an *ICAP preview* mode for Dr.Web ICAPD. *Do not change* the value of this parameter unless necessary. Default value: `Yes`
`Use204` *{boolean}*	Return response code `204` not only in *ICAP preview* mode. *Do not change* the value of this parameter unless necessary. Default value: `Yes`
`AllowEarlyResponse` *{boolean}*	Allow or do not allow Dr.Web ICAPD to use an ICAP early response mode, that is, start sending a response to the client before the entire request has been received from the HTTP proxy server. *Do not change* the value of this parameter unless necessary. Default value: `Yes`
`MessageHook` *{path to file | Lua function}*	Script for processing HTTP messages in the Lua language or a path to the script file (see the HTTP Message Processing in Lua section). If the Lua function or the file path are not specified, the messages will be processed in accordance with the rules. If the specified file is not available, the component will fail to start.

Parameter	Description

	Default value: *Generated automatically*. In case of default settings, the script is as follows:
	`local dw = require "drweb"`
	`local cfg = require "drweb.config"`
	`local dwl = require "drweb.lookup"`
	`local rx = require "drweb.regex"`
	
	`function message_hook(ctx)`
	`  if ctx.direction == "request" then`
	`  local url = ctx.request.url`
	`  if url.in_list(cfg.blacklist) then`
	`  return "block"`
	`  end`
	`  if not url.in_list(cfg.whitelist) then`
	`  if rx.search(cfg.adlist, url) or rx.search(cfg.adlist, url.raw) then`
	`  return "block"`
	`  end`
	`  if url.in_categories(cfg.block_url_categories) then`
	`  return "block"`
	`  end`
	`  end`
	`  end`
	`  if ctx.body.has_threat{category = cfg.block_threats} then`
	`  return "block"`
	`  end`
	`  if cfg.block_unchecked and ctx.body.scan_error then`
	`  return "block"`
	`  end`
	`  return "pass"`
	`end`

Traffic monitoring and access blocking rules

In addition to the parameters listed above, the section contains seven *sets of rules* `RuleSet*` (`RuleSet0`, ..., `RuleSet6`) which directly control traffic scanning and blocking of user access to web resources, as well as blocking downloading content from the internet. For some values in conditions (for example, IP address ranges, lists of website categories, black and white lists of web resources and so on), there is a substitution of values loaded from text files as well as extracted from external data sources via LDAP (the Dr.Web LookupD component is used). When configuring connections, all rules are processed from first to last as a single list until the rule that triggered the ultimate resolution is found. The gaps in the rule list, if any, are ignored.

The rules are described in detail in the Traffic Monitoring Rules section of Appendix D.

Viewing and editing the rules

List gaps, that is, `RuleSet<i>` sets that do not contain rules (wherein <*i*> is a `RuleSet` rule set number), are kept for editing the rules easily.



You cannot add items different to the already present `RuleSet<i>` items, but you can add or remove any rule in any `RuleSet<i>` item.

The rules can be viewed and edited in any of the following ways:

- by viewing and editing the configuration file in any text editor (note that this file stores only those parameters whose values differ from the defaults);
- using management web interface;
- via the command-line interface—Dr.Web Ctl (`drweb-ctl cfshow` and `drweb-ctl cfset` commands).



If you have edited the rules and made changes to the configuration file, in order to apply these changes, restart Dr.Web Gateway Security Suite. To do that, run the `drweb-ctl reload` command.

Viewing the rules using the drweb-ctl cfshow command

To view the contents of the `ICAPD.RuleSet1` rule set, run the command:

```
# drweb-ctl cfshow ICAPD.RuleSet1
```

Editing the rules using the drweb-ctl cfset command

Hereinafter <*rule*> is the text of the rule.

- Replace all the rules in the `ICAPD.RuleSet1` set with a new rule:

```
# drweb-ctl cfset ICAPD.RuleSet1 '<rule>'
```

- Add a new rule to the `ICAPD.RuleSet1` rule set:

```
# drweb-ctl cfset -a ICAPD.RuleSet1 '<rule>'
```

- Remove a specific rule from the `ICAPD.RuleSet1` rule set:

```
# drweb-ctl cfset -e ICAPD.RuleSet1 '<rule>'
```

- Reset the `ICAPD.RuleSet1` rule set to its default values:

```
# drweb-ctl cfset -r ICAPD.RuleSet1
```

When you use the `drweb-ctl` tool to edit the rules, put the text of the <*rule*> rule to be added in single or double quotes and use a backward slash (`\`) to escape quotes within the text of the rule.



It is important to remember the following aspects of storing the rules in the `RuleSet<i>` configuration variables:

- The conditional part and colon can be omitted when adding unconditional rules. However, such rules are always stored in the list of rules as the `' : <action>'` string.
- When adding rules that contain several actions (such rules as `'<condition> : <action 1>, <action 2>'`), such rules will be transformed into a chain of elementary rules `'<condition> : <action 1>'` and `'<condition> : <action 2>'`.
- The rules do not allow for disjunction (logical "OR") of conditions in the conditional part, so, in order to implement the logical "OR", construct a chain of rules with each rule having a disjunct-condition in its condition.

To add an unconditional skipping rule (the `PASS` action) to the `ICAPD.RuleSet1` rule set, run the command:

```
# drweb-ctl cfset -a ICAPD.RuleSet1 'PASS'
```

To remove this rule from the specified rule set, run the command:

```
# drweb-ctl cfset -e ICAPD.RuleSet1 ' : PASS'
```

To add a rule to the `ICAPD.RuleSet1` rule set to change a path to standard templates for connections from forbidden addresses and block these connections, run the command:

```
# drweb-ctl cfset -a ICAPD.RuleSet1 'src_ip not in file("/etc/trusted_ip") : set
http_template_dir = "mytemplates", BLOCK'
```

This command adds *two rules* to the specified rule set, so, in order to remove these rules, run these two commands:

```
# drweb-ctl cfset -e ICAPD.RuleSet1 'src_ip not in file("/etc/trusted_ip") : set
http_template_dir = "mytemplates"'
# drweb-ctl cfset -e ICAPD.RuleSet1 'src_ip not in file("/etc/trusted_ip") : BLOCK'
```

To add such rule as "Block upon detecting a malicious object of the *KnownVirus* type or a URL from the *Terrorism* category" to the `ICAPD.RuleSet1` rule set, add the following two rules:

```
# drweb-ctl cfset -a ICAPD.RuleSet1 'threat_category in (KnownVirus) : BLOCK as _match'
# drweb-ctl cfset -a ICAPD.RuleSet1 'url_category in (Terrorism) : BLOCK as _match'
```

To remove them, run two commands, as shown in the example above.

Default set of rules

By default, the following set of rules for blocking is specified:

```
RuleSet0 =
RuleSet1 = direction request, url_host in "ICAPD.Blacklist" : BLOCK as BlackList
RuleSet1 = direction request, url_host not in "ICAPD.Whitelist", url match
"ICAPD.Adlist" : BLOCK as BlackList
RuleSet2 =
RuleSet3 = direction request, url_host not in "ICAPD.Whitelist", url_category in
"ICAPD.BlockCategory" : BLOCK as _match
```

```
RuleSet4 =
RuleSet5 = threat_category in "ICAPD.BlockThreat" : BLOCK as _match
RuleSet6 =
```

The first two rules cover processing of outgoing HTTP connections: if a host (or a URL) to which a connection is being established is included in the black list, the connection is blocked on the black list basis, further checks are not performed. If the host (the URL) is not included in the white list and belongs to any unwanted category of websites or matches the list of the regular expressions covering advertisement websites, then the connection is blocked because the URL belongs to the prohibited category.

The rule specified in `RuleSet5` checks whether the body of the HTTP request or HTTP response contains threats that belong to the categories to be blocked, and, if so, the connection is blocked on the basis of detecting the threat.



Since the `direction` condition is not specified, both client requests and server responses are scanned by default.

Examples of the traffic monitoring and access blocking rules

1. Allow users with IP addresses in the range of *10.10.0.0–10.10.0.254* to access websites of all categories except for *Chats*:

   ```
   src_ip in (10.10.0.0/24), url_category not in (Chats) : PASS
   ```

   If the rule:

   ```
   url_host in "ICAPD.Blacklist" : BLOCK as BlackList
   ```

   is put on the list of rules above the indicated rule, then access to the domains from the black list, that is, the domains listed in the `ICAPD.Blacklist` parameter, will also be blocked for users with IP addresses in the range of *10.10.0.0–10.10.0.254*. At the same time, if this rule is put below, the users with the IP addresses in the range of *10.10.0.0–10.10.0.254* will also get access to websites from the black list.

   Since the `PASS` resolution is final, no more rules are processed; therefore, the downloaded data is not scanned for threats.

   To allow users with IP addresses in the range of *10.10.0.0–10.10.0.254* to access websites of all categories except *Chats*, if they are not on the black list, and to block downloading of threats at the same time, use the following rule:

   ```
   url_category not in (Chats), url_host not in "ICAPD.Blacklist", threat_category not
   in "ICAPD.BlockCategory" : PASS
   ```

2. Do not scan the contents of video files (that is, data of the `video/*` MIME type, where `*` corresponds to any type of the `video` MIME class):

   ```
   content_type in ("video/*") : PASS
   ```

9.4.4. HTTP Messages Processing in Lua

In this section

- General information
- Requirements for the message processing script
- Tables in use (MessageContext, ICAP, User, HeaderField, Request, ContentType, Url, RawUrl, UrlCategoryFilter, Response, Body, Virus, ThreatFilter)
- Available service modules (drweb, drweb.lookup, drweb.regex, drweb.config)

General information

The Dr.Web ICAPD component supports interaction via the Lua program interpreter (version 5.3.4 supplied with Dr.Web Gateway Security Suite is used). Scripts written in Lua can be used by the component for the analysis and processing of the HTTP protocol messages.

The analysis of an HTTP message (request or response) received for scanning from a proxy server via the ICAP protocol is performed by means of the Lua script specified in the Dr.Web ICAPD settings as the value of the `MessageHook` parameter (either script code or a path to the script file can be provided).

Requirements for the message processing script

The script must contain a global function which is an entry point to the message scanning module (Dr.Web ICAPD will call this function for processing a newly received message). The processing function must match the following call conventions:

1. *Function name* is `message_hook`.
2. *The only argument* is the MessageContext table (provides access from the function to the information about the processed email message).
3. *The single return value* is a string. The return value determines a verdict for the scanned message, i.e. to skip it or to block it. Allowed values:
   - `pass`—the HTTP message will be passed to the recipient (the HTTP request—to the server, the HTTP response—to the client).
   - `block`—the HTTP message will not be passed to the recipient, the client will receive an HTTP response with a blocking webpage.

A situation when the function returns a different value or an error occurs when executing the function is treated as a scanning error; the response to the client depends in this case on the value of the `BlockUnchecked` parameter.

Below is an example of a correct definition of the function, which will always return the `pass` verdict to Dr.Web ICAPD for all HTTP messages received for scanning (the `ctx` argument is hereinafter an instance of the *MessageContext* table):

```
function message_hook(ctx)
  return "pass"
end
```

The script from the next example will block access to all resources except Doctor Web documentation website for all users except the members of the Web Admins group in Active Directory:

```
local dwl = require "drweb.lookup"

function message_hook(ctx)
  -- Do not block access to resources at the documentation website
  -- of Doctor Web
  if ctx.request.url.in_list{"download.geo.drweb.com"} then
      return "pass"
  end

  -- Allow access to users from the WebAdmins group
  -- in the Active Directory section
  if dwl.check("WebAdmins", "AD@WinRoot", ctx.icap.user) then
      return "pass"
  end

  -- Block access (to all resources) for everyone else
  return "block"

end
```

Tables in use

MessageContext table

It is used as an input argument of the `message_hook` function. It provides access to the information about the processed HTTP message (its type, headers, body, information about the sender and the recipients, if available).

Field	Description	Data type
`direction`	HTTP message type: • `request`—an HTTP request. • `response`—an HTTP response.	String
`icap`	Information about ICAP request headers.	ICAP table
`request`	Information about HTTP request headers.	Request table
`response`	Information about HTTP request headers.	Response table

Field	Description	Data type
`body`	Information about the HTTP message body.	Body table
Overridden metamethods: *None*		

ICAP table

The table is used as the `icap` field of the MessageContext table. It contains the information about an ICAP request from an HTTP proxy server.

Field	Description	Data type
`user`	Information about the user obtained from the `X-Client-Username` header of the ICAP request.	User table
`src`	IP address of a client who sent the request (the address is contained in the `X-Client-IP` header of the ICAP request from the proxy server), or `nil`, if the address is unknown.	IpAddress table
`field`	Array of ICAP request headers.	Array of HeaderField tables
`search`	Function for searching a header by a regular expression. It accepts one mandatory argument—`patterns`—search patterns, i.e. one (string) or several (array of strings) regular expressions in the Perl syntax (PCRE). The search is performed in all available headers. ⚠ When using quoted strings, the slash character must be escaped. Returns a Boolean value: • `true`—if the `field.name .. ": " .. field.value.decoded` string matches at least one of the specified regular expressions for at least one of the headers; • `false`—if no matches have been found.	Function
`value`	Function that accepts one mandatory argument—the header name (string). It returns the value of the first header with the specified name or `nil`, if there is no header with this name.	Function
Overridden metamethods: *None*		

User table

The table contains the name and the domain of the user; both fields are optional.

Field	Description	Data type
`user`	User name	String
`domain`	User domain	String
Overridden metamethods: • `__tostring`—function that returns the `User` content as a string (in the UTF-8 encoding); • `__concat`—function that concatenates the `User` string value and another string.		

HeaderField table

The table describes HTTP or ICAP messages.

Field	Description	Data type
`name`	Header name	String
`value`	Header value	String
Overridden metamethods: *None*		

Request table

The table describes the headers of an HTTP request.

Field	Description	Data type
`method`	HTTP protocol method used in the request (for example, `POST`) or `nil`, if the ICAP request does not contain the HTTP request header.	String
`url`	URL of the resource at which the HTTP request is sent.	Url table
`content_type`	Information from the `Content-Type` header of the HTTP request.	ContentType table
`field`	Array of HTTP request headers.	Array of HeaderField tables
`search`	Function for searching a header by a regular expression. It accepts one mandatory argument—`patterns`—search patterns, i.e. one (string) or several (array of strings) regular expressions in the Perl syntax (PCRE). The search is performed in all available headers.	Function

Field	Description	Data type
	 When using quoted strings, the slash character must be escaped. Returns a Boolean value: • `true`—if the `field.name .. ": " .. field.value.decoded` string matches at least one of the specified regular expressions for at least one of the headers; • `false`—if no matches have been found.	
`value`	Function that accepts one mandatory argument—the header name (string). It returns the value of the first header with the specified name or `nil`, if there is no header with this name.	Function
Overridden metamethods: *None*		

ContentType table

The table describes a value obtained from the `Content-Type` header.

Field	Description	Data type
`type`	MIME type of the message part	String
`subtype`	Subtype of the message part	String
`param`	Header parameters in the form of a table array with the following fields: • `name` is the name of a parameter (string); • `value` is the value of a parameter (string).	Table array
`match`	Function that accepts one mandatory argument—`media_types`, which is an array of strings describing MIME types. Each string in the list must be formatted as follows: `"type/subtype"`, `"type/*"` or `"*/*"`. Returns a Boolean value: • `true`—if the MIME type of the body matches any of the specified strings (case-insensitively), or the passed array contains the `"*/*"` string; • `false`—if no matches have been found.	Function
Overridden metamethods: • `__tostring`—function that returns a decoded header value; • `__concat`—function that concatenates the decrypted value of the header and a string.		

Url table

The table describes a URL address.

Field	Description	Data type
`scheme`	Scheme (protocol) prefix, for example, `http`. If absent, the value is `nil`.	String
`host`	Host name or IP address, for example, `example.com`. If absent, the value is `nil`.	String
`port`	Port number, for example, `80`. If absent, the value is `nil`.	Number
`path`	Path to a resource, for example, `index.html`. If absent, the value is `nil`.	String
`query`	Decoded request parameters. If absent, the value is `nil`.	String
`legal_url`	If the URL belongs to the `owners_notice` category, the field contains a URL of the owner's website; otherwise, it is `nil`.	String
`in_list`	Function that accepts one mandatory argument—`hosts`, i.e. the host list (an array of strings). It returns a Boolean value: • `true`—if `host` is a subdomain of one of the specified domains or matches one of them; • `false`—if no matches have been found.	Function
`categories`	Function that accepts one optional argument—`filter`, i.e. the UrlCategoryFilter table (no argument is equivalent of using an empty table). It returns an iterator function using which you can iterate over all categories that meet the conditions specified by `filter` and to which URL relates.	Function
`in_categories`	Function that accepts one mandatory argument—`categories`, i.e. the URL category list (an array of strings). It returns a Boolean value: • `true`—if the URL belongs to at least one of the specified categories; • `false`—if the URL does not belong to any of the categories. If the `categories` array is empty, `false` is always returned. See the possible category values in the description of the `category` field in the UrlCategoryFilter table.	Function
`raw`	Raw undecoded URL.	RawUrl table
Overridden metamethods:		

Field	Description	Data type
• `__tostring`—function that returns the `Url` content as a string (in the UTF-8 encoding); • `__concat`—function that concatenates the `Url` string value and another string.		

RawUrl table

The table contains undecoded information about the URL.

Field	Description	Data type
`scheme`	Scheme (protocol) prefix, for example, `http`. If absent, the value is `nil`.	String
`host`	Host name or IP address, for example, `example.com`. If absent, the value is `nil`.	String
`port`	Port number, for example, `80`. If absent, the value is `nil`.	Number
`path`	Path to a resource, for example, `index.html`. If absent, the value is `nil`.	String
`query`	Decoded request parameters. If absent, the value is `nil`.	String
Overridden metamethods: • `__tostring`—function that returns the `RawUrl` content as a string (in the UTF-8 encoding); • `__concat`—function that concatenates the `RawUrl` string value and another string.		

UrlCategoryFilter table

The table describes a filter for URL categories. All table fields are optional.

Field	Description	Data type
`category`	List of categories that the URL must match (case-insensitive): • `infection_source`—infection source; • `not_recommended`—source that is not recommended for visiting; • `adult_content`—adult content; • `violence`—violence; • `weapons`—weapons; • `gambling`—gambling; • `drugs`—drugs; • `obscene_language`—obscene language;	String or table of strings

Field	Description	Data type
	• `chats`—chats; • `terrorism`—terrorism; • `free_email`—free email; • `social_networks`—social networks; • `owners_notice`—websites added due to a notice from a copyright owner; • `online_games`—online games; • `anonymizers`—anonymizers; • `cryptocurrency_mining_pools`—cryptocurrency mining pools; • `jobs`—job search websites; • `black_list`—black list.	
`category_not`	List of categories that the URL must not match (case-insensitive).	String or table of strings
Overridden metamethods: *None*		

If the filter field is not specified (i.e., the value is `nil`), any threat matches the filter. If several filter fields are specified, the condition is combined by a conjunction (logical AND). If the filter field is a table (list), the object being filtered must match at least one of the table (list) items.

Response table

The table describes the headers of an HTTP response.

Field	Description	Data type
`status`	HTTP response code or `nil`, if the ICAP request does not contain the HTTP response header.	Number
`reason`	Comment to the response code or `nil` if absent.	String
`content_type`	Contains information obtained from the `Content-Type` header of the HTTP response.	ContentType table
`field`	Array of HTTP response headers	Array of HeaderField tables
`search`	Function for searching a header by a regular expression. It accepts one mandatory argument—`patterns`—search patterns, i.e. one (string) or several (array of strings) regular expressions in the Perl syntax (PCRE). The search is performed in all available headers.	Function

Field	Description	Data type
	 When using quoted strings, the slash character must be escaped. Returns a Boolean value: • `true`—if the `field.name .. ": " .. field.value.decoded` string matches at least one of the specified regular expressions for at least one of the headers; • `false`—if no matches have been found.	
`value`	Function that accepts one mandatory argument—the header name (string). It returns the value of the first header with the specified name or `nil`, if there is no header with this name.	Function
Overridden metamethods: *None*		

Body table

A table describing the HTTP message body.

Field	Description	Data type
`has_threat`	Function that accepts one optional argument—`filter`, i.e. the ThreatFilter table (the absence of an argument is equivalent to using an empty table). It returns a Boolean value: • `true`—if the HTTP message body contains a threat that meets the specified `filter` condition; • `false`—if no matches have been found.	Function
`threats`	Function that accepts one optional argument—`filter`, i.e. the ThreatFilter table (the absence of the argument is equivalent to using an empty table). It returns an iterator function using which you can iterate over all threats detected in the HTTP message body. The threats are described using the Virus table.	Function
`content_type`	Contains information on the MIME type of the body obtained from the `Content-Type` header of the HTTP request or response (depending on the type of the message being analyzed).	ContentType table
`scan_error`	Body scanning error, if occurred, `nil` otherwise. Allowed values:	String

Field	Description	Data type
	• path_not_absolute—specified path is not absolute; • file_not_found—file has not been found; • file_not_regular—file is not regular; • file_not_block_device—not a block device; • name_too_long—name is too long; • no_access—access is denied; • read_error—reading error has occurred; • write_error—writing error has occurred; • file_too_large—file is too large; • file_busy—file is being used; • unpacking_error—unpacking error has occurred; • password_protected—archive is password-protected; • arch_crc_error—archive CRC error; • arch_invalid_header—invalid archive header; • arch_no_memory—not enough memory to unpack the archive; • arch_incomplete—archive is incomplete; • can_not_be_cured—file cannot be cured; • packer_level_limit—packed object nesting limit has been exceeded; • archive_level_limit—archive nesting limit has been exceeded; • mail_level_limit—mail file nesting limit has been exceeded; • container_level_limit—container nesting limit has been exceeded; • compression_limit—compression rate limit has been exceeded; • report_size_limit—report size limit has been exceeded; • scan_timeout—scanning timeout limit has been exceeded; • engine_crash—scanning engine failure; • engine_hangup—scanning engine hangup; • engine_error—scanning engine error; • no_license—no active license has been found; • multiscan_too_late—multithreaded scanning error has occurred;	

Field	Description	Data type
	• `curing_limit_reached`—limit on curing attempts has been exceeded; • `non_supported_disk`—disk type is not supported; • `unexpected_error`—unexpected error.	
Overridden metamethods: *None*		

Virus table

Table that describes a threat.

Field	Description	Data type
`name`	Threat name (according to the Doctor Web classification)	String
`type`	Threat type (according to the Doctor Web classification). Allowed values: • `known_virus`—known threat (that is, a threat that is covered by virus databases); • `virus_modification`—modification of a known threat; • `unknown_virus`—unknown threat, a suspicious object; • `adware`—adware; • `dialer`—dialer program; • `joke`—joke program; • `riskware`—potentially dangerous program; • `hacktool`—hacktool.	String
Overridden metamethods: *None*		

ThreatFilter table

The table describes the filter for threats.

Field	Description	Data type
`category`	List of categories that the threat must match (case-insensitive). See the list of categories in description of the `type` field of the Virus table.	String or table of strings
`category_not`	List of categories that the threat cannot match (case-insensitive).	String or table of strings
Overridden metamethods: *None*		

If the filter field is not specified (that is, the value is `nil`), any threat matches the filter. If several filter fields are specified, the condition is combined by a conjunction (logical AND). If the filter field is a table (list), the object being filtered must match at least one of the table (list) items.

Usage Examples

1. Log the list of all threats found in the HTTP message:

```
local dw = require "drweb"

function message_hook(ctx)
  for virus in ctx.body.threats() do
   dw.notice("threat found: " .. virus.name)
  end
  return "pass"
end
```

2. Writing to the log the threat names that match the category filter, and the names of the message parts where the threats have been detected:

```
local dw = require "drweb"

function message_hook(ctx)
  for v in ctx.body.threats({category = "known_virus"}) do
   dw.notice("found known virus: " .. v.name)
  end
  return "pass"
end
```

Available service modules

The following custom modules listed in the table can be imported into Lua program space for interaction with Dr.Web Gateway Security Suite.

Name of the module	Function
drweb	Module that provides functions for writing messages from the Lua program to the log of the Dr.Web Gateway Security Suite component which has started the Lua program, as well as the means to run Lua procedures asynchronously
drweb.lookup	Module that provides tools to query data from external sources using the Dr.Web LookupD module
drweb.regex	Module that provides an interface to match strings with regular expressions
drweb.config	Module that provides a table with the Dr.Web ICAPD configuration parameter values

Contents of the drweb module

1. Functions

The module provides a set of functions.

- Storing messages from the Lua program in the Dr.Web Gateway Security Suite component log:
  - `log(<level>, <message>)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the <*level*> level (the required level is defined using one of the following values (a string): *debug*, *info*, *notice*, *warning*, *error*);
  - `debug(<message>)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *debug* level;
  - `info(<message>)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *info* level;
  - `notice(<message>)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *notice* level;
  - `warning(<message>)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *warning* level;
  - `error(<message>)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *error* level.
- Managing synchronization of Lua procedures:
  - `sleep(<sec.>)` pauses execution of a Lua procedure instance for a specified number of seconds;
  - `async(<Lua function>[, <argument list>])` starts the specified function asynchronously and passes the specified argument list to it. The `async` function call completes immediately, and the return value (the `Future` table) allows you to obtain the result of the <*Lua function*>.
- Storing the information about an IP address in the form of the IpAddress table:
  - `ip(<address>)` indicates an IP address (IPv4 or IPv6) in the form of the IpAddress table.
- Retrieving external data from a text file:
  - `load_set(<file path>)` generates a table with its values assigned to `true` from the contents of a specified text file; strings read from the file are used as keys. Empty strings and strings consisting only of white spaces are ignored.
  - `load_array(<path to file>)` generates a string array from the contents of a specified text file. Empty strings and strings consisting only of white spaces are ignored.

2. Tables

- The `Future` table describes a pending result of performing a function using the `async` function.

Field	Description	Data type
`wait`	Function that returns a result of the function started using the `async` function. If the function has not completed its execution yet, it waits for the completion and returns its result. If the function is completed before `wait` is called, the result is returned immediately. If the started function fails, the `wait` call generates the same error.	Function
Overridden metamethods: *None*		

- The `IpAddress` table describes an IP address.

Field	Description	Data type
`belongs`	Function checks an IP address stored in the `IpAddress` table for belonging to specified subnets (IP address ranges). Accepts a single argument—an array of strings such as <*IP address*> or <*IP address*>/<*mask*>, where <*IP address*> is a host address or a network address (for example, `127.0.0.1`), and <*mask*> is a subnet mask (can be specified as an IP address, for example, `255.0.0.0`, or as an integer, for example, `8`). Returns a Boolean value: • `true`, if the address matches at least one of the specified IP addresses or belongs to at least one of the specified subnets (a range of IP addresses); • `false`, if the address does not match any one of the specified IP addresses or does not belong to any of the specified subnets	Function
Overridden metamethods: • `__tostring` is a function that transforms `IpAddress` into a string, for example, `127.0.0.1` (IPv4) or `::1` (IPv6); • `__concat` is a function that joins `IpAddress` to a string; • `__eq` is a function that checks whether two `IpAddress` are equal; • `__band` is a function that allows applying a mask, for example: `dw.ip('192.168.1.2') & dw.ip('255.255.254.0')`		

3. Examples

- Log messages generated by a procedure started asynchronously:

```
local dw = require "drweb"

-- This function returns a string received as an argument
-- after a delay of two seconds
function out_msg(message)
 dw.sleep(2)
 return message
```

```
end

-- "Main" function
function intercept(ctx)
 -- Output a string at the Notice level to the log of Dr.Web Gateway Security Suite
 dw.notice("Intercept function started.")

 -- Run two instances of the out_msg function asynchronously
 local f1 = dw.async(out_msg, "Hello,")
 local f2 = dw.async(out_msg, " world!")

 -- Wait for the completion of the instances of the function
 -- out_msg and output their results
 -- to the Dr.Web Gateway Security Suite log at the Debug level
 dw.log("debug", f1.wait() .. f2.wait())
end
```

- Create a scheduled procedure:

```
local dw = require "drweb"

-- Store the Future table as a future global variable to
-- prevent its removal by the garbage collector
future = dw.async(function()
   while true do
     -- Log the following message every day
     dw.sleep(60 * 60 * 24)
     dw.notice("A brand new day began")
   end
end)
```

- Transform a string to an IP address:

```
local dw = require "drweb"

local ipv4 = dw.ip("127.0.0.1")
local ipv6 = dw.ip("::1")
local mapped = dw.ip("::ffff:127.0.0.1")
```

Contents of the drweb.lookup module

1. Functions

The module provides the following functions:

- `lookup(`<*query*>`, `<*parameters*>`)` queries data from an external storage available via the Dr.Web LookupD module. The <*query*> argument must correspond to a section in Dr.Web LookupD settings (the <*type*>@<*tag*> string). The <*parameters*> argument is optional and describes substitutions to be applied to generate the query. The following automatically resolvable tokens can be used:
  - `$u`, `$U` (replaced with `user`)—user name sent by the client component;
  - `$d`, `$D` (replaced with `domain`)—domain name sent by the client component.

  Arguments are set as a table whose keys and values must be strings. The function returns an array of strings that are query results.

- `check(<checked string>, <query>, <parameters>)` returns `true` if *<checked string>* is found in an external repository accessible via the Dr.Web LookupD module. The *<query>* and *<parameters>* arguments are equivalent to the arguments of the `lookup` function (see above). The *<checked string>* argument must be a string or a table with a `__tostring` metamethod (i.e. that can be transformed into a string).

2. Examples

- Log a list of users retrieved from the `LookupD.LDAP.users` data source:

```
local dw = require "drweb"
local dwl = require "drweb.lookup"

-- "Main" function
function intercept(ctx)
 -- Store the string in the Dr.Web Gateway Security Suite log at the Notice level
 dw.notice("Intercept function started.")

 -- Store in the Dr.Web Gateway Security Suite log the results of quering
 -- the 'ldap@users' data source
 for _, s in ipairs(dwl.lookup("ldap@users", {user="username"})) do
   dw.notice("Result of request to 'ldap@users': " .. s)
 end

end
```

Contents of the drweb.regex module

1. Functions

The module provides the following functions:

- `search(<template>, <text>[, <flags>])` returns `true` if the *<text>* string contains a substring that matches the *<template>* regular expression. The optional *<flags>* parameter (integer) is a set of flags affecting the function behavior and connected with the logical OR.
- `match(<template>, <text>[, <flags>])`—the same as `search` except that the *<template>* regular expression must match the entire *<text>* string, not only its substring.

2. Available flags

- `ignore_case` ignores text case.

3. Examples

```
local rx = require "drweb.regex"

rx.search("te.?t", "some TexT") -- false
rx.search("te.?t", "some TexT", rx.ignore_case) -- true

rx.match("some.+", "some TexT") -- true
```

Contents of the drweb.config module

1. Functions

The module does not provide any functions.

2. Available tables

The module provides a table with the following fields:

Field	Description	Data type
`whitelist`	Value of the `Whitelist` configuration parameter.	String array
`blacklist`	Value of the `Blacklist` configuration parameter.	String array
`adlist`	Value of the `Adlist` configuration parameter.	String array
`block_url_categories`	List of blocked URL categories (based on the `Block*` parameter values set to `Yes`).	String array
`block_threats`	List of blocked threat categories (based on the `Block*` parameter values set to `Yes`).	String array
`block_unchecked`	Value of the `BlockUnchecked` configuration parameter.	Boolean
Overridden metamethods: *None*		

3. Example

```
local cfg = require "drweb.config"

function message_hook(ctx)

  -- Block messages containing threats
  -- from the list of threats to be blocked
  if ctx.body.has_threat{category = cfg.block_threats} then
    return "block"
  end

  -- Allow access to all other resources
  return "pass"

end
```

9.5. SpIDer Gate



This component is included only in the distributions for GNU/Linux OSes.

The component for monitoring network traffic and URLs SpIDer Gate is designed to scan data downloaded from the network to a local computer or passed to the network from a local host for threats and to prevent connections to network hosts covered by the unwanted categories of web resources and by the black lists defined by the administrator.

Types of protocols for scanning can be indicated in the component settings.

To check whether a URL belongs to any of the categories (to scan connections that utilize the HTTP/HTTPS protocol), the component not only uses the database of web resource categories, which is updated regularly from the Doctor Web update servers, but also refers to the Dr.Web Cloud service. Doctor Web keeps track of the following web resources categories:

- *InfectionSource*—websites containing malware ("infection sources").
- *NotRecommended*—fraudulent websites (that use "social engineering") visiting which is not recommended.
- *AdultContent*—websites that contain pornographic or erotic materials, dating websites and so on.
- *Violence*—websites that encourage violence or contain materials about various fatal accidents and so on.
- *Weapons*—websites dedicated to weapons and explosives or providing information on their manufacturing and so on.
- *Gambling*—websites that provide access to online games of chance, casinos, auctions, including websites for placing bets and so on.
- *Drugs*—websites that promote use, production or distribution of drugs and so on.
- *ObsceneLanguage*—websites that contain obscene language (in section titles, articles and so on).
- *Chats*—websites that offer real-time exchange of text messages.
- *Terrorism*—websites that contain aggressive and propaganda materials or description of terrorist attacks, and so on.
- *FreeEmail*—websites that offer the possibility of free registration of an email box.
- *SocialNetworks*—social networking services: general, professional, corporate, interest-based; thematic dating websites.
- *DueToCopyrightNotice*—websites links to which are provided by the copyright holders of some copyrighted work (movies, music, and so on).
- *OnlineGames*—websites that provide access to games using a permanent internet connection.

- *Anonymizers*—websites allowing the user to hide personal information and providing access to blocked websites.
- *CryptocurrencyMiningPool*—websites that provide access to services for cryptocurrency mining.
- *Jobs*—job search websites.

Your system administrator can specify unwanted categories of hosts. Additionally, the user can configure their own black lists of hosts access to which will be blocked, and white lists of hosts access to which will be allowed even if they belong to the unwanted categories. If there is no information about URLs in the local black lists and the database of web resources categories, the component can send requests to the Dr.Web Cloud service to check whether these URLs are malicious. This information is received from other Dr.Web products on a real-time basis.



One and the same website can belong simultaneously to several categories. Access to such website is blocked if it belongs to any of the unwanted categories.

Even if a website is included in the white list, the data downloaded from the website or sent to it is scanned for threats.

If file scanning via HTTP is highly intensive, issues with scanning may arise when available file descriptors are depleted by the Dr.Web Network Checker component. In this case, it is necessary to increase the limit by the number of file descriptors available to Dr.Web Gateway Security Suite.

As part of Dr.Web Gateway Security Suite, the SpIDer Gate component can be used by a company to create a "barrier" between a company server, such as a web server with public access, and the internet, because access of local network users to the internet is managed by default by the Dr.Web ICAPD component. This component operates together with a proxy server providing internet access from the local network.

9.5.1. Operating Principles

The SpIDer Gate component monitors network connections initiated by user applications. The component checks whether a server to which a client application is trying to connect belongs to any of the web resource categories specified in the settings as unwanted. Moreover, the component can use the Dr.Web Cloud service to scan URLs. If the URL belongs to any of the unwanted categories (or is flagged by the Dr.Web Cloud service) or to a black list defined by your system administrator, the connection is terminated and an HTML page with a message of that access is denied is displayed (in case of an HTTP/HTTPS connection). The page is generated by SpIDer Gate on the basis of a template supplied with the component. This page contains a notification of that access to the requested resource is impossible and describes a reason for blocking. A similar page is displayed and returned to the client if SpIDer Gate detects a threat that must be blocked in the data being transmitted. If the connection uses a protocol different from HTTP(S), the component only checks for permission to establish a connection with this server.

The Dr.Web Firewall for Linux service component redirects connections to remote servers established by client applications transparently to them and exercises dynamic control of the rules of NetFilter, a system component of Linux.

With regard to Dr.Web Gateway Security Suite, a client application is a protected server resource of a company (for example, a web server with public access), because by default the Dr.Web ICAPD component manages access of local network users to the internet. This component operates together with a proxy server providing internet access from the local network.

The same Dr.Web Updater component regularly and automatically updates databases of web resource categories from Doctor Web servers and virus databases for Dr.Web Scanning Engine. The Dr.Web Cloud service is maintained by the Dr.Web CloudD component (using the cloud service is configured in the general settings of Dr.Web Gateway Security Suite and can be disabled, if necessary). To scan data being transmitted, SpIDer Gate uses a network scanning agent, Dr.Web Network Checker, which initiates data scanning via Dr.Web Scanning Engine.

9.5.2. Command-Line Arguments

To start the SpIDer Gate component from the command line, run the command:

```
$ /opt/drweb.com/bin/drweb-gated [<parameters>]
```

SpIDer Gate accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-gated --help
```

This command outputs short help information about the SpIDer Gate component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management

daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-gated` command.

9.5.3. Configuration Parameters

The component uses configuration parameters specified in the `[GateD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-gated` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-gated`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down.

Parameter	Description
	Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`
`TemplatesDir` *{path to directory}*	Path to a directory that contains the templates for the HTML notifications sent upon blocking a web resource. Default value: • for GNU/Linux: `/var/opt/drweb.com/templates/gated` • for FreeBSD: `/var/drweb.com/templates/gated`
`CaPath` *{path}*	Path to the directory or file with a list of trusted root certificates. Default value: *Path to the list of trusted certificates*. The path depends on your GNU/Linux distribution. • For Astra Linux, Debian, Linux Mint, SUSE Linux and Ubuntu this is usually the path `/etc/ssl/certs/`. • For CentOS and Fedora—the path `/etc/pki/tls/certs/ca-bundle.crt`. • For other distributions, the path can be determined with the `openssl version -d` command. • If the command is unavailable or your OS distribution cannot be identified, the `/etc/ssl/certs/` value is used.



Changes made to the settings of the connection scanning do not influence the scanning of connections that have already been established by the applications before making changes.

Other parameters of traffic monitoring, as well as its rules, are defined in the settings of the Dr.Web Firewall for Linux service component.

9.6. Dr.Web Firewall for Linux



This component is included only in the distributions designed for the OSes of the GNU/Linux family.

To enable the correct operation of the component, the OS kernel must be built with the following options:

- *CONFIG_NETLINK_DIAG*, *CONFIG_INET_TCP_DIAG*;
- *CONFIG_NF_CONNTRACK_IPV4*, *CONFIG_NF_CONNTRACK_IPV6*, *CONFIG_NF_CONNTRACK_EVENTS*;
- *CONFIG_NETFILTER_NETLINK_QUEUE*, *CONFIG_NETFILTER_NETLINK_QUEUE_CT*, *CONFIG_NETFILTER_XT_MARK*.

The set of required options from the specified list can depend on the GNU/Linux OS in use.

The Dr.Web Firewall for Linux is an auxiliary component functioning as a connection manager for SpIDer Gate. Dr.Web Firewall for Linux passes established connections through SpIDer Gate for scanning the transmitted traffic.

9.6.1. Operating Principles

In this section

- General information
- Mechanism for intercepting connections
  - Actions in iptables and nftables rules
  - Marks of packets and connections
  - Routes and routing policies (ip rule, ip route)
  - NetFilter (iptables) rules
  - NetFilter (nftables) rules
- Order of intercepting connections

General information

The Dr.Web Firewall for Linux component ensures the correct operation of SpIDer Gate by analyzing routing rules adjusted for NetFilter (a system component of Linux) and modifies it so that the connections being established are redirected to SpIDer Gate, which functions as an intermediate (proxy) between a client application and a remote server.

Dr.Web Firewall for Linux can separately manage the rules for redirecting outgoing, incoming and transit connections. To configure the rules for passing or redirecting connections in detail, the component can use the rules incorporated in settings and a Lua script.

Dr.Web Firewall for Linux supports configuring Netfilter using `nftables` and `iptables` interfaces (the latter is used for compatibility).



To get more detailed information about Netfilter subsystem interfaces, run the `$ man iptables` and `$ man nftables` commands.

Mechanism for intercepting connections

To intercept connections, Dr.Web Firewall for Linux uses routing tables specified in the database of routing policies and the *nf_conntrack* interface of the NetFilter system component. The intercepted connections and packets being transmitted are marked with bit marks to ensure proper routing thereby allowing Dr.Web Firewall for Linux to properly redirect connections and process packets being transmitted on various stages of passing NetFilter chains.



To get more detailed information about managing routing policies, run the `$ man ip`, `$ man ip route` and `$ man ip rule` commands.

Actions in iptables and nftables rules

Dr.Web Firewall for Linux uses the following actions in the `iptables` and `nftables` rules:

- *MARK*—assign a specified numeric mark to a packet.
- *TPROXY*—set an initial destination address of the connection and redirect packets from the *PREROUTING* NetFilter chain to a specified network socket (<*IP address*>:<*port*>) without changing contents of a packet.
- *NFQUEUE*—send a packet from the kernel network stack to a process that operates outside the kernel space for scanning. Dr.Web Firewall for Linux connects to the *NFQUEUE* queue with a specified number via a custom *Netlink* socket and receives packets for which a verdict on further processing must be reached (Dr.Web Firewall for Linux must inform NetFilter of one the following verdicts: *DROP*, *ACCEPT* or *REPEAT*).

The *CONNMARK* action (assign a specified numeric mark to a connection) is supported by `iptables`, but not by `nftables`. In `nftables` rules, use `ct mark` or `ct mark set` expressions.

Marks of packets and connections

To mark packets, Dr.Web Firewall for Linux uses the following three bits (out of available 32 bits) in packet and connection marks:

- `LDM` bit (*Local Delivery Mark*)—indicator of a local connection. Packets with this bit in the mark are sent to the local host using set routing rules.
- `CPM` bit (*Client Packets Mark*)—indicator of a connection between a client (a connection initiator) and a proxy, that is, Dr.Web Firewall for Linux.
- `SPM` bit (*Server Packets Mark*)—indicator of a connection between a proxy, that is, Dr.Web Firewall for Linux, and a server (a connection recipient).

`LDM`, `CPM` and `SPM` bits can be any *different* bits that are not used for marking packets by other applications that perform routing of connections. By default, Dr.Web Firewall for Linux chooses appropriate (not used by other applications) bits automatically.

Routes and routing policies (ip rule, ip route)

To ensure correct operation of Dr.Web Firewall for Linux (in any connection scanning mode), the `ip rule` routing policy that uses routing table #100 must be configured on your system:

```
from all fwmark <LDM>/<LDM> lookup 100
```

The following route must be added to this table:

```
local default dev lo scope host
```

This routing policy guarantees that packets with the `LDM` bit in their marks are always sent to the local host.



Hereinafter the expression <*XXX*> for the `XXX` bit is a *hexadecimal* value that equals 2^N, where *N* is a sequential number of the `XXX` bit in the packet mark. For example, if the lowest (zero) bit was selected as an `LDM` bit, then <*LDM*> = 2^0 = *0x1*.

NetFilter (iptables) rules

When using `iptables`, to ensure correct operation of Dr.Web Firewall for Linux (in any connection scanning mode), the following six rules (displayed in the `# iptables-save` output command format) must be present in the `nat` and `mangle` tables of the corresponding chains of the NetFilter subsystem:

```
*nat
-A POSTROUTING -o lo -m comment --comment drweb-firewall -m mark --mark <LDM>/<LDM> -j
ACCEPT
*mangle
-A PREROUTING -m comment --comment drweb-firewall -m mark --mark 0x0/<CPM+SPM> -m
connmark --mark <SPM>/<CPM+SPM> -j MARK --set-xmark <LDM>/<LDM>
-A PREROUTING -p tcp -m comment --comment drweb-firewall -m mark ! --mark
<CPM+SPM>/<CPM+SPM> -m connmark --mark <CPM>/<CPM+SPM> -j TPROXY --on-port <port> --
on-ip <IP address> --tproxy-mark <LDM>/<LDM>
-A OUTPUT -m comment --comment drweb-firewall -m mark --mark <CPM>/<CPM+SPM> -j
CONNMARK --set-xmark <CPM>/0xffffffff
-A OUTPUT -m comment --comment drweb-firewall -m mark --mark <SPM>/<CPM+SPM> -j
CONNMARK --set-xmark <SPM>/0xffffffff
-A OUTPUT -m comment --comment drweb-firewall -m mark --mark 0x0/<CPM+SPM> -m connmark
! --mark 0x0/<CPM+SPM> -j MARK --set-xmark <LDM>/<LDM>
```



These rules are numbered 1–6 in the description below (in the order they are listed here). The expression <*X*+*Y*> means a number equal to bitwise "OR" (a sum) of the corresponding numbers *X* and *Y*.

Parameters <*IP address*> and <*port*> in rule 3 specify a network socket on which Dr.Web Firewall for Linux manages intercepted connections.

Furthermore, the following additional rules (per rule for each of the modes) must be present in `mangle` tables of the corresponding chains (*OUTPUT*, *INPUT*, *FORWARD*) when enabling the interception mode for outgoing, incoming and transit connections in the Dr.Web Firewall for Linux settings:

- To intercept outgoing connections (*OUTPUT*):

```
-A OUTPUT -p tcp -m comment --comment drweb-firewall -m tcp --tcp-flags SYN,ACK SYN -m
mark --mark 0x0/<CPM+SPM> -j NFQUEUE --queue-num <ONum> --queue-bypass
```

- To intercept incoming connections (*INPUT*):

```
-A INPUT -p tcp -m comment --comment drweb-firewall -m tcp --tcp-flags SYN,ACK SYN -m
mark --mark 0x0/<CPM+SPM> -j NFQUEUE --queue-num <INum> --queue-bypass
```

- To intercept transit connections (*FORWARD*):

```
-A FORWARD -p tcp -m comment --comment drweb-firewall -m tcp --tcp-flags SYN,ACK SYN -
m mark --mark 0x0/<CPM+SPM> -j NFQUEUE --queue-num <FNum> --queue-bypass
```



These rules are numbered 7, 8 and 9 in the description below (in the order they are listed here).

Here, <*ONum*>, <*INum*> and <*FNum*> are numbers of queues in *NFQUEUE*, in which Dr.Web Firewall for Linux is waiting for packets that indicate establishing connections with the corresponding directions (these are packets with a set `SYN` flag and an unset `ACK` flag).

NetFilter (nftables) rules

When using `nftables`, to ensure correct operation of Dr.Web Firewall for Linux (in any connection scanning mode), the following six rules (displayed in the `# nft list ruleset` output command format) must be present in the `drweb-firewall` table of the NetFilter subsystem (the output is OS-dependent):

```
table ip drweb-firewall {
	chain prerouting {
		type filter hook prerouting priority mangle; policy accept;
		meta mark & <CPM+SPM> == 0x0 ct mark & <CPM+SPM> == <SPM> meta mark set
meta mark | <LDM> counter packets 0 bytes 0 comment "drweb-firewall"
		meta l4proto tcp meta mark & <CPM+SPM> != <CPM+SPM> ct mark & <CPM+SPM>
== <CPM> tproxy to <IP address>:<port> meta mark set meta mark | <LDM> counter packets 0
bytes 0 accept comment "drweb-firewall"
	}

	chain output {
		type route hook output priority mangle; policy accept;
		meta l4proto tcp @th,104,8 & 0x12 == <CPM> meta mark & <CPM+SPM> == 0x0
counter packets 0 bytes 0 queue flags bypass to 0 comment "drweb-firewall"
		meta mark & <CPM+SPM> == <CPM> ct mark set ct mark & <CPM> | <CPM>
counter packets 0 bytes 0 comment "drweb-firewall"
		meta mark & <CPM+SPM> == <SPM> ct mark set ct mark & <SPM> | <SPM>
counter packets 0 bytes 0 comment "drweb-firewall"
		meta mark & <CPM+SPM> == 0x0 ct mark & <CPM+SPM> != 0x0 meta mark set
meta mark | <LDM> counter packets 0 bytes 0 comment "drweb-firewall"
	}

	chain input {
		type filter hook input priority mangle; policy accept;
	}

	chain forward {
		type filter hook forward priority mangle; policy accept;
	}
}
```

The expression <*X*+*Y*> means a number equal to bitwise "OR" (a sum) of the corresponding numbers *X* and *Y*.

Parameters <*IP address*> and <*port*> specify a network socket on which Dr.Web Firewall for Linux manages intercepted connections.

The following rules (per rule for each of the modes) must be present in the `drweb-firewall` table of the corresponding chains (*OUTPUT*, *INPUT*, *FORWARD*) when enabling the interception mode for outgoing, incoming and transit connections in the Dr.Web Firewall for Linux settings:

- To intercept outgoing connections (*OUTPUT*):

```
meta l4proto tcp @th,104,8 & 0x12 == <CPM> meta mark & <CPM+SPM> == 0x0 counter
packets 0 bytes 0 queue flags bypass to 0 comment "drweb-firewall"
```

- To intercept incoming connections (*INPUT*):

```
meta l4proto tcp @th,104,8 & 0x12 == <CPM> meta mark & <CPM+SPM> == 0x0 counter
packets 0 bytes 0 queue flags bypass to 1 comment "drweb-firewall"
```

- To intercept transit connections (*FORWARD*):

```
meta l4proto tcp @th,104,8 & 0x12 == <CPM> meta mark & <CPM+SPM> == 0x0 counter
packets 0 bytes 0 queue flags bypass to 2 comment "drweb-firewall"
```

Order of intercepting connections

In accordance with any of `iptables` rules 7, 8 and 9, packets indicating a new network connection of the corresponding direction, if not marked by both `CPM` and `SPM` bits, are put in the corresponding *NFQUEUE* queues by NetFilter, where they will be read by Dr.Web Firewall for Linux via the *nf_conntrack* interface. Rules 4 and 5 mark the connection itself as intercepted, that is, a bit indicating the connection direction is set in the connection mark. This bit number matches the bit number in the packet mark. As the result, in accordance with rules 2, 3 and 6, Dr.Web Firewall for Linux will receive packets sent via this connection. Rule 1 is added on top of the *POSTROUTING* chain in the `nat` table, so that if NAT is configured, not to translate addresses for marked packets, because this will interfere with Dr.Web Firewall for Linux logic of connection interception and processing.

When a packet appears in one of the *NFQUEUE* queues, Dr.Web Firewall for Linux performs basic verification of the packet in the event invalid rules are set in NetFilter. Subsequently, Dr.Web Firewall for Linux attempts to connect on behalf of itself to the server from the socket marked with `PSC`. At that, rule 5 is triggered. Rule 6 for local delivery is not triggered, because the packet is marked with `SPM` and this rule is only applicable to packets marked with *<CPM+SPM>*.

- *If the connection to the server fails*, Dr.Web Firewall for Linux generates a client packet with an `RST` bit having replaced the pair *<IP address>*:*<port>* with the address of the network socket of the requested server. At that, the *DROP* verdict is sent to *NFQUEUE*. The socket used for sending the packet with the `RST` bit is marked as *<CPM+SPM>*, so none of the above rules is triggered, and this packet will be delivered to the client in accordance with standard routing rules.
- *If an attempt to connect to a remote server is successful*, Dr.Web Firewall for Linux copies the intercepted SYN packet and resends it from the socket marked as *<LDM+CPM>* to redirect the sent packet to the local network socket. Owing to the set `LDM` bit and in accordance with the specified routing rules, when selecting an output interface, the packet will be sent to the *loopback* interface and then to the NetFilter *PREROUTING* chain, where rule 3 will be

triggered. Thus, the packet will be redirected to the network socket of Dr.Web Firewall for Linux unchanged. This method allows Dr.Web Firewall for Linux to preserve all four elements of the connection address (an IP address and a port of a packet sender, an IP address and a port of a packet recipient).

The `IP_TRANSPARENT` option and the <*LDM+CPM*> mark are set for the network socket on which Dr.Web Firewall for Linux receives intercepted connections in accordance with rule 3; therefore, packets sent by Dr.Web Firewall for Linux from this socket are not included in the *NFQUEUE* queues. When a client connects, search is performed for a paired socket using the preserved four-element address (the IP address and the port of the packet sender, the IP address and the port of the packet recipient). When the connection with the client and the server is established, it is scanned using a Lua procedure, as well as scanning rules specified in the Dr.Web Firewall for Linux settings. If the scans are successful and the connection should not be closed, the associated socket pair that connects the client and server sides of the established connection is passed to the SpIDer Gate component for analyzing the data transmitted over the connection. The further interaction of the client and the server is mediated by SpIDer Gate. In addition to the socket pair associated with the client and server sides of the connection, SpIDer Gate receives the parameters and rules for scanning the established connection from Dr.Web Firewall for Linux.

Simplified diagram of Dr.Web Firewall for Linux operation is provided below.

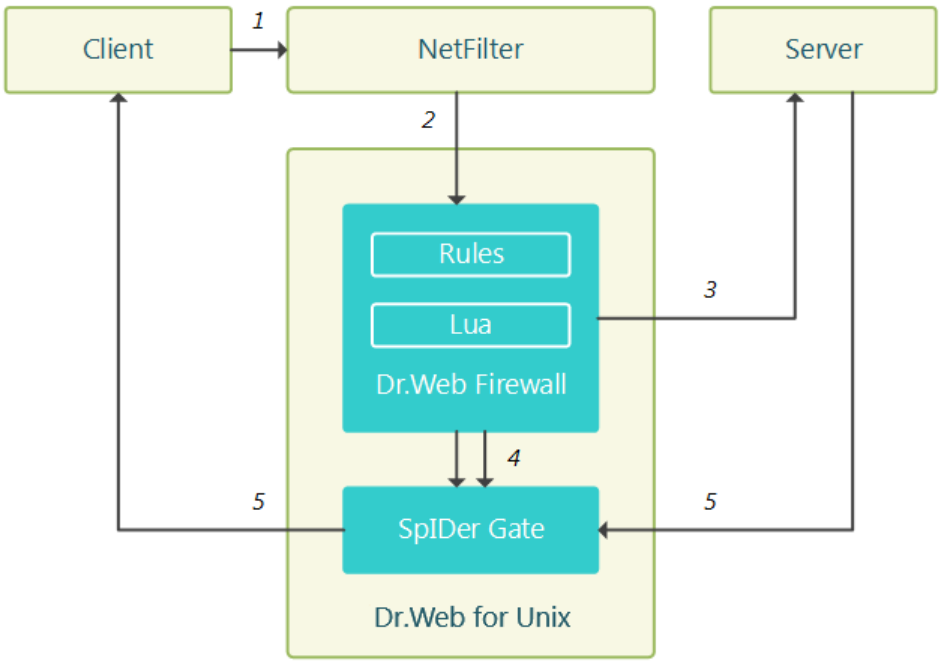


Figure 12. Diagram of Dr.Web Firewall for Linux operation

The following connection processing steps are numbered:

1. Client attempts to connect to the server.
2. NetFilter redirects the connection being established to Dr.Web Firewall for Linux in accordance with the routing rules.
3. Dr.Web Firewall for Linux attempts to connect to the server on behalf of the client; the connection is scanned.

4. Socket pair associated with the client and server sides of the connection is passed to SpIDer Gate for processing the connection along with the parameters and rules for scanning it.
5. Server and the client exchange data via SpIDer Gate that serves as a mediator.



For correct operation of Dr.Web Firewall for Linux, these rules must be present in routing tables with correct numbers of bit marks, *NFQUEUE* queues and network socket addresses for connection interception. By default, the component performs the necessary configuration of rules automatically. If automatic configuration of connections is disabled in the component settings, it is necessary to provide the required rules manually when starting the component.

9.6.2. Command-Line Arguments

To start the Dr.Web Firewall for Linux component from the command line, run the command:

```
$ /opt/drweb.com/bin/drweb-firewall [<parameters>]
```

Dr.Web Firewall for Linux accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-firewall --help
```

This command outputs short help information about the Dr.Web Firewall for Linux component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-firewall` command.

9.6.3. Configuration Parameters

In this section

- Component parameters
- Traffic monitoring and access blocking rules
  - Viewing and editing the rules
  - Viewing the rules using the drweb-ctl cfshow command
  - Editing the rules using the drweb-ctl cfset command
  - Default set of rules
  - Examples of the traffic monitoring and access blocking rules

The component uses configuration parameters specified in the `[LinuxFirewall]` section of the unified configuration file of Dr.Web Gateway Security Suite.

Component parameters

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-firewall` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-firewall`

Parameter	Description
`AutoconfigureIptables` *{Yes | No}*	Enable or disable the mode of configuring the rules for the NetFilter system component via the `iptables` interface. Allowed values: • `Yes`—automatically configure NetFilter rules upon starting the component and remove them upon finishing its operation (*recommended*). • `No`—do not configure the rules automatically. The required rules must be added manually by the administrator before starting the component and removed after it finishes its operation. ⚠ If the automatic configuration of the `iptables` rules is not allowed, the required `iptables` rules must be ensured before the component operation starts. Default value: `Yes`
`AutoconfigureRouting` *{Yes | No}*	The configuration mode for `ip route` and `ip rule` routing rules and policies. Allowed values: • `Yes`—automatically configure `ip route` and `ip rule` rules and routing policies upon starting the component and remove them upon finishing its operation (*recommended*). • `No`—do not configure the rules automatically. The required rules must be added manually by the administrator before starting the component and removed after it finishes its operation. ⚠ If the automatic configuration of the routing rules and policies is not allowed, the required rules for `ip route` and `ip rule` must be available before the component operation starts. Default value: `Yes`
`LocalDeliveryMark` *{integer | Auto}*	The *<LDM>* mark for packets that are redirected to the Dr.Web Firewall for Linux network socket (specified in the `TproxyListenAddress` parameter, see below) to intercept the connection.

Parameter	Description
	Allowed values: • *<integer>*—*<LDM>* mark for packets. Equals 2^N, where *N* is an `LDM` bit number in the packet, $0 \leq N \leq 31$. • `Auto`—allow Dr.Web Firewall for Linux to select the appropriate bit in the packet mark automatically (*recommended*).  When assigning *<LDM>* number manually, make sure that the corresponding bit in the packet mark is not used by any other applications that route connections and packets (including via NetFilter). If an invalid value is specified, the component will fail to start. The specified *<LDM>* number must be used in the routing rules to be added manually, if `AutoconfigureIptables = No` and/or `AutoconfigureRouting = No`. Default value: `Auto`
`ClientPacketsMark` *{integer | Auto}*	The *<CPM>* mark for packets transferred between the client that initiates the connection and Dr.Web Firewall for Linux. Allowed values: • *<integer>*—*<CPM>* mark for packets. Equals 2^N, where *N* is a `CPM` bit number in the packet, $0 \leq N \leq 31$. • `Auto`—allow Dr.Web Firewall for Linux to select the appropriate bit in the packet mark automatically (*recommended*).

Parameter	Description
	 When assigning the <*CPM*> number manually, make sure that the corresponding bit in the packet mark is not used by any other applications that route connections and packets (including via NetFilter). If an invalid value is specified, the component will fail to start. The specified <*CPM*> number must be used in the routing rules to be added manually, if `AutoconfigureIptables = No`. Default value: `Auto`
`ServerPacketsMark` *{integer | Auto}*	<*SPM*> mark for packets transferred between Dr.Web Firewall for Linux and the server that receives the connection. Allowed values: • <*integer*>—<*SPM*> mark for packets. Equals 2^N, where N is an `SPM` bit number in the packet, $0 \leq N \leq 31$. • `Auto`—allow Dr.Web Firewall for Linux to select the appropriate bit in the packet mark automatically (*recommended*).  When assigning the <*SPM*> number manually, make sure that the corresponding bit in the packet mark is not used by any other applications that route connections and packets (including via NetFilter). If an invalid value is specified, the component will fail to start. The specified <*SPM*> number must be used in the routing rules to be added manually, if `AutoconfigureIptables = No` and/or `AutoconfigureRouting = No`. Default value: `Auto`
`TproxyListenAddress` *{network socket}*	Network socket (<*IP address*>:<*port*>) on which Dr.Web Firewall for Linux receives intercepted connections. If

Parameter	Description
	you specify a zero port number, the socket is selected automatically by the system. ⚠ It is necessary to make sure that the corresponding socket is not used by any other applications. If an invalid value is specified, the component will fail to start. The specified IP address and port must be used in the routing rules to be added manually, if `AutoconfigureIptables = No`. Default value: `127.0.0.1:0`
`OutputDivertEnable` *{Yes | No}*	Enable or disable the interception mode for outgoing connections (i.e. connections initiated by applications on the local host). Allowed values: • `Yes`—intercept and process outgoing connections; • `No`—do not intercept or process outgoing connections. ⚠ This setting adds or removes routing rule #5 to be added or removed manually if `AutoconfigureIptables = No`. Default value: `No`
`OutputDivertNfqueueNumber` *{integer | Auto}*	Number of the *NFQUEUE* queue from which Dr.Web Firewall for Linux will retrieve SYN packets that initiate outgoing connections. Allowed values: • *<integer>—<ONum>* queue number to monitor the SYN packets of intercepted outgoing connections in *NFQUEUE*; • `Auto`—allow Dr.Web Firewall for Linux to select an appropriate queue number automatically (*recommended*).

Parameter	Description
	 When assigning the <*ONum*> number manually, make sure that the corresponding queue is not used by any other applications that control connections and packets (including via the NetFilter rules). If an invalid value is specified, the component will fail to start. The specified <*ONum*> number must be used in routing rule #5 to be added manually if `AutoconfigureIptables = No`. Default value: `Auto`
`OutputDivertConnectTransparently` *{Yes | No}*	Enable or disable the emulation mode for connecting to the recipient (server) using the IP address of the sender of an intercepted packet (client) for outgoing connections. Allowed values: • `Yes`—connect to the server using the address of the client that requested the connection instead of your own when intercepting the connection; • `No`—connect to the server from the Dr.Web Firewall for Linux address. As the client and Dr.Web Firewall for Linux addresses are usually the same in the outgoing connection interception mode, the default value is `No`. Default value: `No`
`OutputDivertMark` *{integer | Auto | None}*	Mark for packets in interception mode for outgoing connections if the PARSEC mandatory access subsystem is used. Allowed values: • <*integer*>—mark for packets. Equals 2^N, where N is a mark bit number in the packet, $0 \leq N \leq 31$. • `Auto`—allow Dr.Web Firewall for Linux to select the appropriate bit in the packet mark automatically. • `None`—do not set the mark. Default value: `None`
`InputDivertEnable`	Enable or disable interception of incoming connections (i.e. connections initiated by applications on the remote

Parameter	Description
{Yes | No}	host to applications operating on the local host). Allowed values: • `Yes`— intercept and process incoming connections; • `No`—do not intercept or process outgoing connections. ⚠ This setting adds or removes routing rule #7 to be added or removed manually if `AutoconfigureIptables = No`. If an invalid value is specified, the component will fail to start. Default value: `No`
`InputDivertNfqueueNumber` *{integer | Auto}*	Number of the *NFQUEUE* queue from which Dr.Web Firewall for Linux will retrieve SYN packets that initiate incoming connections. Allowed values: • *<integer>*—*<INum>* queue number to monitor the SYN packets of intercepted incoming connections in *NFQUEUE*; • `Auto`—allow Dr.Web Firewall for Linux to select an appropriate queue number automatically (*recommended*). ⚠ When assigning the *<INum>* number manually, make sure that the corresponding queue is not used by any other applications that control connections and packets (including via the NetFilter rules). If an invalid value is specified, the component will fail to start. The specified *<INum>* number must be used in routing rule #7 to be added manually if `AutoconfigureIptables = No`. Default value: `Auto`
`InputDivertConnectTransparently` *{Yes | No}*	Enable or disable the emulation mode for connecting to the recipient (server) using the IP address of the sender

Parameter	Description
	of an intercepted packet (client) for incoming connections. Allowed values: • `Yes`—connect to the server using the address of the client that requested the connection instead of your own when intercepting the connection; • `No`—connect to the server from the Dr.Web Firewall for Linux address. In the incoming connection interception mode, all traffic goes through Dr.Web Firewall for Linux, and it is possible to connect safely to the server using the fake client address; therefore, the default value is `Yes`. Default value: `Yes`
`InputDivertMark` *{integer | Auto | None}*	Mark for packets in interception mode for incoming connections if the PARSEC mandatory access subsystem is used. Allowed values: • *<integer>*—mark for packets. Equals 2^N, where N is a mark bit number in the packet, $0 \leq N \leq 31$. • `Auto`—allow Dr.Web Firewall for Linux to select the appropriate bit in the packet mark automatically. • `None`—do not set the mark. Default value: `None`
`ForwardDivertEnable` *{Yes | No}*	Enable or disable interception of transit connections (i.e. connections initiated by applications on one remote host to applications on another remote host). Allowed values: • `Yes`—intercept and process transit connections; • `No`—do not intercept or process transit connections. ⚠ This setting adds or removes routing rule #8 to be added or removed manually if `AutoconfigureIptables = No`. Default value: `No`
`ForwardDivertNfqueueNumber` *{integer | Auto}*	Number of the *NFQUEUE* queue from which Dr.Web Firewall for Linux will retrieve SYN packets that initiate transit connections.

Parameter	Description
	Allowed values: • *<integer>*—*<FNum>* queue number to monitor the SYN packets of intercepted transit connections in *NFQUEUE*; • `Auto`—allow Dr.Web Firewall for Linux to select an appropriate queue number automatically (*recommended*).  When assigning the *<FNum>* number manually, make sure that the corresponding queue is not used by any other applications that control connections and packets (including via the NetFilter rules). If an invalid value is specified, the component will fail to start. The specified *<FNum>* number must be used in routing rule #8 to be added manually if `AutoconfigureIptables = No`. Default value: `Auto`
`ForwardDivertConnectTransparently` *{Yes | No}*	Enable or disable the emulation mode for connecting to the recipient (server) using the IP address of the sender of an intercepted packet (client) for transit connections. Allowed values: • `Yes`—connect to the server using the address of the client that requested the connection instead of your own when intercepting the connection; • `No`—connect to the server from the Dr.Web Firewall for Linux address. As there is no guarantee that in the transit connection interception mode all the traffic goes through the same host (router) on which Dr.Web Firewall for Linux is installed, to ensure the correct operation, the default value is `No`. If the network configuration guarantees that protected applications use the same router, the parameter can be set to `Yes`, and in this case, Dr.Web Firewall for Linux will always emulate the connection from the client address when connecting to servers. Default value: `No`
`ForwardDivertMark`	Mark for packets in interception mode for transit connections if the PARSEC mandatory access subsystem

Parameter	Description
{integer | Auto | None}	is used. Allowed values: • <*integer*>—mark for packets. Equals 2^N, where *N* is a mark bit number in the packet, $0 \leq N \leq 31$. • `Auto`—allow Dr.Web Firewall for Linux to select the appropriate bit in the packet mark automatically. • `None`—do not set the mark. Default value: `None`

`Whitelist`
 	
{domain list}
	White list of domains (domains allowed for connection, even if these domains belong to blocked categories of web resources. In addition, user access is allowed to all subdomains of the domains from this list).
	
	The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	
	Example: Add domains `example.com` and `example.net` to the list.
	
	1. Adding values to the configuration file.
	• Two values per line:
	`[LinuxFirewall]`
	`Whitelist = "example.com", "example.net"`
	• Two lines (one value per line):
	`[LinuxFirewall]`
	`Whitelist = example.com`
	`Whitelist = example.net`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset LinuxFirewall.Whitelist -a example.com`
	`# drweb-ctl cfset LinuxFirewall.Whitelist -a example.net`

Parameter	Description
	 Actual usage of the domain list indicated in this parameter depends on the *method* of its usage in the scanning rules defined for Dr.Web Firewall for Linux. The list of default rules (see below) guarantees that access to domains (and their subdomains) from this list will be provided even if it contains domains from the list of blocked web resource categories, but only in case of a request to a host via the HTTP protocol. In addition, the default set of rules guarantees that data downloaded from the white list of domains *will be scanned for threats* (because the data is returned in a response, and the `direction` variable has the `response` value). Default value: *(not specified)*

`Blacklist`
 	
{domain list}
	Black list of domains (i.e. the domains forbidden for connection, even if these domains do not belong to blocked categories of web resources. In addition, user access will be forbidden to all subdomains of the domains from this list).
	
	The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	
	Example: Add domains `example.com` and `example.net` to the list.
	
	1. Adding values to the configuration file.
	• Two values per line:
	`[LinuxFirewall]`
	`Blacklist = "example.com", "example.net"`
	• Two lines (one value per line):
	`[LinuxFirewall]`
	`Blacklist = example.com`
	`Blacklist = example.net`

Parameter	Description

	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset LinuxFirewall.Blacklist -a example.com`
	`# drweb-ctl cfset LinuxFirewall.Blacklist -a example.net`
	
	Actual usage of the domain list indicated in this parameter depends on the *method* of its usage in the scanning rules defined for Dr.Web Firewall for Linux.
	The list of default rules (see below) guarantees that access to the domains (and their subdomains) from this list via the HTTP protocol will always be forbidden. If a domain is added simultaneously to `Whitelist` and `Blacklist`, the default rules guarantee that user access to the domain via the HTTP protocol will be blocked.
	Blacklisted websites are not blocked if scanning outgoing traffic is disabled (ensure that the `OutputDivertEnable` parameter is set to `Yes`, `On` or `True`). If these websites use HTTPS, also set the `UnwrapSsl` parameter to `Yes`, `On` or `True`.
	Default value: *(not specified)*

`InspectHttp` *{On | Off}*	Scan the data transferred over the HTTP protocol. The data will be scanned on the basis of the specified rules (see below). Default value: `On`
`InspectPop3` *{On | Off}*	Scan the data transferred over the POP3 protocol. The data will be scanned on the basis of the specified rules (see below). Default value: `On`
`InspectImap`	Scan the data transferred over the IMAP protocol.

Parameter	Description
{On | Off}	The data will be scanned on the basis of the specified rules (see below). Default value: `On`
`InspectSmtp` *{On | Off}*	Scan the data transferred over the SMTP protocol. The data will be scanned on the basis of the specified rules (see below). Default value: `On`
`InspectFtp` *{On | Off}*	Scan the data transferred over the FTP protocol. The data will be scanned on the basis of the specified rules (see below). Default value: `On`

`ExcludedProc`
 	
{path to file}
	White list of processes (the network activity of which is not controlled).
	
	Multiple values can be specified as a list. List values must be comma-separated and put in quotation marks. The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	
	Example: Add the `wget` and `curl` processes to the list.
	
	1. Adding values to the configuration file.
	• Two values per line:
	`[LinuxFirewall]`
	`ExcludedProc = "/usr/bin/wget", "/usr/bin/curl"`
	• Two lines (one value per line):
	`[LinuxFirewall]`
	`ExcludedProc = /usr/bin/wget`
	`ExcludedProc = /usr/bin/curl`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset LinuxFirewall.ExcludedProc -a /usr/bin/wget`
	`# drweb-ctl cfset LinuxFirewall.ExcludedProc -a /usr/bin/curl`

Parameter	Description
	 Actual usage of the process list indicated in this parameter depends on the *method* of its usage in the scanning rules defined for Dr.Web Firewall for Linux. The list of default rules guarantees that traffic of all processes from the list is allowed *without any scanning*. Default value: *(not specified)*
`UnwrapSsl` *{boolean}*	Scan or do not scan encrypted traffic passing via SSL.  In the current implementation, the value if this variable does not influence scanning of secure traffic. To control scanning, it is necessary to create a rule comprising the action `SET Unwrap_SSL = true/false`. If you change the value of this parameter using the `cfset` command of the `drweb-ctl` utility or using the management web interface, the affected dependent rules will adapt automatically. Default value: `No`
`BlockUnchecked` *{boolean}*	Block incoming and outgoing data if it cannot be scanned.  The value of this parameter influences processing of the rules that are impossible to evaluate to true or false because of an error. If `No` is specified, the rule is skipped as the rule that has not been executed. If `Yes` is specified, the `BLOCK as BlackList` action is performed. Default value: `No`
`BlockInfectionSource` *{boolean}*	Block attempts to connect to websites containing malware (belonging to the *InfectionSource* category).

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `Yes`
`BlockNotRecommended` *{boolean}*	Block attempts to connect to non-recommended websites (belonging to the *NotRecommended* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `Yes`
`BlockAdultContent` *{boolean}*	Block attempts to connect to websites containing adult content (belonging to the *AdultContent* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockViolence` *{boolean}*	Block attempts to connect to websites containing graphic violence (belonging to the *Violence* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockWeapons` *{boolean}*	Block attempts to connect to websites dedicated to weapons (belonging to the *Weapons* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockGambling` *{boolean}*	Block attempts to connect to gambling websites (belonging to the *Gambling* category).

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockDrugs` *{boolean}*	Block attempts to connect to websites dedicated to drugs (belonging to the *Drugs* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockObsceneLanguage` *{boolean}*	Block attempts to connect to websites with obscene language (belonging to the *ObsceneLanguage* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockChats` *{boolean}*	Block attempts to connect to chat websites (belonging to the *Chats* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockTerrorism` *{boolean}*	Block attempts to connect to websites dedicated to terrorism (belonging to the *Terrorism* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockFreeEmail` *{boolean}*	Block attempts to connect to websites of free email services (belonging to the *FreeEmail* category).

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockSocialNetworks` *{boolean}*	Block attempts to connect to social networking websites (belonging to the *SocialNetworks* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockDueToCopyrightNotice` *{boolean}*	Block attempts to connect to websites that were added at the request of a copyright holder (belonging to the *DueToCopyrightNotice* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `Yes`
`BlockOnlineGames` *{boolean}*	Block attempts to connect to online gaming websites (belonging to the *OnlineGames* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockAnonymizers` *{boolean}*	Block attempts to connect to anonymizer websites (belonging to the *Anonymizers* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockCryptocurrencyMiningPools` *{boolean}*	Block attempts to connect to websites combining users to mine cryptocurrencies (belonging to the *CryptocurrencyMiningPool* category).

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockJobs` *{boolean}*	Block attempts to connect to job search websites (belonging to the *Jobs* category). To enable blocking, the settings must contain the following rule (see below): `url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match` Default value: `No`
`BlockKnownVirus` *{boolean}*	Block incoming and outgoing data if it contains a known threat. To enable blocking, the settings must contain the following rule (see below): `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockSuspicious` *{boolean}*	Block incoming and outgoing data if it contains an unknown threat detected by the heuristic analyzer. To enable blocking, the settings must contain the following rule (see below): `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockAdware` *{boolean}*	Block incoming and outgoing data if it contains adware. To enable blocking, the settings must contain the following rule (see below): `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockDialers` *{boolean}*	Block incoming and outgoing data if it contains a dialer.

Parameter	Description
	To enable blocking, the settings must contain the following rule (see below): `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match` Default value: `Yes`
`BlockJokes` *{boolean}*	Block incoming and outgoing data if it contains a joke program. To enable blocking, the settings must contain the following rule (see below): `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match` Default value: `No`
`BlockRiskware` *{boolean}*	Block incoming and outgoing data if it contains riskware. To enable blocking, the settings must contain the following rule (see below): `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match` Default value: `No`
`BlockHacktools` *{boolean}*	Block incoming and outgoing data if it contains a hack tool. To enable blocking, the settings must contain the following rule (see below): `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match` Default value: `No`
`ScanTimeout` *{time interval}*	Timeout for scanning one file at the request of SpIDer Gate. Allowed values: from 1 second (`1s`) to 1 hour (`1h`). Default value: `30s`
`HeuristicAnalysis` *{On | Off}*	Enable or disable the heuristic analysis for detection of unknown threats during file scanning initiated by SpIDer Gate. The heuristic analysis provides higher detection reliability, but, at the same time, increases scanning time. Action applied to threats detected by the heuristic analyzer is specified by the `BlockSuspicious` parameter.

Parameter	Description
	Allowed values: • `On`—enable the heuristic analysis during a scan; • `Off`—disable the heuristic analysis. Default value: `On`
`PackerMaxLevel` *{integer}*	Maximum nesting level for packed objects. A packed object is executable code compressed with special software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects that may also include packed objects and so on. The value of this parameter specifies a nesting limit beyond which packed objects inside other packed objects are not scanned. The nesting level is not limited. If the value is set to 0, nested objects are not scanned. Default value: `8`
`ArchiveMaxLevel` *{integer}*	Maximum nesting level for archives (`.zip`, `.rar` and so on) in which other archives may be enclosed, whereas these archives may also include other archives and so on. The value of this parameter specifies the nesting limit beyond which archives enclosed in other archives are not scanned. The nesting level is not limited. If the value is set to 0, nested objects are not scanned. Default value: `8`
`MailMaxLevel` *{integer}*	Maximum nesting level for mailer files (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects are not scanned. The nesting level is not limited. If the value is set to 0, nested objects are not scanned. Default value: `8`
`ContainerMaxLevel` *{integer}*	Maximum nesting for other types objects inside which other objects are enclosed (HTML pages, `.jar` files and so on). The value of this parameter specifies the nesting limit beyond which objects inside other objects will not be scanned by the request of SpIDer Gate. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned.

Parameter	Description
	Default value: `8`
`MaxCompressionRatio` *{integer}*	Maximum compression ratio of compressed/packed objects (ratio between the uncompressed size and the compressed size). If the ratio of an object exceeds the limit, this object will be skipped during file scanning initiated by SpIDer Gate. The compression ratio must be no less than `2`. Default value: `500`

`InterceptHook`
 	
{path to file | Lua function}
	Script for processing connections in Lua or a path to the file storing this script (see the Processing Connections in Lua section).
	
	If the specified file is unavailable, starting the component causes an error.
	
	Default value:
	
	`local dwl = require 'drweb.lookup'`
	
	`function intercept_hook(ctx)`
	
	`  -- do not check if group == Root.TrustedGroup`
	`  if ctx.divert == "output" and ctx.group == "drweb"`
	`  then`
	`  return "pass"`
	`  end`
	
	`  -- do not check connections from privileged ports`
	`  -- except FTP active mode`
	`  if ctx.src.port >= 0 and ctx.src.port <= 1024`
	`  and ctx.src.port ~= 20`
	`  then`
	`  return "pass"`
	`  end`
	
	`  return "check"`
	`end`

`XtablesLockPath` *{path to file}*	Path to the `iptables` (NetFilter) table blocking file. If the parameter value is not specified, the `/run/xtables.lock` and `/var/run/xtables.lock` paths are checked. If the file is not found in the specified path or default paths, an error occurs upon starting the component. Default value: *(not specified)*

Changes made to the settings of the connection scanning do not influence the scanning of connections that have already been established by applications before making changes. If it is required to apply them to already running applications, it is necessary to force them to disconnect and then connect again, for example, by restarting these applications.

Traffic monitoring and access blocking rules

In addition to the parameters listed above, the section also contains 11 *sets of rules* `RuleSet*` (`RuleSet0`, ..., `RuleSet10`) which directly control traffic scanning and blocking of user access to web resources, as well as blocking downloading content from the internet. For some values in conditions (for example, IP address ranges, lists of website categories, black and white lists of web resources, etc.), there is a substitution of values loaded from text files and also extracted from external data sources via LDAP (the Dr.Web LookupD component is used). When configuring connections, all rules are checked from first to last as a single list, until the rule that triggered the ultimate resolution is found. The gaps in the rule list, if any, are ignored.

The rules are described in detail in section Traffic Monitoring Rules of Appendix D.

Viewing and editing the rules

List gaps, i.e. `RuleSet<i>` sets that do not contain rules (wherein <*i*> is a `RuleSet` rule set number), are kept for editing the rules easily.

You *cannot* add items different to the already present `RuleSet<i>` items, but you can add or remove any rule in any `RuleSet<i>` item.

View and edit the rules in any of the following ways:

- by viewing and editing the configuration file (in any text editor) (note that this file stores only those parameters that values are different from the defaults);
- using management web interface;
- via the command-line interface—Dr.Web Ctl (`drweb-ctl cfshow` and `drweb-ctl cfset` commands).

If you have edited the rules and made changes to the configuration file, in order to apply these changes, restart Dr.Web Gateway Security Suite. To do that, run the `drweb-ctl reload` command.

Viewing the rules using the drweb-ctl cfshow command

To view the contents of the `LinuxFirewall.RuleSet1` rule set, run the command:

```
# drweb-ctl cfshow LinuxFirewall.RuleSet1
```

Editing the rules using the drweb-ctl cfset command

Hereinafter <*rule*> is the text of the rule.

- Replace all the rules in the `LinuxFirewall.RuleSet1` set with a new rule:

```
# drweb-ctl cfset LinuxFirewall.RuleSet1 '<rule>'
```

- Add a new rule to the `LinuxFirewall.RuleSet1` rule set:

```
# drweb-ctl cfset -a LinuxFirewall.RuleSet1 '<rule>'
```

- Remove a specific rule from the `LinuxFirewall.RuleSet1` rule set:

```
# drweb-ctl cfset -e LinuxFirewall.RuleSet1 '<rule>'
```

- Reset the `LinuxFirewall.RuleSet1` rule set to the default state:

```
# drweb-ctl cfset -r LinuxFirewall.RuleSet1
```

When you use the `drweb-ctl` tool to edit the rules, put the text of the <*rule*> rule to be added in single or double quotes and use a backward slash (\) to escape quotes within the text of the rule.

It is important to remember the following aspects of storing the rules in the `RuleSet`<*i*> configuration variables:

- The conditional part and colon can be omitted when adding unconditional rules. However, such rules are always stored in the list of rules as the ' : <*action*>' string.
- When adding rules that contain several actions (such rules as '<*condition*> : <*action 1*>, <*action 2*>'), such rules will be transformed into a chain of elementary rules '<*condition*> : <*action 1*>' and '<*condition*> : <*action 2*>'.
- The rules do not allow for disjunction (logical "OR") of conditions in the conditional part, so, in order to implement the logical "OR", construct a chain of rules with each rule having a disjunct-condition in its condition.

To add an unconditional skipping rule (the `PASS` action) to the `LinuxFirewall.RuleSet1` rule set, run the command:

```
# drweb-ctl cfset -a LinuxFirewall.RuleSet1 'PASS'
```

To remove this rule from the specified rule set, run the command:

```
# drweb-ctl cfset -e LinuxFirewall.RuleSet1 ' : PASS'
```

To add a rule to the `LinuxFirewall.RuleSet1` rule set to change a path to standard templates for connections from forbidden addresses and block these connections, run the command:

```
# drweb-ctl cfset -a LinuxFirewall.RuleSet1 'src_ip not in file("/etc/trusted_ip") : set
http_template_dir = "mytemplates", BLOCK'
```

This command adds *two rules* to the specified rule set, so, in order to remove these rules, run these two commands:

```
# drweb-ctl cfset -e LinuxFirewall.RuleSet1 'src_ip not in file("/etc/trusted_ip") : set
http_template_dir = "mytemplates"'
# drweb-ctl cfset -e LinuxFirewall.RuleSet1 'src_ip not in file("/etc/trusted_ip") :
BLOCK'
```

To add such rule as "Block upon detecting a malicious object of the *KnownVirus* type or a URL from the *Terrorism* category" to the `LinuxFirewall.RuleSet1` rule set, add the following two rules:

```
# drweb-ctl cfset -a LinuxFirewall.RuleSet1 'threat_category in (KnownVirus) : BLOCK as
_match'
# drweb-ctl cfset -a LinuxFirewall.RuleSet1 'url_category in (Terrorism) : BLOCK as
_match'
```

To remove them, run two commands, as shown in the example above.

Default set of rules

By default, the following set of rules for blocking is specified:

```
RuleSet0 =
RuleSet1 = divert output : set HttpTemplatesDir = "output"
RuleSet1 = divert output : set MailTemplatesDir = "firewall"
RuleSet1 = divert input : set HttpTemplatesDir = "input"
RuleSet1 = divert input : set MailTemplatesDir = "server"
RuleSet1 = proc in "LinuxFirewall.ExcludedProc" : PASS
RuleSet1 =  : set Unwrap_SSL = false
RuleSet2 =
RuleSet3 =
RuleSet4 =
RuleSet5 = protocol in (Http), direction request, url_host in "LinuxFirewall.Blacklist"
: BLOCK as BlackList
RuleSet5 = protocol in (Http), direction request, url_host in "LinuxFirewall.Whitelist"
: PASS
RuleSet6 =
RuleSet7 = protocol in (Http), direction request, url_category in
"LinuxFirewall.BlockCategory" : BLOCK as _match
RuleSet8 =
RuleSet9 = protocol in (Http), divert input, direction request, threat_category in
"LinuxFirewall.BlockThreat" : BLOCK as _match
RuleSet9 = protocol in (Http), direction response, threat_category in
"LinuxFirewall.BlockThreat" : BLOCK as _match
RuleSet9 = protocol in (Smtp), threat_category in "LinuxFirewall.BlockThreat" : REJECT
RuleSet9 = protocol in (Smtp), url_category in "LinuxFirewall.BlockCategory" : REJECT
RuleSet9 = protocol in (Smtp), total_spam_score gt 0.80 : REJECT
RuleSet9 = protocol in (Pop3, Imap), threat_category in "LinuxFirewall.BlockThreat" :
REPACK as _match
RuleSet9 = protocol in (Pop3, Imap), url_category in "LinuxFirewall.BlockCategory" :
REPACK as _match
RuleSet9 = protocol in (Pop3, Imap), total_spam_score gt 0.80 : REPACK as _match
RuleSet10 =
```

The first rule indicates that if the connection is established by the process specified in the `ExcludedProc` parameter (see above), the connection is skipped without checking any other

conditions. The next rule (executed without any condition) disables scanning of secure connections. This and all the rules below are analyzed only if the connection is not associated with the excluded process. Moreover, as all subsequent rules depend on the protocol type, if scanning of secure connections is disabled and the connection is secure, the rules are not executed because it is impossible to define whether the conditions are true.

The following five rules regulate processing of outgoing HTTP connections:

1. If the host to which a connection is established is included in a black list, the connection is blocked without performing other checks.
2. If the host is included in a white list, the connection is skipped without performing other checks.
3. If a URL requested by the client belongs to a category of unwanted web resources, the connection is blocked without performing other checks.
4. If the response received from a remote host via HTTP contains a threat belonging to the blocked categories, the connection is blocked without performing other checks.
5. If the data transferred from the local host to a remote host contains a threat belonging to the blocked categories, the connection is blocked without performing other checks.

These five rules are triggered only if `On` is specified in the `InspectHttp` parameter. Otherwise, none of these rules is triggered.

The following six rules specified in `RuleSet9` control the scanning of data sent and received via email protocols (SMTP, POP3 or IMAP); these rules are triggered in the following cases:

- the message contains attachments;
- the message contains a URL belonging to blocked categories;
- the message is qualified as spam with the spam index of no less than 0.8.

If the message is transmitted via the SMTP protocol, an action blocking the transmission (i.e. sending or receipt) of the message is applied, whereas, in case of the IMAP and POP3 protocols, the message is processed such that malicious contents is removed (“repackaging”).



If the Dr.Web Anti-Spam component for scanning of an email message for signs of spam is unavailable, scanning of email messages for signs of spam is not performed. In this case, rules that contain analyzing of a spam level (the `total_spam_score` variable) are unavailable.

The rules for scanning email messages will be triggered only if the corresponding `Inspect<`*EmailProtocol*`>` parameters are set to `On`. Otherwise, none of these rules are triggered.

The Dr.Web MailD component for email scanning is required for scanning transmitted email messages for malware attachments and signs of spam. If the component is not installed, transmitted email will be blocked because of the error *“Unable to check”*. To allow transmitting messages that cannot be checked, set the `BlockUnchecked` parameter to `No` (see above).

Moreover, if the email scanning component is not installed, it is recommended to specify `No` for the `InspectSmtp`, `InspectPop3`, and `InspectImap` parameters.

Dr.Web MailD is not included in Dr.Web Gateway Security Suite.

Examples of the traffic monitoring and access blocking rules

1. Allow users with IP addresses in the range of *10.10.0.0–10.10.0.254* an access via HTTP to websites of all categories except *Chats*:

```
protocol in (HTTP), src_ip in (10.10.0.0/24), url_category not in (Chats) : PASS
```

If the rule:

```
protocol in (HTTP), url_host in "LinuxFirewall.Blacklist" : BLOCK as
BlackList
```

is put on the list of rules above the indicated rule, then access to the domains from the black list, that is, the domains listed in the `LinuxFirewall.Blacklist` parameter, will also be blocked for users with IP addresses in the range of *10.10.0.0–10.10.0.254*. At the same time, if this rule is put below, the users with the IP addresses in the range of *10.10.0.0–10.10.0.254* will also get access to websites from the black list.

Since the `PASS` resolution is final, no more rules are used; therefore, the downloaded data is not scanned for threats.

To allow users with IP addresses in the range of *10.10.0.0–10.10.0.254* to access websites of all categories except *Chats*, if they are not on the black list, and to block downloading of threats at the same time, use the following rule:

```
protocol in (HTTP), url_category not in (Chats), url_host not in
"LinuxFirewall.Blacklist", threat_category not in "LinuxFirewall.BlockCategory" :
PASS
```

2. Do not perform scanning of contents of video files *downloaded from the internet* (i.e. data of the `video/*` MIME type, where `*` corresponds to any type of the `video` MIME class):

```
direction response, content_type in ("video/*") : PASS
```

The files uploaded from the local computer (including those that have the `video/*` MIME type) will be scanned because they are sent *in requests, and not in responses*, i.e. the `direction` variable has the `request` value for them.

9.6.4. Processing Connections in Lua

In this section

- General information
- Requirements for connection processing script
- Examples
- Tables in use (InterceptContext, TcpEndpoint)
- Available auxiliary modules (drweb, drweb.lookup)

General information

The Dr.Web Firewall for Linux component supports interaction with the Lua interpreter (version 5.3.4 is supplied with Dr.Web Gateway Security Suite). Lua scripts can be used by Dr.Web Firewall for Linux for scanning a connection in advance prior to analyzing it using SpIDer Gate.

The connection will be analyzed using a script, if a path to this script is specified in Dr.Web Firewall for Linux settings (the `InterceptHook` parameter). Otherwise, connection processing is performed by using default settings and processing rules specified in the component settings (`RuleSet*` parameters).



For more examples of connection processing scripts, follow the link.

Requirements for connection processing script

The script must contain a global function being an entry point in the connection scanning module (Dr.Web Firewall for Linux calls this function for processing newly received connections). The function must match the following calling conventions:

- *function name*—`intercept_hook`;
- *the only argument*—Lua InterceptContext table (provides access from the function to information about the connection being processed; see the table description below);
- *the only return value*—string value from the table below:

Value	Verdict description
`pass`	Skip the connection without using SpIDer Gate to scan it
`check`	Scan the connection using SpIDer Gate
`reject`	Reject the connection (the client that initiated the connection will receive a TCP package with an `RST` flag)

Value	Verdict description
`drop`	Disconnect (the client that initiated the connection will receive no acknowledgement)

Examples

1. A simple script returning the `pass` verdict to Dr.Web Firewall for Linux, which indicates that the connections are not to be scanned:

```
-- Connection scanning function written by the user
function intercept_hook(ctx)
  return "pass" -- do not scan the connection
end
```

2. The script instructs Dr.Web Firewall for Linux to scan all connections being established with the exception of:
   - outgoing local connections from applications running with the privileges of the user from the `drweb` group;
   - connections from privileged ports (regardless of the owner of a connection and its direction);
   - outgoing connections from IP addresses of the local subnet.

```
function intercept_hook(ctx)
  -- Do not scan connections initiated from the local
  -- host (divert == "output") by an application on behalf of the group
  -- "drweb" (group == "drweb")
  if ctx.divert == "output" and ctx.group == "drweb" then
    return "pass"
  end

  -- Do not scan connections initiated from
  -- privileged ports (the range is from 0 to 1024)
  if ctx.src.port >= 0 and ctx.src.port <= 1024 then
    return "pass"
  end

  -- Do not scan connections from local subnet addresses
  -- (the IP address range of 127.0.0.1/8)
  if ctx.src.ip.belongs("127.0.0.0/8") then
    return "pass"
  end

  -- Connection is scanned by default
  return "check"
end
```

Tables in use

InterceptContext table

The table is used to pass data on the connection being processed to the `intercept_hook` function. One of the following actions can be performed on the basis of this data:

- skip the connection,
- disconnect,
- pass the connection to SpIDer Gate for scanning.

The Dr.Web Firewall for Linux component fills this table with data. Some data in the table is already available by the time the `intercept_hook` function is called, the remaining (so called "lazy") data will be calculated directly upon quering the corresponding field of this table.

Field	Description	Data type

`src`	Address and port of the client that initiated the connection
	Example:
	`if ctx.src.port >= 0 and ctx.src.port <= 1024 then`
	`  return "pass"`
	`end`
		TcpEndpoint table

`dst`	Address and port of the server, the connection to which was initiated by the client
	Example:
	`if ctx.dst.ip.belongs("10.20.30.41/8") then`
	`  return "reject"`
	`end`
		TcpEndpoint table

`divert`	Type of an intercepted connection:
	• `output`—outgoing connection;
	• `input`—incoming connection;
	• `forward`—transit connection.
	Example:
	`if ctx.divert == "forward" then`
	`  return "check"`
	`end`
		String

`iface_in`	Name of the interface from which the connection was initiated. If the name of the interface was not identified, the field has the `nil` value.	String
`iface_out`	Name of the interface to which packets were sent after initiating the connection.	String

Field	Description	Data type
	If the name of the interface was not identified, the field has the `nil` value.	
`uid`	Identifier of the user on behalf of whom the outgoing connection was initiated. If the connection type (`divert`) is not `output` or UID cannot be identified, the field has the `nil` value.	Integer
`gid`	Identifier of the group on behalf of which the outgoing connection was initiated. If the connection type (`divert`) is not `output` or GID cannot be identified, the field has the `nil` value.	Integer
`user`	User on behalf of whom the outgoing connection was initiated. If the connection type (`divert`) is not `output` or the user name cannot be identified, the field has the `nil` value.	String
`group`	Group on behalf of which the outgoing connection was initiated. If the connection type (`divert`) is not `output` or the group name cannot be identified, the field has the `nil` value.	String
`pid`	Identifier of the process (PID) on behalf of which the outgoing connection was initiated. If the connection type (`divert`) is not `output` or the PID cannot be identified, the field has the `nil` value.	Integer
`exe_path`	Executable path of the application that initiated the outgoing connection. If the `divert` field (see above) has the `input` or `forward` value or the executable path cannot be identified, the field has the `nil` value.	String
Overridden metamethods: *None*		

TcpEndpoint table

The table describes a client or server connection point address.

Field	Description	Data type
`ip`	IP address	IpAddress table
`port`	Port number	Integer

Field	Description	Data type
Overridden metamethods: • `__tostring` is a function that transforms `TcpEndpoint` into a string, for example, `"127.0.0.1:443"` (IPv4) or `"[::1]:80"` (IPv6); • `__concat` is a function that joins `TcpEndpoint` to a string		

Available auxiliary modules

The following custom modules listed in the table can be imported into Lua program space for interaction with Dr.Web Gateway Security Suite.

Name of the module	Function
drweb	Module that provides functions for writing messages from a Lua program to the log of the Dr.Web Gateway Security Suite component that started this program, as well as means to run Lua procedures asynchronously
drweb.lookup	Module that provides tools to query data from external sources using the Dr.Web LookupD module

Contents of the drweb module

1. Functions

The module provides a set of functions.

- Storing messages from the Lua program in the Dr.Web Gateway Security Suite component log:
  - `log(`<*level*>, <*message*>`)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the <*level*> level (the required level is defined using one of the following values (a string): *debug*, *info*, *notice*, *warning*, *error*);
  - `debug(`<*message*>`)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *debug* level;
  - `info(`<*message*>`)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *info* level;
  - `notice(`<*message*>`)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *notice* level;
  - `warning(`<*message*>`)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *warning* level;
  - `error(`<*message*>`)` writes the <*message*> string to the Dr.Web Gateway Security Suite log at the *error* level.

- Managing synchronization of Lua procedures:
  - `sleep(<sec.>)` pauses execution of a Lua procedure instance for a specified number of seconds;
  - `async(<Lua function>[, <argument list>])` starts the specified function asynchronously and passes the specified argument list to it. The `async` function call completes immediately, and the return value (the `Future` table) allows you to obtain the result of the *<Lua function>*.
- Storing the information about an IP address in the form of the IpAddress table:
  - `ip(<address>)` indicates an IP address (IPv4 or IPv6) in the form of the IpAddress table.
- Retrieving external data from a text file:
  - `load_set(<file path>)` generates a table with its values assigned to `true` from the contents of a specified text file; strings read from the file are used as keys. Empty strings and strings consisting only of white spaces are ignored.
  - `load_array(<path to file>)` generates a string array from the contents of a specified text file. Empty strings and strings consisting only of white spaces are ignored.

2. Tables

- The `Future` table describes a pending result of performing a function using the `async` function.

Field	Description	Data type
`wait`	Function that returns a result of the function started using the `async` function. If the function has not completed its execution yet, it waits for the completion and returns its result. If the function is completed before `wait` is called, the result is returned immediately. If the started function fails, the `wait` call generates the same error.	Function
Overridden metamethods: *None*		

- The `IpAddress` table describes an IP address.

Field	Description	Data type
`belongs`	Function checks an IP address stored in the `IpAddress` table for belonging to specified subnets (IP address ranges). Accepts a single argument—an array of strings such as *<IP address>* or *<IP address>*/*<mask>*, where *<IP address>* is a host address or a network address (for example, `127.0.0.1`), and *<mask>* is a subnet mask (can be specified as an IP address, for example, `255.0.0.0`, or as an integer, for example, `8`). Returns a Boolean value: • `true`, if the address matches at least one of the specified IP	Function

Field	Description	Data type
	addresses or belongs to at least one of the specified subnets (a range of IP addresses); • `false`, if the address does not match any one of the specified IP addresses or does not belong to any of the specified subnets	

Overridden metamethods:

- `__tostring` is a function that transforms `IpAddress` into a string, for example, `127.0.0.1` (IPv4) or `::1` (IPv6);
- `__concat` is a function that joins `IpAddress` to a string;
- `__eq` is a function that checks whether two `IpAddress` are equal;
- `__band` is a function that allows applying a mask, for example: `dw.ip('192.168.1.2') & dw.ip('255.255.254.0')`

3. Examples

- Log messages generated by a procedure started asynchronously:

```
local dw = require "drweb"

-- This function returns a string received as an argument
-- after a delay of two seconds
function out_msg(message)
 dw.sleep(2)
 return message
end

-- "Main" function
function intercept(ctx)
 -- Output a string at the Notice level to the log of Dr.Web Gateway Security Suite
 dw.notice("Intercept function started.")

 -- Run two instances of the out_msg function asynchronously
 local f1 = dw.async(out_msg, "Hello,")
 local f2 = dw.async(out_msg, " world!")

 -- Wait for the completion of the instances of the function
 -- out_msg and output their results
 -- to the Dr.Web Gateway Security Suite log at the Debug level
 dw.log("debug", f1.wait() .. f2.wait())
end
```

- Create a scheduled procedure:

```
local dw = require "drweb"

-- Store the Future table as a future global variable to
-- prevent its removal by the garbage collector
future = dw.async(function()
   while true do
     -- Log the following message every day
     dw.sleep(60 * 60 * 24)
     dw.notice("A brand new day began")
   end
end)
```

- Transform a string to an IP address:

```
local dw = require "drweb"

local ipv4 = dw.ip("127.0.0.1")
local ipv6 = dw.ip("::1")
local mapped = dw.ip("::ffff:127.0.0.1")
```

Contents of the drweb.lookup module

1. Functions

The module provides the following functions:

- `lookup(<query>, <parameters>)` queries data from an external storage available via the Dr.Web LookupD module. The *<query>* argument must correspond to a section in Dr.Web LookupD settings (the *<type>*@*<tag>* string). The *<parameters>* argument is optional and describes substitutions to be applied to generate the query. The following automatically resolvable tokens can be used:
  - `$u`, `$U` (replaced with `user`)—user name sent by the client component;
  - `$d`, `$D` (replaced with `domain`)—domain name sent by the client component.

  Arguments are set as a table whose keys and values must be strings. The function returns an array of strings that are query results.
- `check(<checked string>, <query>, <parameters>)` returns `true` if *<checked string>* is found in an external repository accessible via the Dr.Web LookupD module. The *<query>* and *<parameters>* arguments are equivalent to the arguments of the `lookup` function (see above). The *<checked string>* argument must be a string or a table with a `__tostring` metamethod (i.e. that can be transformed into a string).

2. Examples

- Log a list of users retrieved from the `LookupD.LDAP.users` data source:

```
local dw = require "drweb"
local dwl = require "drweb.lookup"

-- "Main" function
function intercept(ctx)
 -- Store the string in the Dr.Web Gateway Security Suite log at the Notice level
 dw.notice("Intercept function started.")

 -- Store in the Dr.Web Gateway Security Suite log the results of quering
 -- the 'ldap@users' data source
 for _, s in ipairs(dwl.lookup("ldap@users", {user="username"})) do
   dw.notice("Result of request to 'ldap@users': " .. s)
 end

end
```

9.7. Dr.Web ClamD

The Dr.Web ClamD component emulates the interface of the `clamd` anti-virus daemon, which is a core component of the Clam AntiVirus (ClamAV®) anti-virus product from Sourcefire, Inc. This interface allows external applications using ClamAV® to scan files with Dr.Web Gateway Security Suite.

9.7.1. Operating Principles

The Dr.Web ClamD component is designed to scan, upon the request of external applications, both the content of files of the local file system and streams of data transmitted by an external application via a socket. Furthermore, the component can scan the contents of those files for which an external application passed an open file descriptor via a socket.



File scans based on a passed file descriptor can be performed only if the descriptor was passed via a local Unix socket.

If an external application has provided a path to a file in the local file system, the component sends the scanning task to Dr.Web File Checker; otherwise, the component transmits data received via the socket to Dr.Web Network Checker.

By default, the component is not automatically started together with Dr.Web Gateway Security Suite. To enable starting of the component, it is necessary not only to adjust the `Start` parameter, but also to define at least one connection point for client applications. After that, the component starts waiting for external application requests for scanning files or data streams. You can configure multiple connection points for external applications in the settings and set individual scanning parameters for each of the points.

The external applications can be specifically represented by HTTP proxy servers (such as Squid and HAVP), if they have an integration module with Dr.Web ClamD. For details, see Integration with External Applications.



Detected threats *are not neutralized* by Dr.Web Gateway Security Suite; the external application only receives the scanning results. Thus, any detected threat should be neutralized by the external application itself.

9.7.2. Command-Line Arguments

To start the Dr.Web ClamD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-clamd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-clamd [<parameters>]
```

Dr.Web ClamD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-clamd --help
```

This command outputs short help information about the Dr.Web ClamD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary (usually, at the startup of the operating system). To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-clamd` command.

9.7.3. Configuration Parameters

In this section

- Component parameters
- Special aspects of component configuration

The component uses configuration parameters specified in the `[ClamD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

Component parameters

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-clamd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-clamd`
`Start` *{boolean}*	The component is started by the Dr.Web ConfigD configuration management daemon. Setting this parameter to `Yes` instructs the configuration management daemon to start the component immediately, and setting this parameter to `No`—to shut down the component immediately. Default value: `No`
`Endpoint.<tag>.ClamdSocket` *{IP address | Unix socket}*	Mount point named <*tag*> and a socket (IPv4 address or a Unix socket address) for clients who need to scan files for threats. Only one socket can be specified for one <*tag*> point. Default value: not set
`[Endpoint.<tag>.]ReadTimeout` *{time interval}*	Time-out for waiting data from a client. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `5s`

Parameter	Description
`[Endpoint.<tag>.]StreamMaxLength` *{size}*	Maximum size of data that can be received from a client (while passing data for scanning as a stream of bytes). If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `25mb`
`[Endpoint.<tag>.]ScanTimeout` *{time interval}*	Time-out for scanning one file or one chunk of data received from a client. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Allowed values: from 1 second (`1s`) to 1 hour (`1h`). Default value: `3m`
`[Endpoint.<tag>.]HeuristicAnalysis` *{On | Off}*	Enable heuristic analysis during scanning. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `On`
`[Endpoint.<tag>.]PackerMaxLevel` *{integer}*	Maximum nesting level for packed objects. A packed object is executable code compressed with special software (UPX, PELock, PECompact, Petite, ASPack, Morphine and so on). Such objects may include other packed objects which may also include packed objects and so on. The maximum nesting level is the limit beyond which packed objects inside other packed objects are not scanned. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[Endpoint.<tag>.]ArchiveMaxLevel` *{integer}*	Maximum nesting level for archives (`.zip`, `.rar` and so on). Archives may include archives in which other

Parameter	Description
	archives may also be enclosed and so on. The maximum nesting level is the limit beyond which archives inside archives are not scanned. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[Endpoint.<tag>.]MailMaxLevel` *{integer}*	Maximum nesting level for files of mailers (`.pst`, `.tbb` and so on) in which other files may be enclosed, whereas these files may also include other files and so on. The value of this parameter specifies the nesting limit beyond which objects inside other objects will not be scanned. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[Endpoint.<tag>.]ContainerMaxLevel` *{integer}*	Maximum nesting level for other types of objects containing other objects (for instance, HTML pages or `.jar` files). These objects may include other objects, which in turn may also include others and so on. The maximum nesting level is the limit beyond which objects inside objects are not scanned. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. The nesting level is not limited. If the value is set to `0`, nested objects are not scanned. Default value: `8`
`[Endpoint.<tag>.]MaxCompressionRatio` *{integer}*	Maximum allowed compression ratio of compressed/packed objects (ratio between the uncompressed size and the compressed size). If the ratio of an object exceeds the limit, this object is skipped during a scan.

Parameter	Description
	If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. The compression ratio must be no less than `2`. Default value: `500`
`[Endpoint.<tag>.]DetectSuspicious` *{boolean}*	Inform about suspicious files detected by the heuristic analyzer. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `Yes`
`[Endpoint.<tag>.]DetectAdware` *{boolean}*	Inform about files containing adware. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `Yes`
`[Endpoint.<tag>.]DetectDialers` *{boolean}*	Inform about files containing dialers. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `Yes`
`[Endpoint.<tag>.]DetectJokes` *{boolean}*	Inform about files containing jokes. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `No`
`[Endpoint.<tag>.]DetectRiskware` *{boolean}*	Inform about files containing riskware. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `No`

Parameter	Description
`[Endpoint.<tag>.]DetectHacktools` *{boolean}*	Inform about files containing hacktools. If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter. Default value: `No`

`[Endpoint.<tag>.]`
`ReportPasswordProtected`
 	
{integer}
	Report an error, if the scanned files are protected with a password.
	
	If the `Endpoint.<tag>` prefix is specified, the parameter value is specified only for the <*tag*> point; otherwise, it is specified for all points that do not have a defined value of this parameter.
	
	Default value: `No`

Special aspects of component configuration

Parameters marked with an optional `Endpoint.<tag>` prefix can be grouped. Each group defines a unique connection *point* (*endpoint*) that can be used by clients to connect to the component and has a unique <*tag*> identifier assigned to it. All the scanning parameters belonging to the same group define the settings that are applicable only when data is scanned for the clients connected to the corresponding connection point. If a parameter is specified without an `Endpoint.<tag>` prefix, this sets the value for all connection points. If you delete some parameter from some connection point, the program will use a value of the corresponding "parent" parameter of the same name (set without the `Endpoint.<tag>` prefix) instead of a default value.



The `ClamdSocket` parameter must always be specified with an `Endpoint.<tag>` prefix, as it defines both a listening socket and a group (connection point) to which this socket corresponds.

Example

Let us assume that we need to set up two connection points for two groups of external applications (servers)—let the groups be called *servers1* and *servers2*. The servers from the *servers1* group can connect through a Unix socket, whereas the servers form the *servers2* group can use a network connection. Moreover, let us assume that the heuristic analysis must be disabled by default, but must be used for the servers from the *servers2* group. The following example shows how to configure this:

1) In the configuration file:

```
[ClamD]
HeuristicAnalysis = Off
```

```
[ClamD.Endpoint.servers1]
ClamdSocket = /tmp/srv1.socket

[ClamD.Endpoint.servers2]
ClamdSocket = 127.0.0.1:1234
HeuristicAnalysis = On
```

2) Using the Dr.Web Ctl command-line management tool:

```
# drweb-ctl cfset ClamD.HeuristicAnalysis Off
# drweb-ctl cfset ClamD.Endpoint -a servers1
# drweb-ctl cfset ClamD.Endpoint -a servers2
# drweb-ctl cfset ClamD.Endpoint.servers1.ClamdSocket /tmp/srv1.socket
# drweb-ctl cfset ClamD.Endpoint.servers2.ClamdSocket 127.0.0.1:1234
# drweb-ctl cfset ClamD.Endpoint.servers2.HeuristicAnalysis On
```



Both ways have the same effect; however, if you edit the configuration file directly, you will also need to apply the changed settings by running the `drweb-ctl reload` command.

9.7.4. Integration with External Applications

The emulation of the interface of the `clamd` anti-virus daemon included in the ClamAV anti-virus solution makes it possible to integrate Dr.Web ClamD with any external applications capable of connecting to the `clamd` anti-virus daemon.

The table below shows examples of applications that can use `clamd` for anti-virus scans:

Product	Integration
HTTP services	
HTTP proxy server Squid	**Use of clamd** Scan files received from the internet. **Integration requirements** Use an intermediate component: `squidclamav` or HAVP. **Links to documentation** Squid documentation. Description and source code of squidclamav

HTTP proxy server that can perform anti-virus scans HAVP	**Use of clamd**
	Scan files received from the internet.
	Integration requirements
	Enable `clamd` for anti-virus scanning in the HAVP configuration file:
	`ENABLECLAMD true`
	`CLAMDSOCKET <path_to_clamd_Unix_socket>`

Product	Integration

	or (if a TCP connection is used instead of a Unix socket):
	`ENABLECLAMD true`
	`CLAMDSERVER <IP>`
	`CLAMDPORT <port>`
	where <*path_to_clamd_Unix_socket*> or the <*IP*>:<*port*> pair corresponds to the socket of the *endpoint* configured in Dr.Web ClamD.
	Links to documentation
	HAVP product documentation

In the settings of the external software component that communicates directly with Dr.Web ClamD as with the `clamd` anti-virus daemon, specify a path to a Unix socket or a TCP socket listened by Dr.Web ClamD at one of its *endpoints* set up in its configuration as an address for connecting to `clamd`.

Example of connecting HAVP to Dr.Web ClamD:

1. Dr.Web ClamD configuration:

```
[ClamD]
Start = yes

[ClamD.Endpoint.proxy]
ClamdSocket = /var/run/drweb.clamd
```

2. HAVP configuration:

```
ENABLECLAMD true
CLAMDSOCKET /var/run/drweb.clamd
```

Settings for connecting to other anti-virus products (`ENABLE*` parameters) must be set to `false`.

9.8. Dr.Web File Checker

The Dr.Web File Checker file scanning component is designed for scanning files and directories in the file system. It is used by other components of Dr.Web Gateway Security Suite to scan file system objects. In addition, this component permanently registers all threats detected in the file system and also functions as a quarantine manager, as it manages the contents of the directories where isolated files are kept.

9.8.1. Operating Principles

The Dr.Web File Checker component is started with superuser (the *root* user) privileges and scans file system objects (files, directories and boot records).

Dr.Web File Checker indexes all scanned files and directories and stores data about scanned objects in a special cache to avoid repeated scanning of the objects that have been already scanned and have not been modified since that (in this case, if a request to scan such an object is received, the previous scan result retrieved from the cache is returned).

When requests to scan file system objects are received from other components, Dr.Web File Checker checks whether the requested object requires scanning. If so, a scanning task is generated for Dr.Web Scanning Engine. If the scanned object contains a threat, Dr.Web File Checker puts it in the registry of detected threats and applies an action to neutralize it (cure, delete or quarantine), if this action has been specified by the client component that initiated the scanning as a reaction to the threat. The scanning can be initiated by various components of Dr.Web Gateway Security Suite.

During scanning of the requested file system objects, the file-checking component generates a report detailing scanning results and applied actions to neutralize threats, if any, and sends this report to the client component that requested scanning.

Apart from the standard file scanning method, the following special methods are available for internal use:

- *The "flow" method*—a method for scanning files in stream. A component, which uses this method, initializes parameters of scanning and threat neutralization only once. These parameters are further applied to all file scanning requests from this component.
- *The "proxy" method*—a method for file scanning consisting in that the file-checking component only scans files for threats without applying any actions to them and without registering the detected threats (these actions are fully delegated to the component that initiated the scanning). This method is used by the Dr.Web ClamD component.

During its operation, the file-checking component not only maintains threat registry and manages quarantine, but also collects overall file scan statistics, averaging the number of files scanned per second for the last minute, last 5 minutes, last 15 minutes.

9.8.2. Command-Line Arguments

To start the Dr.Web File Checker component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-filecheck [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-filecheck [<parameters>]
```

Dr.Web File Checker accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-filecheck --help
```

This command outputs short help information about the Dr.Web File Checker component.

Startup notes

The component cannot be started directly from the command line in standalone mode. It is started automatically by the Dr.Web ConfigD configuration management daemon upon receiving requests for file system scanning from other components of Dr.Web Gateway Security Suite. To manage the operation parameters of the component as well as to scan files on demand, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To scan an arbitrary file or directory using Dr.Web File Checker, use the `scan` command of the Dr.Web Ctl tool:

```
$ drweb-ctl scan <path to file or directory>
```

To get documentation on this component from the command line, run the `man 1 drweb-filecheck` command.

9.8.3. Configuration Parameters

The component uses configuration parameters specified in the `[FileCheck]` section of the unified configuration file of Dr.Web Gateway Security Suite.

This section stores the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-filecheck` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-filecheck`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`
`DebugClientIpc` *{boolean}*	Log or do not log detailed IPC messages at the debug level (with `LogLevel = DEBUG`). Default value: `No`

Parameter	Description
`DebugScan` *{boolean}*	Log or do not log detailed messages received during file scanning at the debug level (with `LogLevel = DEBUG`). Default value: `No`
`DebugFlowScan` *{boolean}*	Log or do not log detailed messages about file scanning using the *"flow"* method at the debug level (with `LogLevel = DEBUG`). Default value: `No`
`DebugProxyScan` *{boolean}*	Log or do not log detailed messages about file scanning using the *"proxy"* method at the debug level (with `LogLevel = DEBUG`). The *"proxy"* method is usually used by the Dr.Web ClamD component. Default value: `No`
`DebugCache` *{boolean}*	Log or do not log detailed messages about the cache status of scanned files at the debug level (with `LogLevel = DEBUG`). Default value: `No`
`MaxCacheSize` *{size}*	Maximum allowed size of cache to store data about scanned files. If `0` is specified, caching is disabled. Default value: `50mb`
`RescanInterval` *{time interval}*	Period of time during which a file will not be rescanned if the results of its previous scan are available in the cache (the period during which the stored information is considered up-to-date). Allowed values: from 0 seconds (`0s`) to 1 minute (`1m`). If the set interval is less than `1s`, there will be no delay—the file will be scanned upon any request. Default value: `1s`

9.9. Dr.Web Network Checker

The Dr.Web Network Checker component is designed to scan data received over the network, with the scanning engine, as well as for distributed file scanning for threats. The component allows to establish a connection between network hosts with Dr.Web Gateway Security Suite installed on them for receiving and sending data (for example, file contents) via the network hosts for scanning. The component manages automatic distribution of scanning tasks (by sending and receiving them over the network) to all available network hosts with which the connection is configured. The component balances the load caused by the scanning tasks between the hosts. If there are no configured connections with remote hosts, the component sends all the data only to the local instance of Dr.Web Scanning Engine.



This component is always used to scan the data received over network connections. Thus, if the component is absent or unavailable, the operation of the components that send data for scanning over a network connection will be hindered (Dr.Web ClamD, SpIDer Gate, Dr.Web ICAPD).

This component is not designed to manage distributed scanning of files located in a local file system, since it cannot replace the Dr.Web File Checker component. To manage distributed scanning of local files, use the Dr.Web MeshD component.

If data scanning over the network is highly intensive, issues with scanning may arise when available file descriptors are depleted. In this case, it is necessary to increase the limit by the number of file descriptors available to Dr.Web Gateway Security Suite.

Data can be shared either over an insecure channel or over a secure channel via SSL/TLS. To use a secure connection it is required to provide hosts that share files with valid certificates and SSL keys. To generate keys and certificates, you can use the `openssl` utility. An example of how to use the `openssl` utility to generate certificates and private keys is given in the Appendix E. Generating SSL Certificates section.

9.9.1. Operating Principles

The component allows sending the data not represented by files of a local file system for scanning to Dr.Web Scanning Engine located on a local or remote host. This data is processed by the components that send data for scanning over a network connection (SpIDer Gate, Dr.Web ICAPD, Dr.Web ClamD).



These components always use Dr.Web Network Checker (even if it is located on a local host) to send files for scanning to Dr.Web Scanning Engine. Thus, if Dr.Web Network Checker is unavailable, these components *cannot operate correctly*.

In addition, Dr.Web Network Checker allows to connect Dr.Web Gateway Security Suite to a given set of hosts on the network that run Dr.Web Gateway Security Suite (or any other Dr.Web

for Unix solution 10.1 or later) in order to manage a distributed search for data not represented by files of the local file system. Thereby, this component allows to create and configure a *scanning cluster*, which is a set of network hosts that exchange data for scanning (each host must run its own instance of the Dr.Web Network Checker distributed scanning agent). On each network host comprised by the scanning cluster, Dr.Web Network Checker performs automatic distribution of tasks for scanning data, sending them over the network to all available hosts for which the connection is configured. At the same time, the host load balancing caused by data scanning is performed depending on the amount of resources available on remote hosts. The number of child scanning processes generated by Dr.Web Scanning Engine on a host comprised by the cluster indicates the amount of resources available for load. The lengths of the queues of the files for scanning on each host in use are also considered.

In this case, any network host comprised by the scanning cluster can act both as a scanning client that sends data for remote scanning and as a scanning server that receives data from the specified network hosts for scanning. If necessary, the distributed scanning agent can be configured so that the host acts only as a scanning server or only as a scanning client.

The data received over the network for scanning is stored on the local file system as temporary files and sent to Dr.Web Scanning Engine or, in case it is unavailable or heavily loaded, to another host of the scanning cluster.

The `InternalOnly` parameter of the component settings allows to manage the Dr.Web Network Checker operation mode. The parameter defines if the component is used for including Dr.Web Gateway Security Suite in the scanning cluster or only for internal purposes of the Dr.Web Gateway Security Suite components operating locally.



You can create your own component (external application) which will use Dr.Web Network Checker to scan files (including distributing scanning tasks between the hosts of the scanning cluster). For this, the Dr.Web Network Checker component provides a custom API based on the Google Protobuf technology. The description of the Dr.Web Network Checker API , as well as client application sample code that uses Dr.Web Network Checker, are supplied as part of the `drweb-netcheck` package.

An example of creating a scanning cluster can be found in the Creating a Scanning Cluster section.

9.9.2. Command-Line Arguments

To start the Dr.Web Network Checker component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-netcheck [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-netcheck [<parameters>]
```

Dr.Web Network Checker accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-netcheck --help
```

This command outputs short help information about the Dr.Web Network Checker component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary (usually, at the startup of the operating system). At the same time, if a `FixedSocket` parameter value is specified in the component settings and the `InternalOnly` parameter is set to `No`, the component will be started by the configuration management daemon and always available to clients via a Unix socket. To manage component operation parameters and to start network scanning (if there is a configured connection to other network hosts), use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line. If there is no configured connection to other network hosts, normal scanning will be started using the local scanning engine instead of network scanning.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To scan an arbitrary file or directory with Dr.Web Network Checker, use the `netscan` command of the Dr.Web Ctl tool:

```
$ drweb-ctl netscan <path to file or directory>
```

To get documentation on this component from the command line, run the `man 1 drweb-netcheck` command.

9.9.3. Configuration Parameters

The component uses configuration parameters specified in the `[NetCheck]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-netcheck` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-netcheck`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. If the `LoadBalanceAllowFrom` or `FixedSocket` parameter is set, this setting is ignored (the component does not finish its operation after the time interval expires). Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`

Parameter	Description

`FixedSocket`
{path to file | address}
	Socket of the Dr.Web Network Checker agent fixed instance.
	If this parameter is specified, the Dr.Web ConfigD configuration management daemon ensures that there is always a running instance of the distributed scanning agent available to clients via this socket.
	Allowed values:
	• *<path to file>*—path to a local Unix socket;
	• *<address>*—network socket set as an *<IP address>*:*<port>* pair.
	⚠ Ensure that the following permissions are assigned to the directory with the socket file:
	`drwxr-xr-x`
	To view the permissions, run the command:
	`$ ls -ld <path to directory>`
	Default value: *(not specified)*

`InternalOnly` *{boolean}*	Component operation mode. If the value is set to `Yes`, the component is used for internal purposes of Dr.Web Gateway Security Suite components only, but not for servicing a scanning cluster or external (to Dr.Web Gateway Security Suite) client applications regardless of `LoadBalance*` settings and the specified value of the `FixedSocket` parameter. Default value: `No`
`LoadBalanceUseSsl` *{boolean}*	Use or do not use SSL/TLS to connect to other hosts. Allowed values: • `Yes`—use SSL/TLS; • `No`—do not use SSL/TLS. If the parameter is set to `Yes`, a certificate and a private key must be specified for this host and for all hosts with which it interacts (the `LoadBalanceSslCertificate` and `LoadBalanceSslKey` parameters). Default value: `No`
`LoadBalanceSslCertificate` *{path to file}*	Path to the SSL certificate used by Dr.Web Network Checker on the current host for communication with other hosts via a secure SSL/TLS connection.

Parameter	Description
	 The certificate file and the private key file (specified by the `LoadBalanceSslKey` parameter) must match each other. Default value: *(not specified)*
`LoadBalanceSslKey` *{path to file}*	Path to the private key file used by Dr.Web Network Checker on the current host for communication with other hosts via a secure SSL/TLS connection.  The certificate file (specified by the `LoadBalanceSslCertificate` parameter) and the private key file must match each other. Default value: *(not specified)*
`LoadBalanceSslCa` *{path}*	Path to the directory or file with the list of trusted root certificates. Among these certificates, there must be a certificate that certifies the authenticity of the certificates used by agents within the scanning cluster when exchanging data via SSL/TLS protocols. If the parameter value is empty, Dr.Web Network Checker operating on the current host does not authenticate certificates of interacting agents; however, depending on the settings, these agents can verify authenticity of the certificate used by the agent operating on this host. Default value: *(not specified)*
`LoadBalanceServerSocket` *{address}*	Network socket (an IP address and a port) listened by Dr.Web Network Checker on the current host for receiving files sent by remote hosts for scanning (in case of operating as a network scanning server). Default value: *(not specified)*
`LoadBalanceAllowFrom` *{IP address}*	IP address of a remote network host from which the Dr.Web Network Checker operating on the current host can receive files for scanning (as a network scanning server). Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list). Example: Add host addresses 192.168.0.1 and 10.20.30.45 to the list. 1. Adding values to the configuration file.

Parameter	Description

	• Two values per line:
	`[NetCheck]`
	`LoadBalanceAllowFrom = "192.168.0.1", "10.20.30.45"`
	• Two lines (one value per line):
	`[NetCheck]`
	`LoadBalanceAllowFrom = 192.168.0.1`
	`LoadBalanceAllowFrom = 10.20.30.45`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset NetCheck.LoadBalanceAllowFrom -a 192.168.0.1`
	`# drweb-ctl cfset NetCheck.LoadBalanceAllowFrom -a 10.20.30.45`
	If the parameter is empty, remote files are not accepted for scanning (the host does not operate as a scanning server).
	Default value: *(not specified)*

`LoadBalanceSourceAddress` *{IP address}*	IP address of a network interface used by Dr.Web Network Checker on the current host to transfer files for remote scanning (if the host operates as a network scanning client and has several network interfaces). If an empty value is specified, the network interface is automatically selected by the OS kernel. Default value: *(not specified)*

`LoadBalanceTo`
{address}
	Socket (an IP address and a port) of a remote host to which Dr.Web Network Checker operating on the current host can send files for remote scanning (as a network scanning client).
	Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list).
	Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list.
	1. Adding values to the configuration file.
	• Two values per line:
	`[NetCheck]`
	`LoadBalanceTo = "192.168.0.1:1234", "10.20.30.45:5678"`

Parameter	Description

	• Two lines (one value per line):
	`[NetCheck]`
	`LoadBalanceTo = 192.168.0.1:1234`
	`LoadBalanceTo = 10.20.30.45:5678`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset NetCheck.LoadBalanceTo -a 192.168.0.1:1234`
	`# drweb-ctl cfset NetCheck.LoadBalanceTo -a 10.20.30.45:5678`
	If the parameter value is empty, local files cannot be transferred for a remote scanning (the host does not operate as a network scanning client).
	Default value: *(not specified)*

`LoadBalanceStatusInterval` *{time interval}*	Time interval the current host waits to inform all distributed scanning agents specified in the `LoadBalanceAllowFrom` parameter about its workload. Default value: `1s`
`SpoolDir` *{path to directory}*	Local file system directory used to store files received from clients by Dr.Web Network Checker over the network for scanning. Default value: `/tmp/com.drweb.ncheck`
`LocalScanPreference` *{fractional number}*	Relative weight (priority) of the host upon selecting a server to scan a file (a local file or a file received over the network). If at some instant a relative weight of the local host is greater than the total weight of all hosts available as scanning servers, the file is kept by the agent for local scanning. Minimal value: `1`. Default value: `1`
`LoadBalanceSslCrl` *{path}*	Path to the directory or file with a list of revoked certificates. If a parameter value is not specified, Dr.Web Network Checker running on the current host does not verify certificates of interacting agents; however, depending on the settings, they may verify relevance of the certificate used by the agent running on the current host. Default value: *(not specified)*

9.9.4. Creating a Scanning Cluster

In this section

- Introductory remarks
- How to create a scanning cluster
- Configuring cluster hosts
- Verifying cluster operability

Introductory remarks

To create a scanning cluster that allows performing distributed scans (while scanning files or other objects), you need to have a set of network hosts with the Dr.Web Network Checker component installed on each host. To ensure that a cluster host not only receives and transmits data to be scanned, Dr.Web Scanning Engine must be installed on the host. Thus, to create a scanning cluster host, install at least the following components on the server (other components of Dr.Web Gateway Security Suite that are installed automatically to ensure operability of the components listed here are not mentioned):

1. Dr.Web Network Checker (the `drweb-netcheck` package) is a component for receiving and sending data between hosts over the network.
2. Dr.Web Scanning Engine (the `drweb-se` package) is an engine for scanning data received over the network. The component may be absent; in this case, the host only transmits data to be scanned to other hosts of the scanning cluster.

The hosts that constitute the scanning cluster form a *peer to peer* network, i.e. each of the hosts, depending on the settings defined for the Dr.Web Network Checker component on this host, is capable of acting either as a *scanning client* (which transmits data for scanning to other hosts) or as a *scanning server* (which receives data for scanning from other hosts). With the appropriate settings, a cluster host can be both a scanning client and a scanning server at the same time.

Dr.Web Network Checker parameters related to scanning cluster configuration have names starting with `LoadBalance`.

How to create a scanning cluster

Let us consider the scanning cluster example displayed in the figure below.

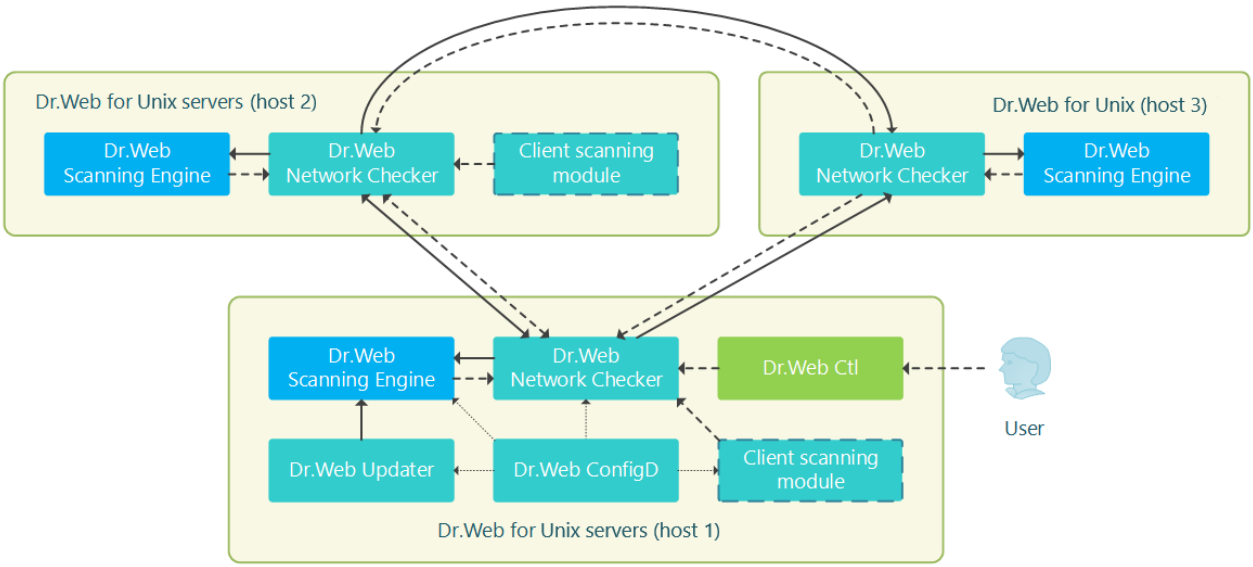


Figure 13. Scanning cluster structure

In this case, it is assumed that the cluster consists of three hosts (displayed in the figure as *host 1*, *host 2* and *host 3*). Host 1 and host 2 are servers with a fully-fledged Dr.Web product for Unix servers installed (for example, Dr.Web Gateway Security Suite or Dr.Web Mail Security Suite, the product type does not matter), and host 3 only assists in scanning files transferred from hosts 1 and 2. Therefore, only the minimal required component set is installed (Dr.Web Network Checker and Dr.Web Scanning Engine, other components that are automatically installed to ensure host operability, such as Dr.Web ConfigD, are not displayed in the figure). Hosts 1 and 2 can operate both as servers and scanning clients with regard to each other (perform mutual distribution of scanning load), and host 3—only as a server receiving tasks from hosts 1 and 2.

In the scheme, "Client scanning module" designates components of the servers that create scanning tasks for the Dr.Web Network Checker component. These tasks will be distributed (depending on the load balance) between local Dr.Web Scanning Engine and cluster partner hosts acting as scanning servers.

Only components that scan data not represented as a file in the local file system can act as the client scanning module. This means that the scanning cluster cannot be used for distributed scanning of files by the SpIDer Guard file system monitor and the Dr.Web File Checker component.

Configuring cluster hosts

To set up the specified cluster configuration, adjust Dr.Web Network Checker settings on all three cluster hosts. All of the following settings are provided in the `.ini` format (refer to configuration file format description).

Host 1

```
[NetCheck]
InternalOnly = No
LoadBalanceUseSsl = No
LoadBalanceServerSocket = <Host 1 IP address>:<Host 1 port>
LoadBalanceAllowFrom = <Host 2 IP address>
LoadBalanceSourceAddress = <Host 1 IP address>
LoadBalanceTo = <Host 2 IP address>:<Host 2 port>
LoadBalanceTo = <Host 3 IP address>:<Host 3 port>
```

Host 2

```
[NetCheck]
InternalOnly = No
LoadBalanceUseSsl = No
LoadBalanceServerSocket = <Host 2 IP address>:<Host 2 port>
LoadBalanceAllowFrom = <Host 1 IP address>
LoadBalanceSourceAddress = <Host 2 IP address>
LoadBalanceTo = <Host 1 IP address>:<Host 1 port>
LoadBalanceTo = <Host 3 IP address>:<Host 3 port>
```

Host 3

```
[NetCheck]
InternalOnly=No
LoadBalanceUseSsl = No
LoadBalanceServerSocket = <Host 3 IP address>:<Host 3 port>
LoadBalanceAllowFrom = <Host 1 IP address>
LoadBalanceAllowFrom = <Host 2 IP address>
```

Notes:

- Other Dr.Web Network Checker parameters (not mentioned here) are left unchanged.
- Change IP addresses and port numbers to relevant ones.
- This example does not use SSL for data exchange between hosts. To use SSL, set the `LoadBalanceUseSsl` to `Yes`, as well as set relevant values for parameters `LoadBalanceSslCertificate`, `LoadBalanceSslKey` and `LoadBalanceSslCa`.

Verifying cluster operability

To verify cluster operability in data distribution mode, run the command on hosts 1 and 2:

```
$ drweb-ctl netscan <path to file or directory>
```

While this command is being run, Dr.Web Network Checker scans files from the specified directory having distributed load between cluster hosts. To view statistics on network scanning for each host, output statistics of Dr.Web Network Checker by running the command before scanning:

```
$ drweb-ctl stat -n
```

To interrupt outputting statistics, press CTRL+C or Q.

9.10. Dr.Web Scanning Engine

Dr.Web Scanning Engine is designed to search for viruses and other malicious objects in files and boot records (*MBR—Master Boot Record*, *VBR—Volume Boot Record*) of disk devices. The component loads Dr.Web Virus-Finding Engine into memory and starts it as well as loads Dr.Web virus databases used by the engine to detect threats.

The scanning engine operates in daemon mode, as a service that receives requests for scanning file system objects from other Dr.Web Gateway Security Suite components (these are Dr.Web File Checker and Dr.Web Network Checker and, potentially, Dr.Web MeshD). If Dr.Web Scanning Engine and Dr.Web Virus-Finding Engine are absent or unavailable, no antivirus scanning is performed on this host (except when Dr.Web Gateway Security Suite contains the Dr.Web MeshD component, whose settings define a connection to local cloud hosts providing scanning engine services).

9.10.1. Operating Principles

The Dr.Web Scanning Engine component operating in daemon mode receives requests from other Dr.Web Gateway Security Suite components to scan file system objects (files and boot records) for embedded threats, queues scanning tasks and scans requested objects with Dr.Web Virus-Finding Engine. If a threat is detected in a scanned object that must be cured according to the scanning task, the scanning engine attempts to cure it if this action is applicable.

The scanning engine, Dr.Web Virus-Finding Engine and virus databases form a single entity and cannot be separated. The scanning engine downloads the virus databases and provides an operation environment for cross-platform Dr.Web Virus-Finding Engine. The virus databases and the scanning engine are updated by the Dr.Web Updater component included in Dr.Web Gateway Security Suite but not being a part of the scanning engine. The update component is run by the Dr.Web ConfigD configuration management daemon periodically or forcibly in response to a user command. In addition, if Dr.Web Gateway Security Suite operates in centralized protection mode, the virus databases and the scanning engine are updated by Dr.Web ES Agent, which interacts with a centralized protection server and receives updates from it.

Dr.Web Scanning Engine can be controlled by the Dr.Web ConfigD configuration management daemon or operate in standalone mode. In the former case, the daemon starts the engine and ensures that the virus databases are up to date. In the latter case, the engine is started and the virus databases are updated by an external application that uses the engine. Both the Dr.Web Gateway Security Suite components that make requests to the scanning engine for scanning files and external applications use the same API.

You can create your own component (an external application) using Dr.Web Scanning Engine for scanning files. For this purpose, Dr.Web Scanning Engine provides a custom API based on the Google Protobuf technology. To obtain the Dr.Web Scanning Engine API guide and examples of client application code using Dr.Web Scanning Engine, contact the partner relations department of the Doctor Web company.

Received scanning tasks are automatically distributed in queues with different priorities (high, normal and low). Selection of a queue for a task depends on the component that created the task. For example, tasks received from file system monitors are placed in a high-priority queue, because response time is important while monitoring file system objects. The scanning engine collects statistics on its usage, including the number of all tasks received for scanning and queue lengths. As an average load rate, the scanning engine uses an average length of queues per second. This rate is averaged for the last minute, last 5 minutes and last 15 minutes.

Dr.Web Virus-Finding Engine supports a signature analysis (signature-based detection of threats covered by virus databases) and other methods of heuristic and behavioral analyses designed for detection of potentially dangerous objects based on machine instructions and other attributes of executable code.

Heuristic analyzer cannot guarantee highly reliable results and may cause the following errors:

- *Errors of the first type*. These errors occur when a safe object is detected as malicious (false positive detections).
- *Errors of the second type*. These errors occur when a malicious object is detected as safe.

Thus, objects detected by the heuristic analyzer are treated as *Suspicious*.

It is recommended that you quarantine suspicious objects. After virus databases are updated, such files can be scanned using the signature analysis. Keep the virus databases up to date in order to avoid errors of the second type.

Dr.Web Virus-Finding Engine allows scanning both unpacked and packed files and objects inside various containers, such as archives, email messages and so on.

9.10.2. Command-Line Arguments

To start Dr.Web Scanning Engine from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-se <socket> [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-se <socket> [<parameters>]
```

where the mandatory <*socket*> argument indicates an address of a socket used by Dr.Web Scanning Engine for processing requests of client components. It can be set only as a file path (a Unix socket).

Dr.Web Scanning Engine accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None
Startup options (identical to configuration file parameters and substitute them when necessary):	
`--CoreEnginePath`	Function: Path to the library of Dr.Web Virus-Finding Engine. Short form: None. Arguments: <*path to file*>—full path to the library in use.
`--VirusBaseDir`	Function: Path to a directory with virus database files. Short form: None. Arguments: <*path to directory*>—full path to the virus database directory.
`--TempDir`	Function: Path to a directory with temporary files. Short form: None. Arguments: <*path to directory*>—full path to the directory with temporary files.
`--KeyPath`	Function: Path to the key file in use. Short form: None. Arguments: <*path to file*>—full path to the key file.
`--MaxForks`	Function: Maximum allowed number of child processes that can be spawned by Dr.Web Scanning Engine during scanning. Short form: None. Arguments: <*number*>—maximum allowed number of child processes.
`--WatchdogInterval`	Description: Frequency with which Dr.Web Scanning Engine checks whether child processes are operable and ends those that stopped responding. Short form: None. Arguments: <*time interval*>—frequency of checking child processes.
`--AbortForkOnTimeout`	Function: Terminate scan processes by sending them SIGABRT on timeout.

	Short form: None. Arguments: None
`--ShellTrace`	Function: Shell tracing (log detailed information about file scanning performed by Dr.Web Virus-Finding Engine). Short form: None. Arguments: None
`--LogLevel`	Description: Logging level of Dr.Web Scanning Engine during its operation. Short form: None. Arguments: <*logging level*>. Allowed values: • `DEBUG`—the most detailed logging level. All messages and debug information are logged. • `INFO`—all messages are logged. • `NOTICE`—all error messages, warnings and notifications are logged. • `WARNING`—all error messages and warnings are logged. • `ERROR`—only error messages are logged.

`--Log`	Description: Logging method of the component.
	Short form: None.
	Arguments: <*log type*>. Allowed values:
	• `Stderr[:ShowTimestamp]`—output messages to the standard error stream *stderr*.
	The `ShowTimestamp` option is used to add a timestamp to every message.
	• `Syslog[:`<*facility*>`]`—pass messages to the `syslog` system logging service.
	The additional <*facility*> option is used to specify a type of a log to store messages logged by `syslog`. Allowed values:
	o `DAEMON`—messages from daemons;
	o `USER`—messages from user processes;
	o `MAIL`—messages from mailers;
	o `LOCAL0`—messages from local processes 0;
	...
	o `LOCAL7`—messages from local processes 7.
	• <*path*>—path to a file for storing log messages.
	Examples:
	`--Log /var/opt/drweb.com/log/se.log`
	`--Log Stderr:ShowTimestamp`
	`--Log Syslog:DAEMON`

`--ForkMemoryLimit`	Description: Maximum amount of operating memory allocated for operation of a scanning engine instance. Short form: None. Arguments: <*size in bytes*>—maximum amount of operating memory.

	Allowed values: From 1073741824 bytes (1 Gb) to 4294967296 bytes (4 Gb). Default value: `4294967296` (4 Gb) ⚠ This parameter has no effect in 32-bit distributions of the product.
`--EngineDallocLimit`	Description: Maximum amount of operating memory allocated for storing temporary files. Short form: None. Arguments: <*size in bytes*>—maximum amount of operating memory. Allowed values: From 52428800 bytes (50 Mb) to 1073741824 bytes (1 Gb). Default value: `524288000` (500 Mb) ⚠ It is strongly recommended to change the default value of this parameter only if you are sure that this is necessary.

Example:

```
$ /opt/drweb.com/bin/drweb-se /tmp/drweb.ipc/.se --MaxForks=5
```

This command starts an instance of Dr.Web Scanning Engine and instructs it to create the `/tmp/drweb.ipc/.se` Unix socket to interact with client components and limit the number of child scanning processes to 5.

Startup notes

If necessary, any number of instances of Dr.Web Scanning Engine can be started, which provide the scanning service for client applications (not only for Dr.Web Gateway Security Suite components). At that, if a value of the `FixedSocketPath` parameter is specified in the component settings, one instance of the scanning engine is always run by the Dr.Web ConfigD configuration management daemon and available to the clients via the Unix socket. Instances of the scanning engine started directly from the command line will operate in standalone mode without establishing connection to the configuration management daemon, even if it is running. To manage the operation of the component, as well as to scan files upon request, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To scan an arbitrary file or directory with Dr.Web Scanning Engine, use the `rawscan` command of the Dr.Web Ctl tool:

```
$ drweb-ctl rawscan <path to file or directory>
```

To get documentation on this component from the command line, run the `man 1 drweb-se` command.

9.10.3. Configuration Parameters

The component uses configuration parameters specified in the `[ScanEngine]` section of the unified configuration file of Dr.Web Gateway Security Suite.

This section stores the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-se` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-se`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. If the `FixedSocketPath` parameter is set, this setting is ignored (the component does not finish its operation after the time interval expires). Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `1h`

Parameter	Description

`FixedSocketPath`
 	
{path to file}
	Path to the Unix socket of the fixed instance of Dr.Web Scanning Engine.
	
	If this parameter is specified, the Dr.Web ConfigD configuration management daemon ensures that there is always a running scanning engine instance available to clients via this socket.
	
	⚠ Ensure that the following permissions are assigned to the directory with the socket file:
	`drwxr-xr-x`
	To view the permissions, run the command:
	`$ ls -ld <path to directory>`
	
	Default value: *(not specified)*

`MaxForks` *{integer}*	Maximum allowed number of child scanning processes that can run simultaneously spawned by Dr.Web Scanning Engine. Default value: *Automatically determined at startup* as twice the number of available CPU cores; or `4`, if the resulting number is less than 4.
`ForkMemoryLimit` *{integer}*	Maximum amount of operating memory allocated for operation of a scanning engine instance. The value is specified as an integer with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Allowed values: from 1 gigabyte (`1GB`) to 4 gigabytes (`4GB`). Default value: `4GB` ⚠ This parameter has no effect in 32-bit distributions of Dr.Web Gateway Security Suite.
`EngineDallocLimit` *{integer}*	Maximum amount of operating memory allocated for storing temporary files. If the specified value is less than the size of a temporary file, then it will be created on the storage device, which will slow down the operation of Dr.Web Gateway Security Suite. The value is specified as an integer with a suffix (*b, kb, mb, gb*). If no suffix is specified, the value is treated as a size in bytes. Allowed values: from 50 megabytes (`50MB`) to 1 gigabyte (`1GB`). Default value: `500MB` ⚠ It is strongly recommended to change the default value of this parameter only if you are sure that this is necessary.

Parameter	Description
`WatchdogInterval` *{time interval}*	Rate at which Dr.Web Scanning Engine checks whether child scanning processes spawned by it are operable to detect deadlocks ("watchdog timer"). Default value: `15s`
`BufferedIo` *{boolean}*	Enable or disable buffered input/output while scanning files. Using buffered input/output on OSes of the GNU/Linux family and on FreeBSD can increase speed of scanning files on slow disks. Allowed values: • `On`, `Yes`, `True`—enable buffered input/output; • `Off`, `No`, `False`—disable buffered input/output. Default value: `Off`
`AbortForkOnTimeout` *{boolean}*	Terminate or do not terminate scan processes by sending them SIGABRT on timeout. Allowed values: • `On`, `Yes`, `True`—terminate scan processes on timeout; • `Off`, `No`, `False`—do not terminate scan processes on timeout. Default value: `Off`

9.11. Dr.Web Updater

The Dr.Web Updater component is designed for receiving all available updates for virus databases and Dr.Web Virus-Finding Engine from Doctor Web update servers.

If Dr.Web Gateway Security Suite operates in centralized protection mode, updates are received from a centralized protection server; at that, all updates are received from the server via Dr.Web ES Agent, and Dr.Web Updater is not used for downloading updates.

9.11.1. Operating Principles

The component is designed to establish connections to Doctor Web update servers to check for updates for virus databases and Dr.Web Virus-Finding Engine, for the database of web resource categories. The lists of the servers that constitute an available update zone are stored in a special file signed to prevent its modification. Only basic and digest authentication are supported for connection to the update servers using a proxy server.

If Dr.Web Gateway Security Suite is not connected to a centralized protection server or is connected to it in mobile mode, Dr.Web Updater is automatically started by the Dr.Web ConfigD configuration management daemon at intervals specified in the settings. The component can also be started by Dr.Web ConfigD upon receiving a corresponding command from the user (unscheduled update).

When updates become available on update servers, they are downloaded to the directory `/var/opt/drweb.com/cache/` for GNU/Linux or `/var/drweb.com/cache/` for FreeBSD; after that they are moved to the working directories of Dr.Web Gateway Security Suite.

By default, all updates are downloaded from the update zone that is common for all Dr.Web products. The list of the servers used by default and included in the update zone is specified in the files located in directories defined by `*DrlDir` parameters grouped by an update type: for virus databases and the scanning engine, for the database of web resource categories). Upon a client request a custom update zone can be created (for each update type), the server list of which is specified in a separate file (named `update.drl` by default). This file is located in the directory specified by the corresponding `*CustomDrlDir` parameter. In this case, the update component will download updates only from these servers without using the servers from the default zone.

If you do not want to use the custom update zone, clear the value of the corresponding `*CustomDrlDir` parameter in the component settings.



The content of the files with server lists is signed so that the files cannot be modified. If you need to create a custom list of update servers, contact our technical support.

The component can back up the files to be updated to further roll back updates at user request. You can specify a backup location and an update history depth in the component settings. To roll back updates, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line (run the tool using the `drweb-ctl` command).

9.11.2. Command-Line Arguments

To start the Dr.Web Updater component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-update [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-update [<parameters>]
```

Dr.Web Updater accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-update --help
```

This command outputs short help information about the Dr.Web Updater component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration daemon when necessary. To manage the operation parameters of the component, as well as to update virus databases and the scan engine, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-update` command.

9.11.3. Configuration Parameters

The component uses configuration parameters specified in the `[Update]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-update` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-update`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. The component is started upon the next update on schedule. When the update is completed, it is waiting for the specified time interval, and, if there are no new requests, it shuts down until the next update attempt.

Parameter	Description
	Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the `SIGTERM` signal will not be sent if the component goes idle. Default value: `10m`
`Start` *{boolean}*	Enable or disable the component at the startup of Dr.Web Gateway Security Suite. This parameter has priority over the `DwsUpdateEnabled` parameter. Allowed values: • `Yes`—enable the component at the startup of Dr.Web Gateway Security Suite; • `No`—disable the component at the startup of Dr.Web Gateway Security Suite. Default value: `Yes`
`UpdateInterval` *{time interval}*	The frequency to check for updates on Dr.Web update servers. This is a time period between a previous successful attempt to connect to the update servers (initiated automatically or manually) and the next attempt to perform an update. Default value: `30m`
`NetworkTimeout` *{time interval}*	A time-out period imposed on the network-related operations of the component while downloading updates from Dr.Web servers. This parameter is used when a connection is temporarily lost: if the connection is established again before the time-out expires, the interrupted updating process will be continued. Specifying the time-out value greater than `75s` has no effect as the connection is closed by the TCP timeout. Minimal value: `5s`. Default value: `60s`
`RetryInterval` *{time interval}*	Frequency of repeated attempts to perform an update using the update servers if the previous attempt failed. Allowed values: from 1 minute (`1m`) to 30 minutes (`30m`). Default value: `3m`
`MaxRetries` *{integer}*	Number of repeated attempts to perform an update using Dr.Web update servers (the frequency is specified in the `RetryInterval` parameter) if the previous attempt failed.

Parameter	Description
	If the value is set to `0`, repeated attempts are not made (the next update will be performed after the time period specified in the `UpdateInterval` parameter). Default value: `3`
`UseHttps` *{Always | ResListOnly | Never}*	Use or do not use HTTPS while downloading updates. Allowed values: • `Always`—always use HTTPS while downloading updates. • `ResListOnly`—use HTTPS only while downloading an `.lst` file containing a list of update files. At that, update files will be downloaded via HTTP. • `Never`—always use HTTP while downloading updates. Default value: `ResListOnly`
`Proxy` *{connection string}*	Parameters for connecting to a proxy server that is used by the Dr.Web Updater component when it is connecting to Dr.Web update servers (for example, if connecting directly to external servers is prohibited by network security policies). If a parameter value is not specified, the proxy server is not used. Allowed values: <*connection string*>—proxy server connection string. The string has the following format (URL): `[`<*protocol*>`://][`<*user*>`:`<*password*>`@]`<*host*>`:`<*port*> where: • <*protocol*> is a type of the protocol in use (in the current version, only `http` is available); • <*user*> is a user name for connecting to the proxy server; • <*password*> is a password for connecting to the proxy server; • <*host*> is the host address of the proxy server (an IP address or a domain name, i.e. FQDN); • <*port*> is a port to be used. The URL parts <*protocol*> and <*user*>:<*password*> may be absent. The proxy server address <*host*>:<*port*> is mandatory. If the user name or the password contains characters `@`, `%` or `:`, these characters must be replaced with the corresponding HEX codes: `%40`, `%25` and `%3A` respectively.

Parameter	Description

	Examples:
	1. In the configuration file:
	• Connection to a proxy server hosted at `proxyhost.company.org` using port `123`:
	`Proxy = proxyhost.company.org:123`
	• Connection to a proxy server hosted at `10.26.127.0` using port `3336` via HTTP protocol as the `legaluser` user with the `passw` password:
	`Proxy = http://legaluser:passw@10.26.127.0:3336`
	• Connection to the proxy server hosted at `10.26.127.0` using port `3336` as the `user@company.com` user with the `passw%123` password:
	`Proxy = user%40company.com:passw%25123%3A@10.26.127.0:3336`
	2. Setting the same values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset Update.Proxy proxyhost.company.org:123`
	`# drweb-ctl cfset Update.Proxy http://legaluser:passw@10.26.127.0:3336`
	`# drweb-ctl cfset Update.Proxy user%40company.com:passw%25123%3A@10.26.127.0:3336`
	Default value: *(not specified)*

`ExcludedFiles`
{file name}
	Name of a file that will not be updated by the Dr.Web Updater component.
	Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list).
	Example: Add the `123.vdb` and `456.dws` files to the list.
	1. Adding values to the configuration file.
	• Two values per line:
	`[Update]`
	`ExcludedFiles = "123.vdb", "456.dws"`
	• Two lines (one value per line):
	`[Update]`
	`ExcludedFiles = 123.vdb`
	`ExcludedFiles = 456.dws`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset Update.ExcludedFiles -a 123.vdb`
	`# drweb-ctl cfset Update.ExcludedFiles -a 456.dws`
	Default value: `drweb32.lst`

Parameter	Description
`BaseUpdateEnabled` *{boolean}*	Allow or do not allow updating the virus databases and the scan engine. Allowed values: • `Yes`—updating is allowed and will be performed; • `No`—updating is not allowed and will not be performed. Default value: `Yes`
`BaseDrlDir` *{path to directory}*	Path to a directory that contains files used for connection to servers of a standard update zone, which are used to update virus databases and the scan engine. Default value: • for GNU/Linux: `/var/opt/drweb.com/drl/bases` • for FreeBSD: `/var/drweb.com/drl/bases`
`BaseCustomDrlDir` *{path to directory}*	Path to a directory that contains files used for connection to a special (“customized”) update zone, which are used to update virus databases and the scan engine. If the directory defined in this parameter contains a non-empty signed server list file (the `.drl` file), the update is performed only from these servers, and the main zone servers (see above) are not used to update the virus databases and the scanning engine. Default value: • for GNU/Linux: `/var/opt/drweb.com/custom-drl/bases` • for FreeBSD: `/var/drweb.com/custom-drl/bases`
`VersionUpdateEnabled` *{boolean}*	Allow or do not allow updating versions of Dr.Web Gateway Security Suite components. Allowed values: • `Yes`—updating is allowed and will be performed; • `No`—updating is not allowed and will not be performed. Default value: `Yes`
`VersionDrlDir` *{path to directory}*	Path to a directory that contains files for connection to servers that are used for updating versions of Dr.Web Gateway Security Suite components. Default value: • for GNU/Linux: `/var/opt/drweb.com/drl/version` • for FreeBSD: `/var/drweb.com/drl/version`

Parameter	Description
`DwsUpdateEnabled` *{boolean}*	Allow or do not allow updating of the database of web resource categories. Allowed values: • `Yes`—updating is allowed and will be performed; • `No`—updating is not allowed and will not be performed. Default value: `Yes`
`DwsDrlDir` *{path to directory}*	Path to a directory that contains the files for connecting to servers of a standard update zone, which are used for updating the database of web resource categories. Default value: • for GNU/Linux: `/var/opt/drweb.com/drl/dws` • for FreeBSD: `/var/drweb.com/drl/dws`
`DwsCustomDrlDir` *{path to directory}*	Path to a directory that contains the files for connecting to servers of a special (“customer”) update zone, which are used for updating the database of web resource categories. If the directory defined in this parameter contains a non-empty signed server list file (the `.drl` file), updating is performed only from these servers, and the main zone servers (see above) are not used to update the database of web resource categories. Default value: • for GNU/Linux: `/var/opt/drweb.com/custom-drl/dws` • for FreeBSD: `/var/drweb.com/custom-drl/dws`
`BackupDir` *{path to directory}*	Path to a directory where old versions of updated files are saved for possible rollback. Only updated files are backed up upon every update. Default value: • for GNU/Linux: `/var/opt/drweb.com/backup` • for FreeBSD: `/var/drweb.com/backup`
`MaxBackups` *{integer}*	The maximum number of the previous versions of updated files, which are saved. If this number is exceeded, the oldest copy is removed upon the next update. If the parameter value is `0`, the previous versions of backup files are not stored. Default value: `0`

9.12. Dr.Web ES Agent

The centralized protection agent Dr.Web ES Agent is designed for connecting Dr.Web Gateway Security Suite to a centralized protection server.

When Dr.Web Gateway Security Suite is connected to the centralized protection server, Dr.Web ES Agent synchronizes the license key file with the key files stored on the centralized protection server. In addition, Dr.Web ES Agent sends statistics on virus events, the list of running components and their status to the centralized protection server.

Furthermore, Dr.Web ES Agent updates Dr.Web Gateway Security Suite virus databases directly from the connected centralized protection server without using the Dr.Web Updater component.

9.12.1. Operating Principles

The Dr.Web ES Agent component connects to a centralized protection server, which allows a network administrator to implement a common security policy within the entire network, in particular, to configure the same scanning settings and reaction on threat detection for all workstations and servers of the network. In addition, the centralized protection server also acts as an internal update server on the protected network and stores up-to-date virus databases (in this case, updating is performed via Dr.Web ES Agent, Dr.Web Updater is not used).

When Dr.Web ES Agent connects to the centralized protection server, the agent receives up-to-date settings of the program components and the license key file from the server, which are then passed to the Dr.Web ConfigD configuration management daemon for applying them to the managed components. In addition, the component can also receive tasks from the centralized protection server for scanning file system objects on the station (including on schedule).

Dr.Web ES Agent collects statistics on detected threats and applied actions and sends it to the server to which the agent is connected.

To connect Dr.Web ES Agent to the centralized protection server, the password and identifier of the host (“station” in terms of the centralized protection server) are required as well as a public encryption key file, which is used by the server for authentication. Instead of the station identifier, while connecting, you can specify the identifier of the main and tariff groups in which the station is to be included. For the required identifiers and public key file, contact the administrator of your anti-virus network.

In addition, you can connect a protected host (“station”) to the centralized protection server as a “newbie”, if this is allowed by the centralized protection server. In this case, after the administrator confirms the request to connect the station, the centralized protection server automatically generates new identifier and password for the host and sends them to the agent for future connections.

9.12.2. Command-Line Arguments

To start the Dr.Web ES Agent component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-esagent [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-esagent [<parameters>]
```

Dr.Web ES Agent accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-esagent --help
```

This command outputs short help information about the Dr.Web ES Agent component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration daemon at the startup of the operating system. To manage the operation parameters of the component as well as to connect Dr.Web Gateway Security Suite to a centralized protection server, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-esagent` command.

9.12.3. Configuration Parameters

The component uses configuration parameters specified in the `[ESAgent]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-esagent` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-esagent`
`MobileMode` *{On | Off | Auto}*	Enable or disable the mobile mode of Dr.Web Gateway Security Suite while connected to a centralized protection server. Allowed values: • `On`—enable the mobile mode if it is allowed by the centralized protection server (install updates from the update servers of Doctor Web using Dr.Web Updater); • `Off`—disable the mobile mode and continue operation in centralized protection mode (always receive updates from the centralized protection server); • `Auto`—enable the mobile mode if allowed by the centralized protection server and install updates both from the update servers of Doctor Web using Dr.Web Updater and from the centralized protection server depending on which connection is available and which connection quality is higher. Behavior of this parameter depends on server permissions: if the mobile mode is not allowed on the server in use, this parameter has no effect. Default value: `Auto`
`Discovery` *{On | Off}*	Allow or do not allow the agent to receive *discovery* requests from a network inspector built in the centralized protection server (*discovery* requests are

Parameter	Description
	used by the inspector to check the structure and state of the anti-virus network). Allowed values: • `On`—allow the agent to receive and process *discovery* requests; • `Off`—do not allow the agent to receive and process *discovery* requests.  This parameter has priority over the settings of the centralized protection server: if the parameter value is set to `Off`, the agent does not receive discovery requests even if this feature is enabled on the server. Default value: `On`
`UpdatePlatform` *{platform name}*	Designation of a platform for which the agent will receive updates for the scanning engine from the centralized protection server. Allowed values: • for GNU/Linux: `unix-linux-32`, `unix-linux-64-engine64`, `unix-linux-aarch64`, `unix-linux-e2k`, `unix-linux-e2k-engine64`, `unix-linux-mips`, `unix-linux-ppc64le`; • for FreeBSD: `unix-freebsd-32`, `unix-freebsd-64-engine64`; • for Darwin: `unix-darwin-32`, `unix-darwin-64-engine64`, `unix-darwin-aarch64`.  It is strongly recommended to change the parameter value only if you are sure that this is necessary. Default value: *Depends on the platform in use*
`DebugIpc` *{boolean}*	Log or do not log IPC messages at the debug level (with `LogLevel = DEBUG`) (interaction of Dr.Web ES Agent with the Dr.Web ConfigD configuration management daemon). Default value: `No`
`DebugConfig` *{boolean}*	Log or do not log information about settings received from the centralized protection server. Default value: `No`
`SrvMsgAutoremove` *{integer}*	Storage period after which the messages from the centralized protection server are removed automatically. Allowed values: from 1 week (`1w`) to 365 days (`365d`). The storage period is specified as an integer with a suffix (`s`, `m`, `h`, `d`, `w`). Default value: `1w`

9.13. Dr.Web HTTPD

The Dr.Web HTTPD component provides infrastructure for local and remote interaction with Dr.Web Gateway Security Suite via HTTP (for example, using a web browser).

In addition to managing Dr.Web Gateway Security Suite via the Dr.Web web interface, you can also use a command-line interface (HTTP API) of Dr.Web HTTPD directly to interact with Dr.Web Gateway Security Suite components via HTTPS. This capability allows you to create your own interface to manage Dr.Web Gateway Security Suite.

The HTTP API of Dr.Web HTTPD is described in the corresponding section.

To use a secure HTTPS connection, it is required to provide a valid SSL server certificate and private key for Dr.Web HTTPD. By default, a server certificate and an SSL private key are generated for Dr.Web HTTPD automatically during the installation procedure, but, if necessary, you can generate your own certificate and key. In addition, a user personal authorization certificate signed by a certificate issued by a certificate authority and trusted by Dr.Web HTTPD can be used for automatic client authorization when connecting to Dr.Web HTTPD.

To generate keys and certificates, you can use the `openssl` utility. Examples of using the `openssl` utility to generate certificates and private keys are provided in the Appendix E. Generating SSL Certificates section.

9.13.1. Operating Principles

Dr.Web HTTPD is a web server designed for managing Dr.Web Gateway Security Suite without a need for either third-party servers (for example, Apache HTTP Server and Nginx) or management services, such as Webmin. Furthermore, the component can function simultaneously with them on the same host without impeding their operation.

The Dr.Web HTTPD server processes requests received via HTTP and HTTPS to sockets specified in the settings. For this reason, the server does not have conflicts with other web servers used on the same host. Dr.Web Gateway Security Suite is managed via the secure protocol, HTTPS.



The Dr.Web management web interface is not required for the operation of Dr.Web Gateway Security Suite and can be absent; therefore, the corresponding block is circled with a dashed line in the scheme.

The Dr.Web HTTPD component issues commands to the Dr.Web ConfigD configuration management daemon, to the Dr.Web File Checker component designed for file scanning and to other components on the basis of commands received via the HTTP API.

The management web interface, which uses Dr.Web HTTPD, is described in the corresponding section.

If the Dr.Web management web interface is not included in Dr.Web Gateway Security Suite, you can connect the product to any third-party management interface that uses the HTTP API (described in the Description of the HTTP API section) of the Dr.Web HTTPD component.

9.13.2. Command-Line Arguments

To start the Dr.Web HTTPD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-httpd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-httpd [<parameters>]
```

Dr.Web HTTPD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-httpd --help
```

This command outputs short help information about the Dr.Web HTTPD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary (usually, at the startup of the operating system). If the component is running, to manage the components of Dr.Web Gateway Security Suite, you can use any standard web browser to access, via HTTPS, any of the addresses at which the web interface is served. To manage the operation of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.

To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-httpd` command.

9.13.3. Configuration Parameters

The component uses configuration parameters specified in the `[HTTPD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-httpd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-httpd`
`Start` *{boolean}*	Start or stop the component with the Dr.Web ConfigD configuration management daemon. Setting this parameter to `Yes` instructs the configuration management daemon to start the component immediately, and setting this parameter to `No`—to shut down the component immediately. Default value: *Depends on whether the management web interface is installed*
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`

Parameter	Description

`AdminListen`
 	
{address, ...}
	List of network sockets (every network socket is defined by the *<IP address>:<port>* pair) listened by Dr.Web HTTPD for HTTPS connections from clients with administrative privileges. These sockets are used both to connect to the management web interface and to access the HTTP API.
	
	The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	
	Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list.
	
	1. Adding the values to the configuration file.
	• Two values per line:
	`[HTTPD]`
	`AdminListen = "192.168.0.1:1234", "10.20.30.45:5678"`
	• Two lines (one value per line):
	`[HTTPD]`
	`AdminListen = 192.168.0.1:1234`
	`AdminListen = 10.20.30.45:5678`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset HTTPD.AdminListen -a 192.168.0.1:1234`
	`# drweb-ctl cfset HTTPD.AdminListen -a 10.20.30.45:5678`
	
	If no value is specified, it is impossible to use HTTP API and the management web interface.
	
	Default value: `127.0.0.1:4443`

`AdminSslCertificate` *{path to file}*	Path to the server certificate file used by the management web interface server to communicate with clients that establish HTTPS connections to the administrative socket. This file is generated automatically during the installation of the component. ⚠ The certificate file and the private key file (specified by the `AdminSslKey` parameter) must match each other. Default value: • for GNU/Linux: `/etc/opt/drweb.com/certs/serv.crt` • for FreeBSD: `/usr/local/etc/drweb.com/certs/serv.crt`
`AdminSslKey` *{path to file}*	Path to the private key file used by the management web interface server to communicate with clients that establish HTTPS connections to the administrative socket. The private key file is generated automatically during the installation of the component.

Parameter	Description
	 The certificate file (specified by the `AdminSslCertificate` parameter) and the private key file must match each other. Default value: • for GNU/Linux: `/etc/opt/drweb.com/certs/serv.key` • for FreeBSD: `/usr/local/etc/drweb.com/certs/serv.key`
`AdminSslCA` *{path to file}*	Path to a Certification Authority (CA) certificate file for certificates used by clients that establish HTTPS connections to the administrative socket. If a client certificate is signed with the certificate specified by this parameter, client authentication by providing the login/password pair is not performed. Furthermore, this authentication method is prohibited for clients that use client certificates signed with this certificate. A client who passed the certificate-based authentication is always assumed to be a superuser (*root*). Default value: *(not specified)*

`PublicListen`
 	
{address, ...}
	List of network sockets (every network socket is defined by the *<IP address>:<port>* pair) listened by Dr.Web HTTPD for HTTP connections from clients with limited privileges.
	
	The values of the list must be comma-separated (each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all parameter values are combined into one list).
	
	Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list.
	
	1. Adding the values to the configuration file.
	• Two values per line:
	`[HTTPD]`
	`PublicListen = "192.168.0.1:1234", "10.20.30.45:5678"`
	• Two lines (one value per line):
	`[HTTPD]`
	`PublicListen = 192.168.0.1:1234`
	`PublicListen = 10.20.30.45:5678`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset HTTPD.PublicListen -a 192.168.0.1:1234`
	`# drweb-ctl cfset HTTPD.PublicListen -a 10.20.30.45:5678`
	
	You cannot access the full scope of the HTTP API commands or access the management web interface at these addresses.
	
	Default value: *(not specified)*

Parameter	Description
`WebconsoleRoot` *{path to directory}*	Path to a directory with the files used by the management web interface (similar to the `htdocs` directory of Apache HTTP Server). Default value: • for GNU/Linux: `/opt/drweb.com/share/drweb-httpd/webconsole` • for FreeBSD: `/usr/local/libexec/drweb.com/share/drweb-httpd/webconsole`
`HelpRoot` *{path to directory}*	Path to a directory with the files of the documentation accessible via the management web interface. Default value: • for GNU/Linux: `/opt/drweb.com/share/doc/html` • for FreeBSD: `/usr/local/libexec/drweb.com/share/doc/html`
`AccessLogPath` *{path to file}*	Path to the file in which all HTTP/HTTPS requests from clients to the management web interface server are logged. If the parameter is not specified, requests are not logged. Default value: *(not specified)*

9.13.4. Description of the HTTP API

In this section

- General information
  - Format of an HTTP response to an HTTP request
  - Encoding of strings in JSON objects
  - General restrictions on transmitting data
- User authentication and authorization
  - By specifying a login and a password (SCS)
  - Authorization using a personal certificate
- Managing Dr.Web Gateway Security Suite
- Scanning objects
- Managing the list of threats
- Managing quarantine
- HTTP API usage examples

General information

The HTTP API is provided as a means to integrate Dr.Web Gateway Security Suite with third-party applications via HTTP (to ensure security, HTTPS is used).

HTTP 1.0 is used to interact with the API. All requests use standard methods of HTTP: `GET` and `POST`. All data is transferred in the form of JSON objects, except as otherwise specified. If you are sending a JSON object in the body of your HTTP POST request, set the `Content-Type:` header to have the value of `application/json`. For clarity, the examples of JSON objects provided below have comments (starting with `//` characters). The comments are not supported by the JSON format and must be deleted upon pasting into code.

Format of an HTTP response to an HTTP request

- In response to all requests, except as otherwise specified, the API always returns a JSON object. If an error occurs, the API returns an Error JSON object.
- If the JSON object sent as a response has a field of an Array type, but this array does not contain any elements, this field is omitted in the server response.
- In all responses, except as otherwise specified, the `Content-Type:` header field has the `application/json` value.
- If the client requests an endpoint that does not exist, the Error JSON object in which the `code` field with the `EC_UNEXPECTED_MESSAGE` value is returned.
- If SCS (*Secure Cookie Sessions for HTTP*) is used, the responses contain a cookie proving the client authorization (hereinafter referred to as the *SCS cookie*).

Encoding of strings in JSON objects

- The strings are passed in the UTF-8 encoding (without BOM). Characters that are not part of the ASCII table are not escaped with sequences of the form `\uXXXX`, but are passed in the UTF-8 encoding.
- JSON objects can contain both UTF-8 encoded characters and escaped sequences of the form `\uXXXX`.

General restrictions on transmitting data

- In POST requests with JSON objects in their body any characters complying with RFC 7159 are allowed.
- In GET requests any characters complying with RFC 1945 are allowed in the URI.
- Characters complying with RFC 1945 can be used in any other part of the request (either in the headers or in the body).

User authentication and authorization

To start using the API, you must be authenticated by the server. Two methods of authorization are provided:

1. Using SCS in accordance with RFC 6896.

2. Using client SSL certificates attested by a trusted CA's certificate on behalf of Dr.Web HTTPD. In this case the client is considered authorized as a superuser (X.509 client certificates are used).

If SCS is used, SCS cookies are passed in the following HTTP headers: `Cookie:`—in the request and `Set-Cookie:`—in the response.
When authorizing with SSL certificates, SCS cookies are not required.

When authorizing with SCS, using the API starts with sending the `login` command. If this command is run successfully, the client receives an SCS cookie confirming the authorization. When authorizing with a client certificate, you do not need to run the `login` command. If you try to run it, the Error JSON object is returned.

By specifying a login and a password (SCS)

User authentication and authorization commands:

API command	Description
`login`	Action: Authorize the client based on the specified user name and password to further use the HTTP API. If the authentication is successful, an SCS cookie will be returned. URI: `/api/10.2/login` HTTP method: POST Input parameters: AuthOptions object Result of successful execution: an empty object, SCS cookie
`logout`	Action: Revoke (deactivate) the SCS cookie provided earlier. The Error object with the `EC_NOT_AUTHORIZED` error code is returned in response to a request with the revoked SCS cookie. URI: `/api/10.2/logout` HTTP method: GET Input parameters: SCS cookie Result of successful execution: an empty object
`whoami`	Action: Get the name of the authorized user. URI: `/api/10.2/whoami` HTTP method: GET Input parameters: (SCS cookie)* Result of successful execution: whoami object, (SCS cookie)

*) Here and below the SCS cookie is put in parentheses, because it is required only when authorizing via SCS.

The `login` and `logout` commands are used only when authorizing via SCS.

Description of used objects

1) `AuthOptions`—object containing the user login and password:

```
{
 "user": <string>, //User name
 "password": <string> //User password
}
```

You can specify any user who is a member of the admin group (`sudoers` in Debian and Ubuntu, `wheel` in CentOS and Fedora, `astra-admin` in Astra Linux, etc). If the user is not a member of the `sudoers` group, the `EC_NOT_AUTHORIZED` error is returned in the response.

2) `whoami`—object containing the name of the user who was authorized to use the HTTP API:

```
{
 "whoami":
 {
 "user": <string>, //User name
 }
}
```

3) `Error`—object containing information about the occurred error:

```
{
 "error":
 {
 "code": <string>, //String specifying an error code of the form EC_XXX
 *"what": <string> //Error description
 }
}
```

The parameter marked with `*` is optional.

The Error JSON object, that is returned in response to HTTP API commands if an error occurs while processing the requests, has the `code` field containing an internal string-type code used by the components of Dr.Web Gateway Security Suite rather than a numeric error code. This code is a string in the form of `EC_XXX`. To find out the numeric code and get detailed information about the error, refer to the Appendix F. Known Errors section.

Authorization using a personal certificate

This authorization method implies using a personal certificate signed by a certificate authority certificate specified as trusted in the Dr.Web HTTPD settings. If you have authorized with a certicate, all your requests are considered executed as a superuser (the *root* user).

To authorize with a personal user certificate

1. Create a personal certificate signed by a certificate authority certificate.
2. In the Dr.Web HTTPD settings (the `AdminSslCA` parameter), specify a path to the certificate authority certificate used to sign your personal certificate.
3. Every time you connect to Dr.Web HTTPD, use a signed certificate.

If necessary, refer to the Appendix E. Generating SSL Certificates section.

Managing Dr.Web Gateway Security Suite

API commands for viewing and modifying the current values of configuration parameters:

API command	Description
Configuration management commands	
`get_lexmap`	Action: Get the parameter values of the current configuration (a “lexical map” of parameters). URI: `/api/10.2/get_lexmap` HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: LexMaps object, (SCS cookie)
`set_lexmap`	Action: Set or reset the specified parameter values of the current configuration (sent as a “lexical map” of parameters). URI: `/api/10.2/set_lexmap` HTTP method: POST Input parameters: (SCS cookie), LexMap object Result of successful execution: SetOptionResult object, (SCS cookie)
Updating commands	
`start_update`	Action: Start updating.

API command	Description
	URI: `/api/10.2/start_update` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: the StartUpdate object, (SCS cookie)
`stop_update`	Action: Stop an active update process. URI: `/api/10.2/stop_update` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: an empty object, (SCS cookie)
`baseinfo`	Action: View information about the loaded virus bases URI: `/api/10.2/baseinfo` HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: the BaseInfoResult object containing the VirusBaseInfo object (SCS cookie)
Licence management commands	
`install_license`	Action: Install the specified key file. URI: `/api/10.2/install_license` HTTP method: POST Input parameters: (SCS cookie), body of the key file (or of an archive with the key file) Result of successful execution: an empty object, (SCS cookie)
Commands for connection to the centralized protection server	
`esconnect`	Action: Enable the centralized protection mode. URI: `/api/10.2/esconnect` HTTP method: POST Input parameters: (SCS cookie), the ESConnection object

API command	Description
	Result of successful execution: an empty object, (SCS cookie)
`esdisconnect`	Action: Disable the centralized protection mode. URI: `/api/10.2/esdisconnect` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: an empty object, (SCS cookie)

Configuration of the product components is returned and set as the so-called *lexical map*, i.e. as a sequence of parameter-value pairs. A LexMaps object always contains three nested LexMap objects containing the following lexical maps:

- *active*—active (valid) parameter values;
- *hardcoded*—default values (automatically assigned to the parameters which values are not specified or invalid);
- *master*—map of configuration parameter values specified by the client.



The `get_lexmap` command always returns all three sets of configuration parameter values for all the components that can be included in Dr.Web Gateway Security Suite, and not only for those that are installed and running.

Description of used JSON objects

1) `LexMaps`—object containing an active, a predefined and a user-defined lexical maps of parameter values:

```
{
 "active": <LexMap>, //Active values of configuration parameters
 "hardcoded": <LexMap>, //Predefined values of configuration parameters
 "master": <LexMap> //Parameter values set by the user
}
```

   Each of these fields is a LexMap object, which in turn contains an array of LexOption objects.

2) `LexMap`—object containing a lexical map of parameters:

```
{
 "option": [<LexOption>] //Array of configuration options
}
```

3) `LexOption`—object containing a parameter or a configuration section (a group of parameters):

```
{
 "key": <string>, //Option name (a parameter name or a section name)
 *"value": <LexValue>, //If this option is a parameter
```

```
 *"map": <LexMap> //If this option is a section
}
```

The parameters marked with * are optional.

The LexOption object represents a section or a parameter of the Dr.Web Gateway Security Suite configuration. It always has a key field corresponding to the name of the section or of the parameter, as well as a value field (if the object is a parameter), or a map field (in case it describes a section). A section is also an object of the LexMap type; whereas the parameter value is an object of the LexValue type containing an item field specifying the parameter value in the string format.

4) LexValue—object containing an array of parameter values:

```
{
 "item": [<string>] //Array of parameter values
}
```

The set_lexmap command accepts a LexMap object that must contain all the parameters which values you want to change to new ones or reset to their defaults. Those parameters that you want to reset to their default values must not contain the value field. Parameters that are not mentioned in the lexical map passed to the set_lexmap command will not be changed. As a result of its execution, the set_lexmap command returns a SetOptionResult object containing changed values of every parameter passed to the command.

5) SetOptionResult—object with an item field containing an array of modified parameter values:

```
{
 "item": [<SetOptionResultItem>] //Array of results
}
```

The object contains an array of SetOptionResultItem objects containing changed values of every parameter passed to the command.

6) SetOptionResultItem—object containing information about a change of a parameter value:

```
{
 "option": <string>, //Parameter name
 "result": <string>, //Result of changing the value (an error code)
 *"lower_limit": <string>, //Lowest allowed value
 *"upper_limit": <string> //Highest allowed value
}
```

The parameters marked with * are optional.

The option field contains the name of the parameter to which your action was applied, and the result field contains the result of an attempt to change the value of this parameter. If a new value was successfully assigned, the field has the EC_OK value. In case of an error, this object may contain a lower_limit field and an upper_limit field that store the maximum and the minimum allowed value for this parameter.

7) The StartUpdate object contains data on the started update process:

```
{
 "start_update":
  {
    "attempt_id": <number> //Identifier of the started update process
```

```
  }
}
```

8) The `ESConnection` object contains data on connection to a centralized protection server:

```
{
 *"server": <string>, //<Host address>:<port>(without the http/https prefix)
 "certificate": <string>, //Base64 server key
 *"newbie": <boolean>, //Default value: false
 *"login": <string>, //User name
 *"password": <string> //Password
}
```

The parameters marked with `*` are optional.

The `login` and `password` parameters are only specified if `newbie = true`.
Before connecting, download the certificate file from the centralized protection server and run the command:

```
$ cat certificate.pem | base64
```

The string obtained from running this command is used as the `certificate` parameter value.

9) The `BaseInfoResult` object contains data on the loaded virus bases:

```
{
 "vdb_base_stamp": <number>, //Base timestamp
 "vdb_bases": [<VirusBaseInfo>] //Detailed information about the base
}
```

10) The `VirusBaseInfo` object contains the detailed information about the virus base:

```
{
  "path": <string>, //Path to the base file
  "virus_records": <number>, //Number of records in the base
  "version": <number>, //Base version
  "timestamp": <number>, //Base timestamp
  "md5": <string>, //Base MD5 hash
  "load_result": <string>, //Result of loading the base (EC_OK if the base has been
loaded successfully)
  *"sha1": <string> //Base SHA1 hash
}
```

The parameter marked with `*` is optional.

Scanning objects

API commands for scanning objects:

API command	Description
Data scanning (calling the Dr.Web Network Checker component)	
`scan_request`	Action: Request to create a connection (*endpoint*) to scan data with the necessary parameters. URI: `/api/10.2/scan_request`

API command	Description
	HTTP method: POST Input parameters: (SCS cookie), the ScanOptions object Result of successful execution: the ScanEndpoint object, (SCS cookie)
`scan_endpoint`	Action: Start data scanning (for instance, file body) at the created *endpoint* connection. URI: `/api/10.2/scan_endpoint/<endpoint>` HTTP method: POST Input parameters: (SCS cookie), verifiable data Result of successful execution: the ScanResult object, (SCS cookie)
`scan_path`	Action: Scan a file or a directory at the specified path. URI: `/api/10.2/scan_path` HTTP method: POST Input parameters: (SCS cookie), ScanPathOptions object Result of successful execution: the ScanPathResult object, (SCS cookie)
`scan_stat`	Action: Get scan statistics. URI: `/api/10.2/scan_stat` HTTP method: GET Input parameters: (SCS cookie), statistics format (JSON or CSV) Result of successful execution: the ScanStat object (if the JSON format is selected), (SCS cookie) If the CSV format is selected, a table whose columns correspond to the fields of the ScanStat object is returned.

Description of used JSON objects

1) `ScanOptions` is an object containing parameters used for creating *endpoint* for file scanning:

```
{
 "scan_timeout_ms": <number>, //Timeout for scanning one file, in ms
 "cure": <boolean>, //Cure or do not cure an infected file
 "heuristic_analysis": <boolean>, //Enable or disable the heuristic analysis
 "packer_max_level": <number>, //Maximum nesting level for packed objects
 "archive_max_level": <number>, //Maximum nesting level for archives
```

```
 "mail_max_level": <number>, //Maximum nesting level for email objects
 "container_max_level": <number>, //Maximum nesting level for other compound objects
(containters)
 "max_compression_ratio": <number>, //Maximum archive compression ratio
 "min_size_to_scan": <number>, //Minimal size of an object to be scanned
 "max_size_to_scan": <number>, //Maximal size of an object to be scanned
 "threat_hash": <boolean> //Get hashes of detected threats
}
```

2) `ScanPathOptions` is the object containing parameters for scanning a file or a directory at the specified path:

```
{
 "path": <string>, //Absolute path to a file or a directory to be scanned
 *"exclude_path": [<string>], //List of paths excluded from scanning (masks are
allowed)
 *"scan_timeout_ms": <number>, //Timeout for scanning one object
 *"archive_max_level": <number>, //Maximum nesting level for archived objects
 *"packer_max_level": <number>, //Maximum nesting level for packed objects
 *"mail_max_level": <number>, //Maximum nesting level for email messages
 *"container_max_level": <number>, //Maximum nesting level for other compound objects
(containters)
 *"max_compression_ratio": <number>, //Maximum archive compression ratio
 *"heuristic_analysis": <boolean>, //Enable or disable the heuristic analysis (default
value: true)
 *"follow_symlinks": <boolean>, //Follow or do not follow symbolic links
 *"min_size_to_scan": <number>, //Minimal size of an object to be scanned
 *"max_size_to_scan": <number>, //Maximal size of an object to be scanned
 *"timeout_ms": <number>, //Timeout for scanning all objects
 *"threat_hash": <boolean> //Return or do not return SHA1 and SHA256 hashes of threats
}
```

The parameters marked with `*` are optional.

3) `ScanPathResult` is an object containing results of scanning a file or a directory at the specified path:

```
{
  //ScanPathResult:
  "results": [<ScanResult>], //Scan results
  *"error": <string> //Error if the process has finished with an error (for example, by
timeout)
}
```

The parameter marked with `*` is optional.

If the scanning is successful, the `error` string will be absent in the response.

4) `ScanResult` is an object containing scan results:

```
{
 //ScanResult:
 "scan_report": <ScanReport>, //Scan report
 *"sha1": <string>, //SHA1 hash of the threat
 *"sha256": <string> //SHA256 hash of the threat
}
```

The parameters marked with `*` are optional.

5) `ScanReport` is an object containing information about the file with a threat:

```
{
 //ScanReport:
 "object": <string>, //Name of the scanned object
```

```
     // Absolute path for a file, the file name for a nested object
     // Always refers to a temporary file when calling scan_endpoint
 *"size": <number>, //Object size
 *"compressed_size": <number>, //Compressed object size
 *"core_fingerprint": <string>, //Engine fingerprint
 *"packer": [<string>], //List of packers used to pack the object
 *"compression_ratio": <number>, //Archive compression ratio
 *"archive": <Archive>, //Information about the archive (container) type, if the
object was identified as such
 *"virus": [<Virus>], //Threats detected in the object (if found)
 *"item": [<ScanReport>], //Reports on scanning nested objects (if any)
 *"error": <string>, //Scanning error (if occurred)
 *"heuristic_analysis": <boolean>, //Whether the heuristic analysis has been enabled
(true—enabled, false—disabled)
 *"cured": <boolean>, //Whether the object has been cured (true—cured, false—not
cured)
 *"cured_by_deletion": <boolean>, //Whether the object has been deleted while curing
(true—deleted, false—not deleted)
 *"new_path": <string>, //New path to the object renamed while curing
 *"user_time": <number>, //Time spent in the userspace
 *"system_time": <number> //Time spent on system calls while scanning
}
```

The parameters marked with `*` are optional.

The fields `virus` and `error` may be absent in the response if no threats were detected and no errors occurred during the scan. To call `scan_endpoint`, the `scan_endpoint` field always specifies a temporary file created by the Dr.Web Network Checker component in the server local file system and containing the data for scanning sent in the body of the `scan_endpoint` request).

6) `ScanEndpoint` is an object containing information about *endpoint* created for file scanning:

```
{
 "endpoint": <string> //Unique identifier of the created endpoint
}
```

The `endpoint` identifier returned in the object body. The identifier is used to start file scanning with the `scan_endpoint` command (as part of URI).

7) `VirusInfo` is an object containing information about the detected threat:

```
{
 "type": <string>, //Detected threat type
 "name": <string> //Name of the threat
}
```

The `type` field (threat type) is a string in the form of `SE_XXX`:

- `SE_KNOWN_VIRUS`—known threat,
- `SE_VIRUS_MODIFICATION`—modification of a known threat,
- `SE_UNKNOWN_VIRUS`—unknown threat (a suspicious object),
- `SE_ADWARE`—adware,
- `SE_DIALER`—dialer program,
- `SE_JOKE`—joke program,
- `SE_RISKWARE`—potentially dangerous software,

- SE_HACKTOOL—hacktool.

8) Archive is an object containing information about an archive, packed objects, email messages and other containers:

```
{
 "type": <string> //Archive type, can be one of the following:
         "SE_ARCHIVE" //Archive
         "SE_MAIL" //Email message
         "SE_CONTAINER" //Another container type
,"name": <string> //Archive format
}
```

9) ScanStat is an object containing scanning statistics:

```
{
 "origin": <string>, //Application at the request of which the scanning has been
performed
 // Counters for infected objects:
 "known_virus": <number>, //Number of objects infected by known threats
 "virus_modification": <number>, //Number of objects infected by a modification of a
known threat
 "unknown_virus": <number>, //Number of objects infected by unknown threats
 "adware": <number>, //Number of objects with SE_ADWARE
 "dialer": <number>, //Number of objects with SE_DIALER
 "joke": <number>, //Number of objects with SE_JOKE
 "riskware": <number>, //Number of objects with SE_RISKWARE
 "hacktool": <number>, //Number of objects with SE_HACKTOOL
 "cured": <number>, //Number of cured objects
 "quarantined": <number>, //Number of quarantined objects
 "deleted": <number> //Number of deleted objects
}
```

Managing the list of threats

To manage the list of threats that have been detected while scanning, the following HTTP API commands are available:

API command	Description
threats	Action: Get a list of identifiers of all detected threats. URI: /api/10.2/threats/ HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: an array of threat identifiers
threat_info	Action: Get information about a threat having the <*threat ID*> identifier. URI: /api/10.2/threat_info/<*threat ID*> HTTP method: GET

API command	Description
	Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), FileThreat object
`cure_threat`	Action: Attempt to cure a threat having the <*threat ID*> identifier. URI: `/api/10.2/cure_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`delete_threat`	Action: Delete the file containing the threat having the <*threat ID*> identifier. URI: `/api/10.2/delete_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`ignore_threat`	Action: Ignore the threat having the <*threat ID*> identifier. URI: `/api/10.2/ignore_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`quarantine_threat`	Action: Quarantine a threat having the <*threat ID*> identifier. URI: `/api/10.2/quarantine_threat/`<*threat ID*> HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object

Each found threat has a unique integer identifier <*threat ID*>. The list of identifiers for all detected threats is returned by the `threats` command. The `threat_info`, `cure_threat`, `delete_threat`, `ignore_threat`, and `quarantine_threat` commands allow the threat identifiers from this list only.

All information about a specific threat, including action history, can be obtained using the `threat_info` request. The information is returned as the FileThreat object.

Description of used JSON objects

1) `FileThreat` is an object containing the following information:

```
{
 "threat_id": <number>, //Threat identifier
 "detection_time": <Unix time>, //Date and time of threat detection
 "report": <ScanReport>, //File scan report
 "stat": <FileStat>, //Information about the file
 "origin": <string>, //Name of a component that detected the threat
 "origin_pid": <number>, //PID of the component that detected the threat
 "task_id": <number>, //Identifier of the scanning task for the scanning engine
 "history": [<ActionResult>] //History of actions applied to the threat (an array)
}
```

The `report` field contains a ScanReport object; the `stat` field contains a FileStat object, and the `history` field contains an array of ActionResult objects (the history of actions applied to the file).

2) `ScanReport` is an object containing information about the file with a threat:

```
{
 "object": <string>, //File system object containing the threat
 "size": <number>, //Size (in bytes) of the file containing the threat
 "virus": [<VirusInfo>], //List of details about the detected threats
 *"error": <string>, //Error message
 "heuristic_analysis": <boolean> //Whether the heuristic analysis has been enabled
(true—enabled, false—disabled)
}
```

The parameter marked with `*` is optional.

The `virus` field is an array of VirusInfo objects containing information about all detected threats. The `error` field is only present if an error has occurred.

3) `FileStat` is an object containing file statistics:

```
{
 "dev": <number>, //Device
 "ino": <number>, //Index node (inode)
 *"size": <number>, //File size
 *"uid": <User>, //User/owner identifier
 *"gid": <Group>, //Owner group identifier
 *"mode": <number>, //File access mode
 *"mtime": <Unix time>, //File modification date and time
 *"ctime": <Unix time>, //File creation date and time
 *"rsrc_size": <number>,
 *"finder_info": <string>,
 *"ext_finder_info": <string>,
 *"uchg": <string>,
 *"volume_name": <string>, //Volume name
 *"volume_root": <string>, //Volume root (mount point)
 *"xattr": [<Xattr>] //Extended information about the file
}
```

The parameters marked with `*` are optional.

The `xattr` field contains an array of `Xattr` objects. This object contains two string-type fields: `name` and `value`. The `uid` and `gid` fields represent `User` and `Group` objects that store information about a user/owner and an owner group respectively. These objects contain two fields each:

- `uid` (`gid`)—numeric ID of the user (group);
- `username` (`groupname`)—name of the user (group) as a string.

4) `ActionResult` is an object containing information about the action applied to the file and its result:

```
{
 "action": <string>, //Applied action
 "action_time": <Unix time>, //Date and time of applying the action
 "result": <string>, //Result of applying the action
 "cure_report": <ScanReport> //Report on applying the action
}
```

The `cure_threat`, `delete_threat`, `ignore_threat` and `quarantine_threat` commands return an empty object when run successfully. If the requested action to be applied to the threat fails (for example, the threat cannot be neutralized), an Error object is returned instead of an empty object.

Managing quarantine

To manage quarantined threats, HTTP API provides the following commands:

API command	Description
`quarantine`	Action: List identifiers of quarantined objects. URI: `/api/10.2/quarantine/` HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), array of QuarantineId objects (quarantined objects)
`qentry_info`	Action: Get information about the quarantined object having the *<entry ID>* identifier. URI: `/api/10.2/qentry_info/`*<entry ID>* HTTP method: GET Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), QEntry object
`cure_qentry`	Action: Attempt to cure the quarantined object with the *<entry ID>* identifier.

API command	Description
	URI: `/api/10.2/cure_qentry/<entry ID>` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`delete_qentry`	Action: Delete the quarantined object with the *<entry ID>* identifier. URI: `/api/10.2/delete_qentry/<entry ID>` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object
`restore_qentry`	Action: Restore the quarantined object with the *<entry ID>* identifier to its original location. URI: `/api/10.2/restore_qentry/<entry ID>` HTTP method: POST Input parameters: (SCS cookie) Result of successful execution: (SCS cookie), an empty object

The `quarantine` command returns the array of QuarantineId objects, each of which contains the identifier of a quarantined object. The identifier consists of two parts: `chunk_id` and `entry_id`.

Description of used JSON objects

1) `QuarantineId` is an object containing parts of the compound identifier of a quarantined object:

```
{
 "chunk_id": <string>,
 "entry_id": <string>
}
```

A combination of these two fields in the form of *<entry_id>@<chunk_id>* represents the identifier of a quarantined object. This identifier is provided in all requests for performing actions under any quarantined object using the `qentry_info`, `cure_qentry`, `delete_qentry` and `restore_qentry` commands. The `qentry_info` command allows to get the detailed information about a quarantined object with a specified identifier. This command returns a QEntry object.

2) `QEntry` is an object containing information about a quarantined object:

```
{
 "entry_id": <string>, //Parts of the identifier
 *"chunk_id": <string>, //of the quarantined object
 *"quarantine_dir": <string>, //Quarantine directory
 "restore_path": <string>, //Path to restore
 "creation_time": <Unix time>, //Date and time when the object was quarantined
 "report": <ScanReport>, //Object scan report (see ScanReport above)
 "stat": <FileStat>, //File statistics (see FileStat above)
 *"history": [<QEntryOperation>], //History of operations performed on the object
 *"who": <RemoteUser>, //Remote owner of the file (if the file was quarantined from a
file server storage having a network access)
 *"detection_time": <Unix time>, //Date and time of threat detection
 *"origin": <string> //Component that detected the threat
}
```

The parameters marked with `*` are optional.

The `report` field contains a ScanReport object, the `stat` field contains a FileStat object, and the `history` field contains the history of actions applied to the quarantined object. Each action record is described by a QEntryOperation object. The optional `who` field contains information about the remote user in the form of a RemoteUser object.

3) `QEntryOperation` is an object containing information about an operation applied to the quarantined object:

```
{
 "action": <string>, //Action applied to the object (see the designations below)
 "action_time": <Unix time>, //Date and time of applying the action
 "result": <string>, //Error of applying the action (a code in the form of EC_XXX)
 *"restore_path": <string>, //Path for restoring the quarantined object (with action =
"QENTRY_ACTION_RESTORE")
 *"rescan_report": <ScanReport> //Report on rescanning (with action =
"QENTRY_ACTION_RESCAN")
}
```

The parameters marked with `*` are optional.

The `action` field can have the following values:

- `QENTRY_ACTION_DELETE`—attempt to delete the quarantined object;
- `QENTRY_ACTION_RESTORE`—attempt to restore the quarantined object;
- `QENTRY_ACTION_RESCAN`—attempt to rescan the quarantined object;
- `QENTRY_ACTION_CURE`—attempt to cure the quarantined object.

4) `RemoteUser` is an object containing information about the remote user who owns the file (if the file was relocated to quarantine from the server storage):

```
{
 *"ip": <string>, //User IP address
 *"user": <string>, //User name
 *"domain": <string> //User domain
}
```

The parameters marked with `*` are optional.

The `cure_qentry`, `delete_qentry` and `restore_qentry` commands return an empty object if the command execution was successful. In case the command failed, an Error object will be returned in the response.

HTTP API usage examples

To test the HTTP API, you can use the `curl` utility. The structure and format of the request can be provided as follows:

```
$ curl https://<HTTPD.AdminListen>/<HTTP API URI> -k -X <HTTP method>
 [-H 'Content-Type: application/json' --data-binary '@<JSON object file>']
 [-c <cookie file> [-b <cookie file>]] [> <result file>]
```

- the `-k` option indicates that `curl` does not have to verify the used SSL certificate;
- the `-X` option specifies the HTTP method being used (GET or POST);
- the `-H` option is used to add the `Content-Type: application/json` header;
- the `--data-binary` (or `-d`) option is used to add a JSON object read from a text file to the body request;
- if SCS is used for authorization, the files containing the sent and received SCS cookies must be specified in the `-b` and `-c` parameters respectively.

The detailed description of the `curl` utility options can be displayed using the `curl --help` and `man curl` commands.

1. **Authorize a client** by specifying a user name and a password (for SCS).

   An AuthOptions object in the JSON format must be stored in advance in the `user.json` file, for example:

   ```
   {"user":"<username>","password":"<passphrase>"}
   ```

   Request:

   ```
   $ curl https://127.0.0.1:4443/api/10.2/login -k -X POST -H 'Content-Type:
   application/json' --data-binary '@user.json' -c cookie.file
   ```

   Response:

   ```
   HTTP/1.0 200 OK
   Content-Type: application/json
   Content-Length: 2
   Set-Cookie: DWToken=6QXy4wn_JGov9A1GohWP_kvMK3dN6ccKegjNgKcmHpb_AqSrHg9cNX_yFJhxPDgr|
   MTQ2Mjg3Mzg4NQ==|cWd4Ow==|GywBUVOhU4w2LF_BKT5frg==|kR_rip5nrpxWjJ2dfZ7Xfmvi3rE=;
   Secure; HttpOnly; Max-Age: 900; Path=/
   Pragma: no-cache

   {}
   ```

   The `Set-Cookie` header contains an SCS cookie to be used in all further requests to the HTTP API. If the authorization was successful, the body of the response will contain an empty object. If the user has not been authorized, the Error object is returned:

   ```
   HTTP/1.0 403 Forbidden
   Content-Type: application/json
   Content-Length: 35
   ```

```
Pragma: no-cache

{"error":{"code":"EC_AUTH_FAILED"}}
```

2. **Get information about the threat** having the *ID = 1* identifier:

   Request:

```
$ curl https://127.0.0.1:4443/api/10.2/threat_info/1 -k -X GET -c cookie.file -b
cookie.file
```

   Response:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 574
Set-Cookie: DWToken=<…>;
 Secure; HttpOnly; Max-Age: 900; Path=/
Pragma: no-cache

{"threat_id":1,"detection_time":1462881660,
"report":{"object":"/sites/site1/eicar.com.txt","size":68,"packer":[],
"virus":[{"type":"SE_KNOWN_VIRUS","name":"EICAR Test File (NOT a Virus!)"}],
"heuristic_analysis":true,"core_fingerprint":"0D2DD5A869DAB7AE354153A4D5F70F45",
"item":[],"log":[],"user_time":0,"system_time":0},"stat":{"dev":2049,"ino":898,
"size":68,"uid":{"uid":1000,"username":"user"},"gid":{"gid":1000,"groupname":"user"},
"mode":33204,"mtime":1441028214,"ctime":1460738554,"xattr":[]},
"origin":"APP_COMMAND_LINE_TOOL","origin_pid":2726,"task_id":1,"history":[]}
```

3. **Quarantine** the threat having the *ID = 1* identifier:

   Request:

```
$ curl -v -c cookie.jar -b cookie.jar -k -X POST -H 'Content-Type:application/json'
https://127.0.0.1:4443/api/10.2/quarantine_threat/1
```

   Response:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 2
Set-Cookie: DWToken=<…>; Secure; HttpOnly; Max-Age: 900; Path=/
Pragma: no-cache

{}
```

4. **View information about the quarantined object**:

   Request:

```
$ curl -v -k -X GET -c cookie.jar -b cookie.jar
https://127.0.0.1:4443/api/10.2/qentry_info/3070d3ce-7b6e-4143-9d9f-
89ba3473a781@801:2108d
```

   Response:

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 781
Set-Cookie: DWToken=<…>; Secure; HttpOnly; Max-Age: 900; Path=/
Pragma: no-cache

{"entry_id":"3070d3ce-7b6e-4143-9d9f-89ba3473a781","chunk_id":"3830313A3231303864",
"quarantine_dir":"2F686F6D652F757365722F2E636F6D2E64727765622E71756172616E74696E65",
"restore_path":"2E2E2F7473742F65696361722E636F6D2E747874","creation_time":1462888884,
```

```
"report":{"object":"/home/user/tst/eicar.com.txt","size":68,"packer":[],
"virus":[{"type":"SE_KNOWN_VIRUS","name":"EICAR Test File (NOT a Virus!)"}],
"heuristic_analysis":true,"core_fingerprint":"467CD4C6D423C55448B71CD5B8152776",
"item":[],"log":[],"user_time":0,"system_time":0},"stat":{"dev":2049,"ino":898,
"size":68,"uid":{"uid":1000,"username":"user"},"gid":{"gid":1000,"groupname":"user"},
"mode":33204,"mtime":1441028214,"ctime":1462888421,"xattr":[]},"history":[],
"detection_time":1462888667,"origin":"APP_COMMAND_LINE_TOOL"}
```


5. **Change settings:** disable Dr.Web CloudD.

   A LexMap object in the JSON format must be stored in advance in the `lexmap_ls_off.json` file:


   ```
   {"option":[{"key":"Root","map":{"option":
   [{"key":"UseCloud","value":{"item":["no"]}}]}}]}
   ```


   Request:

   ```
   $ curl -v -k -c cookie.jar -b cookie.jar -X POST -H 'Content-Type: application/json'
   --data-binary '@lexmap_ls_off.json' https://127.0.0.1:4443/api/10.2/set_lexmap
   ```

   Response:

   ```
   HTTP/1.0 200 OK
   Content-Type: application/json
   Content-Length: 58
   Set-Cookie: DWToken=<…>; Secure; HttpOnly; Max-Age: 900; Path=/
   Pragma: no-cache

   {"item":[{"option":"Root.UseCloud","result":"EC_OK"}]}
   ```

6. **Change settings:** disable Dr.Web CloudD.

   A LexMap object in the JSON format must be stored in advance in the `lexmap_ls_on.json` file:


   ```
   {"option":[{"key":"Root","map":{"option":
   [{"key":"UseCloud","value":{"item":["yes"]}}]}}]}
   ```


   Request:

   ```
   $ curl -v -k -c cookie.jar -b cookie.jar -X POST -H 'Content-Type: application/json'
   --data-binary '@lexmap_ls_on.json' https://127.0.0.1:4443/api/10.2/set_lexmap
   ```

   Response:

   ```
   HTTP/1.0 200 OK
   Content-Type: application/json
   Content-Length: 58
   Set-Cookie: DWToken=<…>; Secure; HttpOnly; Max-Age: 900; Path=/
   Pragma: no-cache

   {"item":[{"option":"Root.UseCloud","result":"EC_OK"}]}
   ```

9.14. Dr.Web SNMPD

Dr.Web SNMPD is an SNMP agent designed to integrate Dr.Web Gateway Security Suite with monitoring systems running the SNMP protocol. This integration allows you to track the status of the Dr.Web Gateway Security Suite components, as well as collect statistics on the detection and neutralization of threats. The agent supports providing monitoring systems or any SNMP managers with the following information:

- Status of any Dr.Web Gateway Security Suite component.
- Number of detected threats of various types (according to the Dr.Web classification).

Moreover, the agent sends SNMP trap notifications upon detection of a threat and upon failures in neutralization of detected threats. The agent supports SNMP protocol of version 2c and 3.

Description of the information which can be sent by the agent is stored in a special section of MIB (*Management Information Base*) created by Doctor Web. In the MIB section, defined by Dr.Web for Unix-like operating systems, the following information is specified:

1. Formats of SNMP trap notifications about detection and neutralizing of threats and about errors related to the Dr.Web Gateway Security Suite components.
2. Dr.Web Gateway Security Suite operation statistics.
3. Dr.Web Gateway Security Suite component status.

For more details about information that can be obtained over the SNMP protocol, refer to the corresponding section.

9.14.1. Operating Principles

In this section

- General information
- Integration with the system SNMP agent

General information

By default, the component is run automatically upon Dr.Web Gateway Security Suite startup. When run, the component structures data according to the structure described in Dr.Web SNMP MIB and waits for requests to receive data from external SNMP managers. The component receives information on the status of the Dr.Web Gateway Security Suite components and notifications on detected threats directly from the Dr.Web ConfigD configuration daemon.

Threats can be detected by the scan engine during scanning initiated by Dr.Web Gateway Security Suite components. Once any of threats is detected, the appropriate count (of this threat type) is incremented by one and all SNMP managers that can receive notifications get an SNMP trap notifying on the detected threat.

Collected values of counters (for example, counters of detected threats) are not saved between launches of Dr.Web SNMPD. Thus, once Dr.Web SNMPD is relaunched for any reason (including general restart of Dr.Web Gateway Security Suite), the collected values of counters are reset to zero.

Integration with the system SNMP agent

To enable correct operation of Dr.Web SNMP agent if the main system SNMP agent `snmpd` (`net-snmp`) already operates on the server, configure transmission of SNMP requests through the Dr.Web MIB branch from `snmpd` to Dr.Web SNMPD. For that purpose, edit the `snmpd` configuration file (usually, `/etc/snmp/snmpd.conf` for GNU/Linux or `/etc/snmpd.config` for FreeBSD), by adding the following line to it:

```
proxy -v <version> -c <community> <address>:<port> <MIB branch>
```

Where:

- <*version*>—SNMP version in use (2c, 3);
- <*community*>—"community string" used by Dr.Web SNMPD;
- <*address*>:<*port*>—network socket listened by Dr.Web SNMPD;
- <*MIB branch*>—OID of the MIB branch containing descriptions of variables and SNMP notifications (*trap*) used by Dr.Web (the OID equals `.1.3.6.1.4.1.29690`).

If you are using the default settings of Dr.Web SNMP agent, then the added line should look like this:

```
proxy -v 2c -c public localhost:50000 .1.3.6.1.4.1.29690
```

Since port 161 in this case will be used by the system standard `snmpd`, it is required to specify another port for Dr.Web SNMPD in the `ListenAddress` parameter (in this example, `50000`).

9.14.2. Command-Line Arguments

To start the Dr.Web SNMPD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-snmpd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-snmpd [<parameters>]
```

Dr.Web SNMPD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-snmpd --help
```

This command outputs short help information about the Dr.Web SNMPD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary (usually, at the startup of the operating system). To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-snmpd` command.

9.14.3. Configuration Parameters

The component uses configuration parameters specified in the `[SNMPD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component.

Parameter	Description
	If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-snmpd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-snmpd`
`Start` *{boolean}*	Launch/do not launch the component by the Dr.Web ConfigD configuration daemon. When you specify the `Yes` value for this parameter, it the configuration daemon will start the component immediately; and when you specify the `No` value, the configuration daemon will terminate the component immediately. Default value: `No`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`SnmpVersion` *{V2c | V3}*	Current version of SNMP protocol (*SNMPv2c* or *SNMPv3*). Default value: `V2c`
`ListenAddress` *{address}*	Address (IP address and port) listened by Dr.Web SNMPD, which is waiting for client connections (SNMP managers). Interoperability with `snmpd` requires specifying a port different from the standard port (`161`). In addition, `snmpd` must be configured for proxying. Default value: `127.0.0.1:161`

Parameter	Description

`TrapReceiver`
 	
{address list}
	List of addresses (IP address and port) to which Dr.Web SNMPD sends *SNMP trap* notifications when Dr.Web Gateway Security Suite components detect a threat.
	
	Accepts a list of values. The values in the list must be comma-separated (with each value put in quotation marks). The parameter can be specified more than once in the section (in this case, all its values are combined into one list).
	
	Example: Add sockets 192.168.0.1:1234 and 10.20.30.45:5678 to the list.
	
	1. Adding values to the configuration file.
	• Two values per line:
	`[SNMPD]`
	`TrapReceiver = "192.168.0.1:1234", "10.20.30.45:5678"`
	• Two lines (one value per line):
	`[SNMPD]`
	`TrapReceiver = 192.168.0.1:1234`
	`TrapReceiver = 10.20.30.45:5678`
	2. Adding the values using the `drweb-ctl cfset` command:
	`# drweb-ctl cfset SNMPD.TrapReceiver -a 192.168.0.1:1234`
	`# drweb-ctl cfset SNMPD.TrapReceiver -a 10.20.30.45:5678`
	
	Default value: *(not specified)*

`V2cCommunity` *{string}*	"SNMP read community" string for authentication of SNMP managers (*SNMPv2c* protocol) when Dr.Web MIB variables are accessed for reading. The parameter is used if `SnmpVersion = V2c`. Default value: `public`
`V3EngineId` *{string}*	Identifier (string) of *Engine ID* for *SNMPv3* (according to RFC 3411). Default value: `800073FA044452574542`
`V3UserName` *{string}*	User name for authentication of SNMP managers (*SNMPv3* protocol) when Dr.Web MIB variables are accessed for reading. The parameter is used if `SnmpVersion = V3`. Default value: `noAuthUser`

Parameter	Description

`V3Auth`
 	
{SHA(<pwd>) | MD5(<pwd>) | None}
	Method to authenticate SNMP managers (*SNMPv3* protocol) when Dr.Web MIB variables are accessed for reading.
	
	Allowed values:
	• `SHA(<PWD>)`—SHA hash of the password is used (<*PWD*> strings);
	• `MD5(<PWD>)`—MD5 hash of the password is used (<*PWD*> strings);
	• `None`—authentication is disabled;
	where <*PWD*> is a plain text password.
	
	When specifying the parameter value from the command line, you may need to escape the brackets by using the slash mark `\` in some shells.
	
	Examples:
	1. Parameter value in the configuration file:
	`V3Auth = MD5(123456)`
	2. Specifying the same parameter value from the command line using the `drweb-ctl cfset` command:
	`drweb-ctl cfset SNMPD.V3Auth MD5\(123456\)`
	
	The parameter is used if `SnmpVersion = V3`.
	
	Default value: `None`

`V3Privacy` *{DES(<secret>) | AES128(<secret>) | None}*	Encryption method for SNMP messages (*SNMPv3* protocol). Allowed values: • `DES(<secret>)`—DES encryption algorithm; • `AES128(<secret>)`—AES128 encryption algorithm; • `None`—SNMP-messages are not encrypted; where <*secret*> is a secret key shared by the manager and the agent (*plain text*). When specifying the parameter value from the command line, you may need to escape the brackets by using the slash mark `\` in some shells. Examples: 1. Parameter value in the configuration file: `V3Privacy = AES128(supersecret)`

Parameter	Description
	2. Specifying the same parameter value from the command line using the `drweb-ctl cfset` command: `drweb-ctl cfset SNMPD.V3Privacy AES128\(supersecret\)` The parameter is used if `SnmpVersion = V3`. Default value: `None`

9.14.4. Integration with Monitoring Systems

In this section

- Integration with the Munin monitoring system
  - Connecting a host to Munin
- Integration with the Zabbix monitoring system
  - Connecting a host to Zabbix
  - Configuring receipt of SNMP trap notifications for Zabbix
- Integration with the Nagios monitoring system
  - Connecting a host to Nagios
  - Configuring Nagios
  - Configuring receipt of SNMP trap notifications for Nagios

The Dr.Web SNMP agent can act as a data provider for any monitoring system that uses SNMP *2c* or *3*. The list of data available for monitoring and their structure are provided in the Dr.Web MIB description file `DRWEB-SNMPD-MIB.txt` supplied together with Dr.Web Gateway Security Suite. This file is located in the directory `/opt/drweb.com/share/drweb-snmpd/mibs` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/mibs` for FreeBSD.

For ease of configuration, the component is supplied with the required configuration templates for popular monitoring systems such as Munin, Zabbix and Nagios.

Configuration templates for monitoring systems are located in the directory `/opt/drweb.com/share/drweb-snmpd/connectors` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors` for FreeBSD.

Integration with the Munin monitoring system

The Munin monitoring system includes a munin centralized server (master), which collects statistics from munin-node clients installed locally on hosts to be monitored. At the request of the server, each monitoring client collects data about monitored host operation by starting *plug-ins* that provide data to be sent to the server.

To connect Dr.Web SNMPD to the Munin monitoring system, ready-to-use Dr.Web data collection plug-ins used by munin-node are supplied. These plug-ins are located in the directory `/opt/drweb.com/share/drweb-snmpd/connectors/munin/plugins` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/munin/plugins` for FreeBSD and collect data required for drawing the following graphs:

- a number of detected threats;
- file scan statistics.

These plug-ins support SNMP 1, 2c and 3. Based on these template plug-ins, you can create any other plug-ins to poll the status of Dr.Web Gateway Security Suite components via Dr.Web SNMPD.

The directory `/opt/drweb.com/share/drweb-snmpd/connectors/munin` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/munin` for FreeBSD contains the following files.

File	Description
`plugins/snmp__drweb_malware`	`munin-node` plug-in for polling Dr.Web SNMPD via SNMP to get a number of threats detected by Dr.Web Gateway Security Suite on the host.
`plugins/snmp__drweb_filecheck`	`munin-node` plug-in for polling Dr.Web SNMPD via SNMP to gather statistics on scanning files by Dr.Web Gateway Security Suite on the host.
`plugin-conf.d/drweb.cfg`	Example of the `munin-node` configuration for the runtime environment variables of the Dr.Web plug-ins.

Connecting a host to Munin

This instruction assumes that the Munin monitoring system is already deployed on the monitoring server, and the monitored host features installed and functioning Dr.Web SNMPD (optionally, in proxy mode together with snmpd) and `munin-node`.

1. Monitored host configuration
   - Copy the `snmp__drweb_*` files to the directory containing `munin-node` plug-in libraries. This path depends on the OS, for example, the path is `/usr/share/munin/plugins` on Debian and Ubuntu.
   - Configure `munin-node` by connecting the supplied Dr.Web plug-ins to it. To do this, use the `munin-node-configure` utility, which is distributed with `munin-node`.

     For example, the command:

     ```
     $ munin-node-configure --shell --snmp localhost
     ```

will display a list of commands for creation of required symbolic links to plug-ins on a terminal screen. Copy and run them in the command line. The specified command presumes that:

1) `munin-node` is installed on the same host as Dr.Web SNMPD. Otherwise, specify a valid FQDN or IP address of the monitored host instead of `localhost`.
2) Dr.Web SNMPD uses SNMP 2c. Otherwise, specify the correct SNMP version for the `munin-node-configure` command. This command accepts a set of parameters for flexible configuration of plug-ins, for example, you can specify the SNMP version in use, a port that is used by an SNMP agent at the monitored host, a value of *community string* and so on. If required, refer to the manual for the `munin-node-configure` command.

- If necessary, define or redefine parameter values of the environment where the installed Dr.Web plug-ins for `munin-node` must be run. As the environment parameters, the value of *community string*, the port used by the SNMP agent and so on are used. These parameters must be defined in the `/etc/munin/plugin-conf.d/drweb` file (create it if required). You can use the supplied `drweb.cfg` file as an example.
- In the `munin-node` configuration file (`munin-node.conf`), specify a regular expression to include IP addresses of hosts that are allowed to connect munin servers (masters) to `munin-node` on the current host for receiving the values of the monitored parameters, for example:

  ```
  allow ^10\.20\.30\.40$
  ```

  In this case, only a host with the IP address of `10.20.30.40` is allowed to receive the parameters of that host.
- Restart `munin-node`, for example, by running the command:

  ```
  # service munin-node restart
  ```

2. Munin server (master) configuration

   Add the address and identifier of the monitored host to the Munin configuration file (`munin.conf`), which is located, by default, in the `/etc` directory (`/etc/munin/munin.conf` on Debian and Ubuntu):

   ```
   [<ID>;<hostname>.<domain>]
   address <host IP address>
   use_node_name yes
   ```

   where <*ID*> is the displayed host identifier, <*hostname*> is the name of the host, <*domain*> is the name of the domain, <*host IP address*> is the IP address of the host.

For official documentation on configuring the Munin monitoring system, follow the link.

Integration with the Zabbix monitoring system

The directory `/opt/drweb.com/share/drweb-snmpd/connectors/zabbix` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/zabbix` for FreeBSD contains the following template files required for connecting Dr.Web SNMPD to the Zabbix monitoring system:

File	Description
`zbx_drweb.xml`	Template for description of the monitored host that features installed Dr.Web Gateway Security Suite.
`snmptt.drweb.zabbix.conf`	Settings of the `snmptt` utility, which receives *SNMP trap* notifications.

The template for description of the monitored host features:

- A set of descriptions of counters ("*items*" in the terminology of Zabbix). By default, the template is set to be used with SNMP v2.
- A set of predefined graphs: a number of scanned files and distribution of detected threats by their types.

Connecting a host to Zabbix

In the present instruction, it is assumed that the Zabbix monitoring system is already properly deployed on the monitoring server and the monitored host features an installed and properly functioning Dr.Web SNMPD (optionally, in proxy mode together with `snmpd`). Furthermore, if you want to receive *SNMP trap* notifications from the monitored host (including notification of threats detected by Dr.Web Gateway Security Suite on the protected server), install the `net-snmp` package on the monitoring server (the `snmptt` and `snmptrapd` standard tools are used).

1. In the Zabbix web interface, on the **Configuration** → **Templates** tab, import the template of the monitored host from the file `/opt/drweb.com/share/drweb-snmpd/connectors/zabbix/zbx_drweb.xml` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/zabbix/zbx_drweb.xml` for FreeBSD.
2. Add the monitored host to the list of hosts (at **Hosts** → **Create host**). Specify parameters of the host and valid settings of the SNMP interface (they must match the settings of `drweb-snmpd` and snmpd on the host):
   - **Host** tab:

     Host name: *drweb-host*

     Visible name: *DRWEB_HOST*

     Groups: select **Linux servers**

Snmp interfaces: click **Add** and specify the IP address and the port used by Dr.Web SNMPD (by default, it is assumed that Dr.Web SNMPD operates on the local host, so address *127.0.0.1* and standard port *161* are indicated here).

- **Templates** tab:

  Click **Add**, select *DRWEB*, click **Select**.

- **Macros** tab:

  Macro: *{$SNMP_COMMUNITY}*

  Value: select *read community* for SNMP V2c (*public* by default).

  Click **Save**.



The *{$SNMP_COMMUNITY}* macro can be specified directly in the host template.

By default, the imported *DRWEB* template is configured for SNMP v2. If you need to use another version of SNMP, edit the template on the corresponding page.

3. After the template is bound to the monitored host, if SNMP settings are valid, the Zabbix monitoring system will start to collect data for counters (*items*) of the template. The collected data will be displayed on the following tabs of the web interface: **Monitoring** → **Latest Data** and **Monitoring** → **Graphs**.
4. A special *item drweb-traps* is used for collecting *SNMP trap* notifications from Dr.Web SNMPD. The log of received *SNMP trap* notifications is available on the page **Monitoring** → **Latest Data** → **drweb-traps** → **history**. To collect notifications, Zabbix uses the standard `snmptt` and `snmptrapd` utilities from the `net-snmp` package. For details on how to configure these utilities for receiving *SNMP trap* notifications from Dr.Web SNMPD, see below.
5. If necessary, you can configure a trigger for the added monitored host. The trigger will change its state upon receiving *SNMP trap* notifications from Dr.Web SNMPD. Changing the state of this trigger can be used as an event source for generating corresponding notifications. A trigger for the monitored host is added in a common way. The example below shows an expression specified in the **trigger expression** field for the aforesaid trigger.

   - For Zabbix *2.x*:

     ```
     ({TRIGGER.VALUE}=0 & {DRWEB:snmptrap[.*\.1\.3\.6\.1\.4\.1\.29690\..*].nodata(60)}
     =1 )|({TRIGGER.VALUE}=1 &
     {DRWEB:snmptrap[.*\.1\.3\.6\.1\.4\.1\.29690\..*].nodata(60)}=0)
     ```

   - For Zabbix *3.x*:

     ```
     ({TRIGGER.VALUE}=0 and {drweb-host:snmptrap[".29690."].nodata(60)}=1 ) or
     ({TRIGGER.VALUE}=1 and {drweb-host:snmptrap[".29690."].nodata(60)}=0 )
     ```

   This trigger is activated (its value is set to 1) if the log of *SNMP trap* notifications from Dr.Web SNMPD was updated within the last minute. If the log was not updated within the last minute, the trigger is deactivated (its value is set to 0).

It is recommended that a notification type different from *Not classified*, for example, *Warning*, is indicated in the **Severity** field for this trigger.

Configuring receipt of SNMP trap notifications for Zabbix

1. On the monitored host, in Dr.Web SNMPD settings (the `TrapReceiver` parameter), specify an address to be listened by `snmptrapd` on the Zabbix host, for example:

   ```
   SNMPD.TrapReceiver = 10.20.30.40:162
   ```

2. In the `snmptrapd` configuration file (`snmptrapd.conf`), specify the same address and an application for processing received *SNMP trap* notifications (in this example, `snmptthandler`, which is a component of `snmptt`):

   ```
   snmpTrapdAddr 10.20.30.40:162
   traphandle default /usr/sbin/snmptthandler
   ```

   Add the following line to the file, so that `snmptt` does not discard *SNMP trap* notifications sent by Dr.Web SNMPD, as unknown:

   ```
   outputOption n
   ```

3. The `snmptthandler` component stores received *SNMP trap* notifications on the file on the disk in accordance with the specified format, which must correspond to the regular expression set in the host template for Zabbix (the *drweb-traps item*). The format of the stored message on the *SNMP trap* notification is specified in the file `/opt/drweb.com/share/drweb-snmpd/connectors/zabbix/snmptt.drweb.zabbix.conf` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/zabbix/snmptt.drweb.zabbix.conf` for FreeBSD, which must be copied to the `/etc/snmp` directory.

4. Furthermore, the path to the format files must be specified in the `snmptt.ini` configuration file:

   ```
   [TrapFiles]
   # A list of snmptt.conf files (this is NOT the snmptrapd.conf file).
   # The COMPLETE path and filename. Ex: '/etc/snmp/snmptt.conf'
   snmptt_conf_files = <<END
   /etc/snmp/snmptt.conf
   /etc/snmp/snmptt.drweb.zabbix.conf
   END
   ```

   After that, restart `snmptt` if it was started in daemon mode.

5. Specify the following settings (or check if they are already specified) in the configuration file of the Zabbix server (`zabbix-server.conf`):

   ```
   SNMPTrapperFile=/var/log/snmptt/snmptt.log
   StartSNMPTrapper=1
   ```

   where `/var/log/snmptt/snmptt.log` is a log file used by `snmptt` to register information about received SNMP trap notifications.

For official documentation on Zabbix, follow the link.

Integration with the Nagios monitoring system

The directory `/opt/drweb.com/share/drweb-snmpd/connectors/nagios` for GNU/Linux or `/usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/nagios` for FreeBSD contains the following Nagios example configuration files required for connecting Dr.Web SNMPD to the Nagios monitoring system:

File	Description
`nagiosgraph/rrdopts.conf-sample`	Example of the RRD configuration file
`objects/drweb.cfg`	Configuration file describing *drweb* objects
`objects/nagiosgraph.cfg`	Configuration file of the Nagiosgraph component for graph plotting, used by Nagios
`plugins/check_drweb`	Script for collecting data from a Dr.Web Gateway Security Suite host
`plugins/eventhandlers/submit_check_result`	Script for handling *SNMP trap* notifications
`snmp/snmptt.drweb.nagios.conf`	Settings of the `snmptt` utility, which receives *SNMP trap* notifications

Connecting a host to Nagios

In the present instruction, it is assumed that the Nagios monitoring system is already properly deployed on the monitoring server (including configuring the web server and the Nagiosgraph graphical tool) and the monitored host features installed and properly functioning Dr.Web SNMPD (optionally, in proxy mode together with snmpd). Furthermore, if you want to receive an *SNMP trap* notification from the monitored host (including notifications of threats detected by Dr.Web Gateway Security Suite on the protected server), install the `net-snmp` package on the monitoring server (the `snmptt` and `snmptrapd` standard utilities are used).

In the current manual, the following path conventions are used (actual paths depend on the operating system and Nagios installation):

- <*NAGIOS_PLUGINS_DIR*>—directory with Nagios plug-ins, for example, `/usr/lib64/nagios/plugins`.
- <*NAGIOS_ETC_DIR*>—directory with Nagios settings, for example, `/etc/nagios`.
- <*NAGIOS_OBJECTS_DIR*>—directory with Nagios objects, for example, `/etc/nagios/objects`.
- <*NAGIOSGRAPH_DIR*>—Nagiosgraph directory, for example, `/usr/local/nagiosgraph`.
- <*NAGIOS_PERFDATA_LOG*>—file in which Nagios stores results obtained from running service scan commands (must be the same as the `perflog` file from

<*NAGIOSGRAPH_DIR*>/etc/nagiosgraph.conf). Records from this file are read by the <*NAGIOSGRAPH_DIR*>/bin/insert.pl script and are written to the corresponding RRA archives of RRDtool.

Configuring Nagios

1. Copy the check_drweb file to the <*NAGIOS_PLUGINS_DIR*> directory and the drweb.cfg file to the <*NAGIOS_OBJECTS_DIR*> directory.
2. Add Dr.Web Gateway Security Suite hosts to be monitored to the *drweb* group (Dr.Web SNMPD must be running on them). By default, only the *localhost* local host is added to this group.
3. If required, edit the check_drweb command to communicate with Dr.Web SNMPD on *drweb* hosts via the snmplwalk utility:

   ```
   snmpwalk -c public -v 2c $HOSTADDRESS$:161
   ```

   Specify the correct SNMP version and such parameters as *community string*, or authentication parameters as well as a port. The $HOSTADDRESS$ variable must be included in the command (as this variable is later automatically substituted by Nagios with the correct host address when the command is invoked). It is not required to indicate OID in the command. It is also recommended that you specify the command together with the full path to the executable file (usually /usr/local/bin/snmpwalk).
4. Connect *DrWeb* objects in the <*NAGIOS_ETC_DIR*>/nagios.cfg configuration file by appending it with the following line:

   ```
   cfg_file=<NAGIOS_OBJECTS_DIR>/drweb.cfg
   ```
5. Add RRDtool settings for *DrWeb* graphs from the rrdopts.conf-sample file to the <*NAGIOSGRAPH_DIR*>/etc/rrdopts.conf file.
6. Configure the Nagiosgraph component if it is still not configured:
   - Copy the nagiosgraph.cfg file to the <*NAGIOS_OBJECTS_DIR*> directory and edit the path to the insert.pl script in the process-service-perfdata-for-nagiosgraph command, for example, as follows:

     ```
     $ awk '$1 == "command_line" { $2 = "<NAGIOSGRAPH_DIR>/bin/insert.pl" }{ print }'
     ./objects/nagiosgraph.cfg > <NAGIOS_OBJECTS_DIR>/nagiosgraph.cfg
     ```
   - Connect this file in the <*NAGIOS_ETC_DIR*>/nagios.cfg configuration file by appending the following line to it:

     ```
     cfg_file=<NAGIOS_OBJECTS_DIR>/nagiosgraph.cfg
     ```

7. Verify Nagios parameter values in the <*NAGIOS_ETC_DIR*>/nagios.cfg configuration file:

```
check_external_commands=1
execute_host_checks=1
accept_passive_host_checks=1
enable_notifications=1
enable_event_handlers=1

process_performance_data=1
service_perfdata_file=/usr/nagiosgraph/var/rrd/perfdata.log
service_perfdata_file_template=$LASTSERVICECHECK$||$HOSTNAME$||$SERVICEDESC$||$SERVIC
EOUTPUT$||$SERVICEPERFDATA$
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=30
service_perfdata_file_processing_command=process-service-perfdata-for-nagiosgraph

check_service_freshness=1
enable_flap_detection=1
enable_embedded_perl=1
enable_environment_macros=1
```

Configuring receipt of SNMP trap notifications for Nagios

1. On the monitored host, in Dr.Web SNMPD settings (the TrapReceiver parameter), specify an address to be listened by snmptrapd on the Nagios host, for example:

```
SNMPD.TrapReceiver = 10.20.30.40:162
```

2. Ensure the existence of the <*NAGIOS_PLUGINS_DIR*>/eventhandlers/submit_check_result script to be run when *SNMP trap* notifications are received. If such script does not exist, copy the submit_check_result file to this location from the directory /opt/drweb.com/share/drweb-snmpd/connectors/nagios/plugins/eventhandlers/ for GNU/Linux or /usr/local/libexec/drweb.com/share/drweb-snmpd/connectors/nagios/plugins/eventhandlers/ for FreeBSD. Change the path in this file; the path is specified in the CommandFile parameter. It must have the same value as the command_file parameter in the <*NAGIOS_ETC_DIR*>/nagios.cfg file.
3. Copy the snmptt.drweb.nagios.conf file to the /etc/snmp/snmp/ directory. Change the path to the submit_check_result script in this file, for example, by running the command:

```
$ awk '$1 == "EXEC" { $2 = <NAGIOS_PLUGINS_DIR>/eventhandlers/submit_check_result }
{ print}' ./snmp/snmptt.drweb.nagios.conf > /etc/snmp/snmp/snmptt.drweb.nagios.conf
```

4. Add the "/etc/snmp/snmptt.drweb.nagios.conf" line to the /etc/snmp/snmptt.ini file. After that, restart snmptt if it was started in daemon mode.

After all required Nagios configuration files are added and edited, start Nagios in debug mode by running the command:

```
# nagios -v <NAGIOS_ETC_DIR>/nagios.cfg
```

Here, Nagios will check for configuration errors. If no errors have been found, restart Nagios in the usual manner, for example, by running the command:

```
# service nagios restart
```

For official documentation on Nagios, follow the link.

9.14.5. Dr.Web SNMP MIB

The list of operating parameters of Dr.Web Gateway Security Suite that can be fetched by external monitoring systems via the SNMP protocol is provided in the table below.

Parameter name	OID parameter	Type and description of the parameter
Common prefix for all names: `.iso.org.dod.internet.private.enterprises.drweb.drwebSnmpd` Common OID prefix for all names: `.1.3.6.1.4.1.29690.2`		
alert	**Asynchronous notifications about events (SNMP traps)**	
threatAlert	`.1.1`	Notification about a detected threat
`threatAlertFile`	`.1.1.1`	Name of the infected file (string)
`threatAlertType`	`.1.1.2`	Threat type (integer*)
`threatAlertName`	`.1.1.3`	Name of the threat (string)
`threatAlertOrigin`	`.1.1.4`	Identifier of the component that detected the threat (integer***)
`threatAlertRemoteIp`	`.1.1.5`	IP address of the remote host that was used to access the file (string)
`threatAlertRemoteUser`	`.1.1.6`	Name of the remote user who accessed the file (string)
`threatAlertRemoteDomain`	`.1.1.7`	IP address of the remote host (FQDN) that was used to access the file (string)
threatActionErrorAlert	`.1.2`	Notification about an error which occurred when trying to neutralize the threat

Parameter name	OID parameter	Type and description of the parameter
threatActionErrorAlertFile	.1.2.1	Name of the infected file (string)
threatActionErrorAlertType	.1.2.2	Threat type (integer*)
threatActionErrorAlertName	.1.2.3	Name of the threat (string)
threatActionErrorAlertOrigin	.1.2.4	Identifier of the component that detected the threat (integer***)
threatActionErrorAlertError	.1.2.5	Description of an error (string)
threatActionErrorAlertErrorCode	.1.2.6	Last exit code (an integer corresponding to codes from the error catalog)
threatActionErrorAlertAction	.1.2.7	Failed action (integer: 1—cure; 2—quarantine; 3—delete; 4—report; 5—ignore)
componentFailureAlert	.1.3	Notification about a component failure
componentFailureAlertName	.1.3.1	Component identifier (integer***)
componentFailureAlertExitCodeDescription	.1.3.2	Component exit code description (string)
componentFailureAlertExitCode	.1.3.3	Last exit code (an integer corresponding to codes from the error catalog)
infectedUrlAlert	.1.4	Notification about blocking a malicious URL (for HTTP/HTTPS connections)
infectedUrlAlertUrl	.1.4.1	The URL that was blocked (string)
infectedUrlAlertDirection	.1.4.2	HTTP message direction (integer: 1—request, 2—response)
infectedUrlAlertType	.1.4.3	Threat type (integer*)
infectedUrlAlertName	.1.4.4	Name of the threat (string)
infectedUrlAlertOrigin	.1.4.5	Identifier of the component that detected the threat (integer***)

Parameter name	OID parameter	Type and description of the parameter
infectedUrlAlertSrcIp	.1.4.6	IP address of a connection source (string)
infectedUrlAlertSrcPort	.1.4.7	Port of the connection source (integer)
infectedUrlAlertDstIp	.1.4.8	IP address of a connection destination point (string)
infectedUrlAlertDstPort	.1.4.9	Port of the connection destination point (integer)
infectedUrlAlertSniHost	.1.4.10	SNI of the connection destination point (for SSL connections) (string)
infectedUrlAlertExePath	.1.4.11	Executable path of the program that established the connection (string)
infectedUrlAlertUserName	.1.4.12	Name of the user as whom the program establishing the connection is running (string)
categoryUrlAlert	.1.6	Notification about blocking a URL belonging to the unwanted category
categoryUrlAlertUrl	.1.6.1	The URL that was blocked (string)
categoryUrlAlertCategory	.1.6.2	Web resource category to which the URL belongs (integer**)
categoryUrlAlertOrigin	.1.6.3	Identifier of the component that detected the threat (integer***)
categoryUrlAlertSrcIp	.1.6.4	IP address of a connection source (string)
categoryUrlAlertSrcPort	.1.6.5	Port of the connection source (integer)
categoryUrlAlertDstIp	.1.6.6	IP address of a connection destination point (string)
categoryUrlAlertDstPort	.1.6.7	Port of the connection destination point (integer)

Parameter name	OID parameter	Type and description of the parameter
categoryUrlAlertSniHost	.1.6.8	SNI of the connection destination point (for SSL connections) (string)
categoryUrlAlertExePath	.1.6.9	Executable path of the program that established the connection (string)
categoryUrlAlertUserName	.1.6.10	Name of the user as whom the program establishing the connection is running (string)
blockedConnectionAlert	.1.9	Notification about blocking a network connection
blockedConnectionAlertOrigin	.1.9.1	Identifier of the component that detected the threat (integer***)
blockedConnectionAlertDivert	.1.9.2	Direction of the connection (integer: 1—incoming; 2—outgoing)
blockedConnectionAlertSrcIp	.1.9.3	IP address of a connection source (string)
blockedConnectionAlertSrcPort	.1.9.4	Port of the connection source (integer)
blockedConnectionAlertDstIp	.1.9.5	IP address of a connection destination point (string)
blockedConnectionAlertDstPort	.1.9.6	Port of the connection destination point (integer)
blockedConnectionAlertSniHost	.1.9.7	SNI of the connection destination point (for SSL connections) (string)
blockedConnectionAlertProtocol	.1.9.8	Protocol type (integer: 1—SMTP; 2—POP3; 3—IMAP; 4—HTTP)
blockedConnectionAlertExePath	.1.9.9	Executable path of the program that established the connection (string)
blockedConnectionAlertUserName	.1.9.10	Name of the user as whom the program establishing the connection is running (string)

Parameter name	OID parameter	Type and description of the parameter
expiringLicenseAlert	.1.10	Days before license expiration (integer)
outdatedBasesAlert	.1.11	Timestamp of the last successful attempt of updating databases (*Unix time*)
stat	**Statistics on the operation of Dr.Web Gateway Security Suite**	
threatCounters	.2.1	Counters of detected threats
knownVirus	.2.1.1	Number of detected known threats (counter; integer)
suspicious	.2.1.2	Number of detected suspicious objects (counter; integer)
adware	.2.1.3	Number of detected adware (counter; integer)
dialers	.2.1.4	Number of detected dialers (counter; integer)
joke	.2.1.5	Number of detected joke programs (counter; integer)
riskware	.2.1.6	Number of detected riskware (counter; integer)
hacktool	.2.1.7	Number of detected hacktools (counter; integer)
scanErrors	.2.2	Counters of the errors that occurred while files were scanned
sePathNotAbsolute	.2.2.1	Number of occurrences of the “Path is not absolute” error (counter; integer)
seFileNotFound	.2.2.2	Number of occurrences of the “File not found” error (counter; integer)
seFileNotRegular	.2.2.3	Number of occurrences of the “File is not a regular file” error (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
seFileNotBlockDevice	.2.2.4	Number of occurrences of the “File is not a block device” error (counter; integer)
seNameTooLong	.2.2.5	Number of occurrences of the “Path or file name is too long” error (counter; integer)
seNoAccess	.2.2.6	Number of occurrences of the “Permission denied” error (counter; integer)
seReadError	.2.2.7	Number of occurrences of “Read error” (counter; integer)
seWriteError	.2.2.8	Number of occurrences of “Write error” (counter; integer)
seFileTooLarge	.2.2.9	Number of occurrences of the “File size too big” error (counter; integer)
seFileBusy	.2.2.10	Number of occurrences of the “File is busy” error (counter; integer)
seUnpackingError	.2.2.20	Number of occurrences of “Unpacking error” (counter; integer)
sePasswordProtecetd	.2.2.21	Number of occurrences of the “Password protected” error (counter; integer)
seArchCrcError	.2.2.22	Number of occurrences of “Archive Cyclic Redundancy Check error” (counter; integer)
seArchInvalidHeader	.2.2.23	Number of occurrences of the “Invalid archive header” error (counter; integer)
seArchNoMemory	.2.2.24	Number of occurrences of the “Not enough memory to process archive” error (counter; integer)
seArchIncomplete	.2.2.25	Number of occurrences of the “Incomplete archive” error (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
seCanNotBeCured	.2.2.26	Number of occurrences of the “Object cannot be cured” error (counter; integer)
sePackerLevelLimit	.2.2.30	Number of occurrences of the “Packer nesting level limit reached” error (counter; integer)
seArchiveLevelLimit	.2.2.31	Number of occurrences of the “Archive nesting level limit reached” error (counter; integer)
seMailLevelLimit	.2.2.32	Number of occurrences of the “Mail nesting level limit reached” error (counter; integer)
seContainerLevelLimit	.2.2.33	Number of occurrences of the “Container nesting level limit reached” error (counter; integer)
seCompressionLimit	.2.2.34	Number of occurrences of the “Compression rate limit reached” error (counter; integer)
seReportSizeLimit	.2.2.35	Number of occurrences of the “Report size limit reached” error (counter; integer)
seScanTimeout	.2.2.40	Number of occurrences of the “Scan timeout expired” error (counter; integer)
seEngineCrash	.2.2.41	Number of occurrences of the “Engine crash detected” error (counter; integer)
seEngineHangup	.2.2.42	Number of occurrences of the “Engine hang-up detected” error (counter; integer)
seEngineError	.2.2.44	Number of occurrences of “Engine internal error” (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
seNoLicense	.2.2.45	Number of occurrences of the “No valid license” error (counter; integer)
seCuringLimitReached	.2.2.47	Number of occurrences of the “Curing limit reached” error (counter; integer)
seNonSupportedDisk	.2.2.50	Number of occurrences of the “Unsupported disk” error (counter; integer)
seUnexpectedError	.2.2.100	Number of occurrences of the “Unexpected error” (counter; integer)
scanLoadAverage	.2.3	Metrics of the file scanning load
filesScannedTable	.2.3.1	Average numbers of files scanned at the request of other components
filesScannedEntry	.2.3.1.1	Component (table row; record)
filesScannedIndex	.2.3.1.1.1	Index of the component (identifier; integer***)
filesScannedOrigin	.2.3.1.1.2	Name of the component
filesScanned1min	.2.3.1.1.3	Average (for a minute) number of files scanned per second (string)
filesScanned5min	.2.3.1.1.4	Average (for 5 minutes) number of files scanned per second (string)
filesScanned15min	.2.3.1.1.5	Average (for 15 minutes) number of files scanned per second (string)
bytesScannedTable	.2.3.2	Average speed (in bytes) of scanning performed at the request of other components
bytesScannedEntry	.2.3.2.1	Component (table row; record)
bytesScannedIndex	.2.3.2.1.1	Index of the component (identifier; integer***)

Parameter name	OID parameter	Type and description of the parameter
bytesScannedOrigin	.2.3.2.1.2	Name of the component
bytesScanned1min	.2.3.2.1.3	Average (for a minute) number of bytes scanned per second (string)
bytesScanned5min	.2.3.2.1.4	Average (for 5 minutes) number of bytes scanned per second (string)
bytesScanned15min	.2.3.2.1.5	Average (for 15 minutes) number of bytes scanned per second (string)
cacheHitFilesTable	.2.3.3	Cache usage for scanned files at the request of the components
cacheHitFilesEntry	.2.3.3.1	Component (table row; record)
cacheHitFilesIndex	.2.3.3.1.1	Index of the component (identifier; integer***)
cacheHitFilesOrigin	.2.3.3.1.2	Name of the component
cacheHitFiles1min	.2.3.3.1.3	The average (for a minute) number of reports retrieved from the cache per second (string)
cacheHitFiles5min	.2.3.3.1.4	Average (for 5 minutes) number of reports retrieved from cache per second (string)
cacheHitFiles15min	.2.3.3.1.5	Average (for 15 minutes) number of reports retrieved from cache per second (string)
errorsTable	.2.3.4	Number of errors during scanning performed at the request of the components
errorsEntry	.2.3.4.1	Component (table row; record)
errorsIndex	.2.3.4.1.1	Index of the component (identifier; integer***)
errorsOrigin	.2.3.4.1.2	Name of the component

Parameter name	OID parameter	Type and description of the parameter
errors1min	.2.3.4.1.3	Average (for a minute) number of scanning errors per second (string)
errors5min	.2.3.4.1.4	Average (for 5 minutes) number of scanning errors per second (string)
errors15min	.2.3.4.1.5	Average (for 15 minutes) number of scanning errors per second (string)
net	.2.4	Statistics on network activity
blockedInfectionSource	.2.4.101	Number of blocked URLs belonging to the “Infection Source” category (counter; integer)
blockedNotRecommended	.2.4.102	Number of blocked URLs belonging to the “Not Recommended” category (counter; integer)
blockedAdultContent	.2.4.103	Number of blocked URLs belonging to the “Adult Content” category (counter; integer)
blockedViolence	.2.4.104	Number of blocked URLs belonging to the “Violence” category (counter; integer)
blockedWeapons	.2.4.105	Number of blocked URLs belonging to the “Weapons” category (counter; integer)
blockedGambling	.2.4.106	Number of blocked URLs belonging to the “Gambling” category (counter; integer)
blockedDrugs	.2.4.107	Number of blocked URLs belonging to the “Drugs” category (counter; integer)
blockedObsceneLanguage	.2.4.108	Number of blocked URLs belonging to the “Obscene Language” category (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
blockedChats	.2.4.109	Number of blocked URLs belonging to the “Chats” category (counter; integer)
blockedTerrorism	.2.4.110	Number of blocked URLs belonging to the “Terrorism” category (counter; integer)
blockedFreeEmail	.2.4.111	Number of blocked URLs belonging to the “Free Email Services” category (counter; integer)
blockedSocialNetworks	.2.4.112	Number of blocked URLs belonging to the “Social Networks” category (counter; integer)
blockedOwnersNotice	.2.4.113	Number of blocked URLs belonging to the “Copyright Owner`s Notice” category (counter; integer)
blockedOnlineGames	.2.4.114	Number of blocked URLs belonging to the “Online games” category (counter; integer)
blockedAnonymizers	.2.4.115	Number of blocked URLs belonging to the “Anonymizers” category (counter; integer)
blockedCryptocurrencyMiningPools	.2.4.116	Number of blocked URLs belonging to the “Cryptocurrency mining pools” category (counter; integer)
blockedJobs	.2.4.117	Number of blocked URLs belonging to the "Job search websites" category (counter; integer)
blockedBlackList	.2.4.120	Number of blocked URLs from the user black list (counter; integer)
info	**Information about the current state of Dr.Web Gateway Security Suite**	

Parameter name	OID parameter	Type and description of the parameter
components	.3.1	Status of Dr.Web Gateway Security Suite Components
configd	.3.1.1	drweb-configd component data
configdState	.3.1.1.1	Component status (integer****)
configdExitCode	.3.1.1.2	Last exit code (an integer corresponding to codes from the error catalog)
configdExitTime	.3.1.1.3	Termination time (*Unix time*)
configdInstalledApps	.3.1.1.101	List of installed components
configdAppEntry	.3.1.1.101.1	Information about the installed component (table row; record)
configdAppIndex	.3.1.1.101.1.1	Index (number) of the installed component (integer)
configdAppName	.3.1.1.101.1.2	Name of the installed component (string)
configdAppExePath	.3.1.1.101.1.3	Component executable path (string)
configdAppInstallTime	.3.1.1.101.1.4	Component installation time (*Unix time*)
configdAppIniSection	.3.1.1.101.1.5	Name of the section with the component parameters in the configuration file (string)
scanEngine	.3.1.2	drweb-se component data
scanEngineState	.3.1.2.1	Component status (integer****)
scanEngineExitCode	.3.1.2.2	Last exit code (an integer corresponding to codes from the error catalog)
scanEngineExitTime	.3.1.2.3	Termination time (*Unix time*)
scanEngineStatus	.3.1.2.101	Dr.Web Virus-Finding Engine status (integer)

Parameter name	OID parameter	Type and description of the parameter
scanEngineVersion	.3.1.2.102	Dr.Web Virus-Finding Engine version (string)
scanEngineVirusRecords	.3.1.2.103	Number of threat records (integer)
scanEngineMaxForks	.3.1.2.104	Maximum number of child processes for scanning (integer)
scanEngineQueues	.3.1.2.105	Scan task queues
scanEngineQueuesLow	.3.1.2.105.1	The queue of low-priority tasks
scanEngineQueueLowOut	.3.1.2.105.1.1	Number of low-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueLowSize	.3.1.2.105.1.2	Number of low-priority tasks in the queue waiting to be processed (counter; integer)
scanEngineQueuesMedium	.3.1.2.105.2	The queue of normal-priority tasks
scanEngineQueueMediumOut	.3.1.2.105.2.1	Number of normal-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueMediumSize	.3.1.2.105.2.2	Number of normal-priority tasks in the queue waiting to be processed (counter; integer)
scanEngineQueuesHigh	.3.1.2.105.3	Queue of high-priority tasks
scanEngineQueueHighOut	.3.1.2.105.3.1	Number of high-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueHighSize	.3.1.2.105.3.2	Number of high-priority tasks in the queue waiting to be processed (counter; integer)
scanEngineVirusBasesTable	.3.1.2.106	The list of virus databases
scanEngineVirusBasesEntry	.3.1.2.106.1	Information about the virus database (table row; record)

Parameter name	OID parameter	Type and description of the parameter
scanEngineVirusBaseIndex	.3.1.2.106.1.1	Index of the virus database (integer)
scanEngineVirusBasePath	.3.1.2.106.1.2	Path to the virus database file (string)
scanEngineVirusBaseRecords	.3.1.2.106.1.3	Number of records in the virus database (integer)
scanEngineVirusBaseVersion	.3.1.2.106.1.4	Version of the virus database (integer)
scanEngineVirusBaseTimestamp	.3.1.2.106.1.5	Timestamp of the virus database (*Unix time*)
scanEngineVirusBaseMD5	.3.1.2.106.1.6	MD5 checksum (string)
scanEngineVirusBaseLoadResult	.3.1.2.106.1.7	Result of downloading the virus database (string)
scanEngineQueuesTab	.3.1.2.107	The list of scan task queues
scanEngineQueueEntry	.3.1.2.107.1	Information about the queue (table row; record)
scanEngineQueueIndex	.3.1.2.107.1.1	Index (number) of the queue (integer)
scanEngineQueueName	.3.1.2.107.1.2	Name of the queue (string)
scanEngineQueueOut	.3.1.2.107.1.3	Number of high-priority tasks popped from the queue and passed for processing (counter; integer)
scanEngineQueueSize	.3.1.2.107.1.4	Number of tasks in the queue waiting to be processed (counter; integer)
fileCheck	.3.1.3	drweb-filecheck component data
fileCheckState	.3.1.3.1	Component status (integer****)
fileCheckExitCode	.3.1.3.2	Last exit code (an integer corresponding to codes from the error catalog)
fileCheckExitTime	.3.1.3.3	Termination time (*Unix time*)

Parameter name	OID parameter	Type and description of the parameter
fileCheckScannedFiles	.3.1.3.101	Number of scanned files (counter; integer)
fileCheckScannedBytes	.3.1.3.102	Number of scanned bytes (counter; integer)
fileCheckCacheHitFiles	.3.1.3.103	Number of scan reports retrieved from the cache for scanned files (counter; integer)
fileCheckScanErrors	.3.1.3.104	Number of scanning engine errors (counter; integer)
fileCheckScanStat	.3.1.3.105	List of clients
fileCheckClientEntry	.3.1.3.105.1	Information about a client (table row; record)
fileCheckClientIndex	.3.1.3.105.1.1	Index (number) of the client (integer)
fileCheckClientName	.3.1.3.105.1.2	Name of the client component (string)
fileCheckClientScannedFiles	.3.1.3.105.1.3	Number of files scanned for this client (counter; integer)
fileCheckClientScannedBytes	.3.1.3.105.1.4	Number of bytes scanned for this client (counter; integer)
fileCheckClientCacheHitFiles	.3.1.3.105.1.5	Number of scan reports retrieved for this client from the cache for scanned files (counter; integer)
fileCheckClientScanErrors	.3.1.3.105.1.6	Number of scanning engine errors for this client (counter; integer)
update	.3.1.4	drweb-update component data
updateState	.3.1.4.1	Component status (integer****)
updateExitCode	.3.1.4.2	Last exit code (an integer corresponding to codes from the error catalog)
updateExitTime	.3.1.4.3	Termination time (*Unix time*)

Parameter name	OID parameter	Type and description of the parameter
updateBytesSent	.3.1.4.101	Number of sent bytes (counter; integer)
updateBytesReceived	.3.1.4.102	Number of received bytes (counter; integer)
esagent	.3.1.5	drweb-esagent component data
esagentState	.3.1.5.1	Component status (integer****)
esagentExitCode	.3.1.5.2	Last exit code (an integer corresponding to codes from the error catalog)
esagentExitTime	.3.1.5.3	Termination time (*Unix time*)
esagentWorkStatus	.3.1.5.101	Component operation mode (integer: 1—standalone mode, 2—connecting, 3—awaiting connection, 4—connection has been approved)
esagentIsConnected	.3.1.5.102	Connected to the server (integer: 0—no, 1—yes)
esagentServer	.3.1.5.103	Address of the centralized protection server in use (string)
netcheck	.3.1.6	drweb-netcheck component data
netcheckState	.3.1.6.1	Component status (integer****)
netcheckExitCode	.3.1.6.2	Last exit code (an integer corresponding to codes from the error catalog)
netcheckExitTime	.3.1.6.3	Termination time (*Unix time*)
netcheckLocalSeForks	.3.1.6.101	The number of scan engine processes available locally (integer)
netcheckRemoteSeForks	.3.1.6.102	Number of scan engine processes available remotely (integer)

Parameter name	OID parameter	Type and description of the parameter
netcheckLocalFilesScanned	.3.1.6.103	Number of files that have been scanned locally (counter; integer)
netcheckNetworkFilesScanned	.3.1.6.104	Number of files that have been scanned remotely (counter; integer)
netcheckLocalBytesScanned	.3.1.6.105	Number of bytes that have been scanned locally (counter; integer)
netcheckNetworkBytesScanned	.3.1.6.106	Number of bytes that have been scanned remotely (counter; integer)
netcheckLocalBytesIn	.3.1.6.107	Number of bytes received from local clients (counter; integer)
netcheckLocalBytesOut	.3.1.6.108	Number of bytes sent back to local clients (counter; integer)
netcheckNetworkBytesIn	.3.1.6.109	Number of bytes received from remote hosts (counter; integer)
netcheckNetworkBytesOut	.3.1.6.110	Number of bytes sent to remote hosts (counter; integer)
netcheckLocalScanErrors	.3.1.6.111	Number of error occurrences in local scan engine processes (counter; integer)
netcheckNetworkScanErrors	.3.1.6.112	Number of error occurrences in remote scan engine processes (counter; integer)
httpd	.3.1.7	`drweb-httpd` component data
httpdState	.3.1.7.1	Component status (integer****)
httpdExitCode	.3.1.7.2	Last exit code (an integer corresponding to codes from the error catalog)
httpdExitTime	.3.1.7.3	Termination time (*Unix time*)
snmpd	.3.1.8	`drweb-snmpd` component data
snmpdState	.3.1.8.1	Component status (integer****)

Parameter name	OID parameter	Type and description of the parameter
snmpdExitCode	.3.1.8.2	Last exit code (an integer corresponding to codes from the error catalog)
snmpdExitTime	.3.1.8.3	Termination time (*Unix time*)
statd	.3.1.9	`drweb-statd` component data
statdState	.3.1.9.1	Component status (integer****)
statdExitCode	.3.1.9.2	Last exit code (an integer corresponding to codes from the error catalog)
statdExitTime	.3.1.9.3	Termination time (*Unix time*)
statdConnectionsIn	.3.1.9.101	Number of accepted incoming connections (counter; integer)
statdConnectionsCount	.3.1.9.102	Number of currently established connections (counter; integer)
clamd	.3.1.20	`drweb-clamd` component data
clamdState	.3.1.20.1	Component status (integer****)
clamdExitCode	.3.1.20.2	Last exit code (an integer corresponding to codes from the error catalog)
clamdExitTime	.3.1.20.3	Termination time (*Unix time*)
icapd	.3.1.21	`drweb-icapd` component data
icapdState	.3.1.21.1	Component status (integer****)
icapdExitCode	.3.1.21.2	Last exit code (an integer corresponding to codes from the error catalog)
icapdExitTime	.3.1.21.3	Termination time (*Unix time*)
icapdConnectionsIn	.3.1.21.101	Number of accepted incoming connections (counter; integer)
icapdConnectionsCount	.3.1.21.102	Number of currently established connections (counter; integer)

Parameter name	OID parameter	Type and description of the parameter
icapdOptions	.3.1.21.103	Number of *OPTIONS* requests (counter; integer)
icapdReqmod	.3.1.21.104	Number of *REQMOD* requests (counter; integer)
icapdRespmod	.3.1.21.105	Number of *RESPMOD* requests (counter; integer)
icapdBad	.3.1.21.106	Number of invalid requests (counter; integer)
gated	.3.1.41	drweb-gated component data
gatedState	.3.1.41.1	Component status (integer****)
gatedExitCode	.3.1.41.2	Last exit code (an integer corresponding to codes from the error catalog)
gatedExitTime	.3.1.41.3	Termination time (*Unix time*)
gatedInterceptedIn	.3.1.41.101	Number of intercepted connections (counter; integer)
gatedInterceptedCount	.3.1.41.102	Number of currently monitored connections (counter; integer)
lookupd	.3.1.43	drweb-lookupd component data
lookupdState	.3.1.43.1	Component status (integer****)
lookupdExitCode	.3.1.43.2	Last exit code (an integer corresponding to codes from the error catalog)
lookupdExitTime	.3.1.43.3	Termination time (*Unix time*)
urlcheck	.3.1.45	drweb-urlcheck component data
urlcheckState	.3.1.45.1	Component status (integer****)
urlcheckExitCode	.3.1.45.2	Last exit code (an integer corresponding to codes from the error catalog)
urlcheckExitTime	.3.1.45.3	Termination time (*Unix time*)

Parameter name	OID parameter	Type and description of the parameter
urlcheckConnectionsIn	.3.1.45.101	Number of accepted incoming connections (counter; integer)
urlcheckConnectionsCount	.3.1.45.102	Number of currently established connections (counter; integer)
cloudd	.3.1.50	drweb-cloudd component data
clouddState	.3.1.50.1	Component status (integer****)
clouddExitCode	.3.1.50.2	Last exit code (an integer corresponding to codes from the error catalog)
clouddExitTime	.3.1.50.3	Termination time (*Unix time*)
meshd	.3.1.52	drweb-meshd component data
meshdState	.3.1.52.1	Component status (integer****)
meshdExitCode	.3.1.52.2	Last exit code (an integer corresponding to codes from the error catalog)
meshdExitTime	.3.1.52.3	Termination time (*Unix time*)
linuxfirewall	.3.1.203	drweb-firewall component data (for GNU/Linux)
linuxfirewallState	.3.1.203.1	Component status (integer****)
linuxfirewallExitCode	.3.1.203.2	Last exit code (an integer corresponding to codes from the error catalog)
linuxfirewallExitTime	.3.1.203.3	Termination time (*Unix time*)
ctl	.3.1.300	drweb-ctl component data
ctlState	.3.1.300.1	Component status (integer****)
ctlExitCode	.3.1.300.2	Last exit code (an integer corresponding to codes from the error catalog)
ctlExitTime	.3.1.300.3	Termination time (*Unix time*)
license	.3.2	License status

Parameter name	OID parameter	Type and description of the parameter
licenseEsMode	.3.2.1	The license has been granted by the centralized protection server (integer: 0—no, 1—yes)
licenseNumber	.3.2.2	License number (integer)
licenseOwner	.3.2.3	License owner (string)
licenseActivated	.3.2.4	License activation date (*Unix time*)
licenseExpires	.3.2.5	License expiration date (*Unix time*)

*) Threat types:

Code	Threat Type
1	Known threat (*known virus*)
2	Suspicious object (*suspicious*)
3	Adware (*adware*)
4	Dialer (*dialer*)
5	Joke (*joke program*)
6	Riskware (*riskware*)
7	Hacktool (*hacktool*)

**) URL categories:

Code	Threat Type
1	Infection source (*infectionSource*)
2	Not recommended (*notRecommended*)
3	Adult content (*adultContent*)
4	Violence (*violence*)
5	Weapons (*weapons*)
6	Gambling (*gambling*)

Code	Threat Type
7	Drugs (*drugs*)
8	Obscene language (*obsceneLanguage*)
9	Chats (*chats*)
10	Terrorism (*terrorism*)
11	Free email (*freeEmail*)
12	Social networks (*socialNetworks*)
13	URL added due to a notice from copyright owner (*ownerNotice*)
14	Online games (*onlineGames*)
15	Anonymizers (*anonymizers*)
16	Cryptocurrency mining pools (*cryptocurrencyMiningPools*)
17	Jobs (*Jobs*)
20	Black list (*blackList*)

***) Codes of Dr.Web components:

Code	Component
1	Dr.Web ConfigD (`drweb-configd`)
2	Dr.Web Scanning Engine (`drweb-se`)
3	Dr.Web File Checker (`drweb-filecheck`)
4	Dr.Web Updater (`drweb-update`)
5	Dr.Web ES Agent (`drweb-esagent`)
6	Dr.Web Network Checker (`drweb-netcheck`)
7	Dr.Web HTTPD (`drweb-httpd`)
8	Dr.Web SNMPD (`drweb-snmpd`)
20	Dr.Web ClamD (`drweb-clamd`)
21	Dr.Web ICAPD (`drweb-icapd`)
41	SpIDer Gate (`drweb-gated`)

Code	Component
43	Dr.Web LookupD (`drweb-lookupd`)
50	Dr.Web CloudD (`drweb-cloudd`)
52	Dr.Web MeshD (`drweb-meshd`)
203	Dr.Web Firewall for Linux (`drweb-firewall`)
300	Dr.Web Ctl (`drweb-ctl`)
400	Scanning tasked by a centralized protection server (not a component of Dr.Web Gateway Security Suite)

****) Possible states of the components:

Code	Status
0	Not installed
1	Installed but not started
2	Is starting
3	Is running
4	Is exiting

To get the values of the variables directly, use the `snmpwalk` utility:

```
$ snmpwalk -Os -c <community> -v <SNMP version> <host address> <OID>
```

For example, to get statistics about the threats detected on the local host, if Dr.Web SNMPD has default settings, run the command:

```
$ snmpwalk -Os -c public -v 2c 127.0.0.1 .1.3.6.1.4.1.29690.2.2.1
```

9.15. Dr.Web MeshD

The Dr.Web MeshD component is an agent that integrates a host on which Dr.Web Gateway Security Suite is installed with a "local cloud", which combines hosts with installed Dr.Web for Unix products. This cloud allows some cloud hosts to access the scanning engine of other cloud hosts, in other words, to receive file scanning services.

To combine hosts with Dr.Web for Unix products installed, the Dr.Web MeshD component, which integrates a host with the cloud, must be installed on every host. Host privileges within the cloud and cloud features used by a host are flexibly adjusted with Dr.Web MeshD settings.

Data is shared with other cloud hosts over a protected SSH channel.

9.15.1. Operating Principles

In this section

- Connection types
- Operation modes
- Services
  - Remote file scanning (Engine)
  - Sending files for scanning (File)
  - URL checking (Url)

Dr.Web MeshD is a mediator that ensures interaction of a host with Dr.Web Gateway Security Suite installed and other cloud hosts.

Connection types

Dr.Web MeshD uses the following connection types:

- *Client (service)*—used by Dr.Web MeshD to connect to other cloud hosts that are clients of services provided by the given host.

  Dr.Web Gateway Security Suite components operating on the host and using services provided by the cloud connect to Dr.Web MeshD, which operates on the same host, through a local Unix socket. At that, a client connection is not used.

- *Partner (peer to peer)*—used by Dr.Web MeshD for interaction with peer (within a service) partner cloud hosts. Usually such horizontal connections are used for scaling and distributing the load when interacting with the cloud, as well as for synchronization of cloud hosts.
- *Uplink*—used by Dr.Web MeshD for connecting this host as a client to cloud hosts that provide services (for example, sending files for scanning and so on).

The use of all three types of connections is configured for different cloud services independently from each other. At that, the same host can be configured as a server for processing client requests within one service (for example, checking URLs) and as a client within another service (for example, remote file scanning).

Within cloud, hosts perform authorized interaction via SSH, that is, all sides of interhost communication are always mutually authenticated. For the authentication, *host keys* are used in compliance with RFC 4251. A client connection from a local component is always considered as trusted.

Operation modes

Dr.Web MeshD can either operate in daemon mode or run at the request of other Dr.Web Gateway Security Suite components installed on the local host. If Dr.Web MeshD is configured to serve client connections (the `ListenAddress` parameter is not empty) and at least one of the services is activated, Dr.Web MeshD starts as a daemon and awaits client connections.

If Dr.Web MeshD is not set to process client connections (the `ListenAddress` parameter is empty) and there are no requests to this component during a time interval specified by the `IdleTimeLimit` parameter, the component shuts down automatically.

Services

Remote file scanning (Engine)

This service allows to use Dr.Web Scanning Engine for scanning remote files: hosts acting as clients send files for scanning to a server host, and server hosts provide a service for scanning files sent by the client hosts. Typical client settings are as follows:

```
…
[MeshD]
EngineChannel = On
EngineUplink = <server address>
ListenAddress =
…
```

The following settings are specified on the host acting as a local scanning server:

```
EngineChannel = On
EngineUplink =
ListenAddress = <address>:<port>
```

Here, <*server address*> in the uplink connection of the client must refer to the <*address*> and <*port*> that are used by the server host for managing client connections.

Sending files for scanning (File)

This feature is not used (remote scanning is provided by the *Engine* service).

URL checking (Url)

This service allows to check whether a URL belongs to potentially dangerous and unwanted categories: client hosts send a URL to be checked to a server host. Typical client settings are as follows:

```
…
[MeshD]
UrlChannel = On
UrlUplink = <server address>
ListenAddress =
…
```

The following settings are specified on the host acting as a URL checking server:

```
UrlChannel = On
UrlUplink =
ListenAddress = <address>:<port>
```

Here, *<server address>* in the uplink connection of the client must refer to the *<address>* and *<port>* that are used by the server host for managing client connections.

9.15.2. Command-Line Arguments

To start the Dr.Web MeshD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-meshd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-meshd [<parameters>]
```

Dr.Web MeshD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v`

	Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-meshd --help
```

This command outputs short help information about the Dr.Web MeshD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-meshd` command.

9.15.3. Configuration Parameters

The component uses configuration parameters specified in the `[MeshD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-meshd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-meshd`

Parameter	Description
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`
`DebugSsh` *{boolean}*	Log or do not log SSH messages (messages and data are transmitted via SSH) received and sent by the Dr.Web MeshD component operating on the host, if the logging level is `LogLevel = Debug`. Default value: `No`
`ListenAddress` *<IP address>:<port>*	Network socket (address and port) of the client connection where the component is waiting to receive connections from cloud hosts. These hosts are clients of services provided by this cloud host. In order for the component to listen on the IPv6 interface and detect cloud client hosts via IPv6, this parameter must be defined. If the value is not specified, the component does not receive requests from clients. Default value: `0.0.0.0:7090` if the `drweb-scanning-server` package is installed, not specified otherwise
`DnsResolverConfPath` *{path to file}*	Path to a DNS resolver configuration file. Default value: `/etc/resolv.conf`
`DiscoveryResponderPort` *{port number}*	Port on which a higher-level MeshD host responds to client requests via UDP. The *discovery* mechanism is enabled only if the `ListenAddress` parameter is defined. Default value: `18008`
`FileChannel` *{On | Off}*	Allow or do not allow the Dr.Web MeshD component operating on this host to participate in file exchange services. If the parameter is set to `On`, the Dr.Web MeshD component is automatically started by the Dr.Web ConfigD configuration daemon. Default value: `On`
`FileUplink` *{address}*	Address of a higher-level host of Dr.Web MeshD receiving files from this host for scanning. Allowed values: • *value is not specified*—server is not specified for this service, and Dr.Web MeshD will not establish connections.

Parameter	Description
	• *<IP address>:<port>*—Dr.Web MeshD will connect to a server with the specified address and port. • `dns:<service name>[:<domain>]`—address and port of the server are determined by searching for the SRV record of the DNS domain *<domain>*. If *<domain>* is not specified, the domain from the DNS resolver configuration file is used (the path is specified by `ResolverConfPath`). The domain is taken from the `search` and `domain` fields, depending on which of them is encountered last in the configuration file. • `discover`—search for a higher-level host using the *discovery* mechanism. Default value: *(not specified)*
`FileDebugIpc` *{boolean}*	Log or do not log the debug information for the file exchange service if the logging level is `LogLevel = Debug`. Default value: `No`
`EngineChannel` *{On | Off}*	Allow or do not allow the Dr.Web MeshD component operating on this host to provide scanning engine services. If the parameter is set to `On`, the Dr.Web MeshD component is automatically started by the Dr.Web ConfigD configuration daemon. Default value: `On`
`EngineUplink` *{address}*	Address of a higher-level host of Dr.Web MeshD providing scanning engine services for this host. Allowed values: • *value is not specified*—server is not specified for this service, and Dr.Web MeshD will not establish connections. • *<IP address>:<port>*—Dr.Web MeshD will connect to a server with the specified address and port. • `dns:<service name>[:<domain>]`—address and port of the server are determined by searching for the SRV record of the DNS domain *<domain>*. If *<domain>* is not specified, the domain from the DNS resolver configuration file is used (the path is specified by `ResolverConfPath`). The domain is taken from the `search` and `domain` fields, depending on which of them is encountered last in the configuration file. • `discover`—search for a higher-level host using the *discovery* mechanism. Default value: *(not specified)*
`EngineDebugIpc` *{boolean}*	Log or do not log the debug information for the file exchange service if the logging level is `LogLevel = Debug`. Default value: `No`

Parameter	Description
`UrlChannel` *{On | Off}*	Allow or do not allow the Dr.Web MeshD component operating on this host to provide URL checking services. Default value: `On`
`UrlUplink` *{address}*	Address of a higher-level host of Dr.Web MeshD providing URL checking services for this host. Allowed values: • *value is not specified*—URL checking server is not specified. • *<IP address>:<port>*—Dr.Web MeshD will connect to a server with the specified address and port. • `dns:`*<service name>*`[:`*<domain>*`]`—address and port of the server are determined by searching for the SRV record of the DNS domain *<domain>*. If *<domain>* is not specified, the domain from the DNS resolver configuration file is used (the path is specified by `ResolverConfPath`). The domain is taken from the `search` and `domain` fields, depending on which of them is encountered last in the configuration file. • `discover`—search for a higher-level host using the *discovery* mechanism. Default value: *(not specified)*
`UrlDebugIpc` *{boolean}*	Log or do not log the debug information for the URL checking service if the logging level is `LogLevel = Debug`. Default value: `No`

The current version of Dr.Web Gateway Security Suite does not use the *File* service for sending files for scanning. Use the *Engine* service of the scanning engine instead.

9.16. Dr.Web URL Checker

Dr.Web URL Checker is an auxiliary component for checking whether URLs refer to malicious or unwanted web resources.

Dr.Web URL Checker is used by the following components:

- Dr.Web HTTPD,
- Dr.Web MeshD,
- SpIDer Gate,
- Dr.Web ICAPD

9.16.1. Operating Principles

The Dr.Web URL Checker component is designed to check URLs for belonging to unwanted or potentially dangerous categories.

The check can be performed either by using specialized link bases or by using the Dr.Web CloudD service. To use the Dr.Web CloudD service, run the command:

```
# drweb-ctl cfset Root.UseCloud Yes
```

Dr.Web URL Checker cannot be started by the user. This component is started by the Dr.Web ConfigD configuration management daemon upon request of other components.

9.16.2. Command-Line Arguments

To start the Dr.Web URL Checker component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-urlcheck [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-urlcheck [<parameters>]
```

Dr.Web URL Checker accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-urlcheck --help
```

This command outputs short help information about the Dr.Web URL Checker component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-urlcheck` command.

9.16.3. Configuration Parameters

The component uses configuration parameters specified in the `[Urlcheck]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-urlcheck` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-urlcheck`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `None` if the `drweb-scanning-server` package is installed, `10m` otherwise
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of

Parameter	Description
	numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`

9.17. Dr.Web CloudD

In this section

- Operating principles
- Using the Dr.Web Cloud service
- Command-line arguments
- Configuration parameters

Operating principles

The Dr.Web CloudD component is designed to communicate with the Dr.Web Cloud service of the Doctor Web company. The Dr.Web Cloud service collects up-to-date information from all Dr.Web anti-virus products, which makes it possible to:

- neutralize latest threats that are not yet covered by virus databases;
- prevent users from visiting unwanted websites that are not yet covered by databases of web resource categories;
- reduce a number of false positives of Dr.Web Scanning Engine and of the components monitoring internet access.

If the component is enabled, it will periodically send statistics on detection of infected files to the Dr.Web Cloud service.



All data sent by the component to Doctor Web servers is anonymized and cannot be used to identify the user.

Dr.Web Gateway Security Suite is not connected to the Dr.Web Cloud service by default.

The Dr.Web Cloud service is used as an additional security layer while scanning network traffic and URLs.

Using the Dr.Web Cloud service

Dr.Web Gateway Security Suite is connected to the Dr.Web Cloud service automatically when the Dr.Web CloudD component is enabled.

You can enable or disable the Dr.Web CloudD component using:

- Dr.Web Ctl tool;
- management web interface;
- or edit the configuration file manually.

Enable or disable the component with the Dr.Web Ctl tool

To enable Dr.Web CloudD, run the command:

```
# drweb-ctl cfset Root.UseCloud Yes
```

To disable this component, run the command:

```
# drweb-ctl cfset Root.UseCloud No
```

Enable or disable the component with the management web interface (as in the case of a local computer)

1. Start a web browser and enter the address `https://127.0.0.1:4443/`.
2. Provide the name and password of a user with administrative privileges.
3. Select **Settings** and then **General Settings**.
4. Select the check box opposite the **UseCloud** text to enable the component; otherwise, clear this check box.

Details on configuring Dr.Web Gateway Security Suite using the management web interface.

Enable or disable by editing the configuration file manually



You must be sure that the changes introduced to the configuration file are correct.

1. Open the configuration file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD in a preferred text editor with administrative privileges. To gain administrative privileges, use the `su` command to change the user or the `sudo` command to run the command as another user.
2. If the configuration file does not contain the `[Root]` section, go to the end of the file and add the following in the new line:
   - To enable the component and connect to the service:

     ```
     [Root]
     UseCloud = Yes
     ```

   - To disable the component and disconnect from the service:

     ```
     [Root]
     UseCloud = No
     ```

   If the configuration file already contains the `[Root]` section:
   - If the `UseCloud` parameter is absent, add it by indicating `UseCloud = Yes` to connect to the service or `UseCloud = No` to disconnect from it.
   - If the `[Root]` section already contains the `UseCloud` parameter, change its value to `Yes` to connect to the service or to `No` to disconnect from it.

3. Reload the Dr.Web Gateway Security Suite configuration by running the command:

```
# drweb-ctl reload
```

to apply changes.

The Dr.Web CloudD component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To get documentation on this component from the command line, run the `man 1 drweb-cloudd` command.

Command-line arguments

To show help for the Dr.Web CloudD component, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-cloudd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-cloudd [<parameters>]
```

`drweb-cloudd` accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-cloudd --help
```

This command outputs short help information about the Dr.Web CloudD component.

Configuration parameters

The Dr.Web CloudD component uses configuration parameters specified in the `[CloudD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

To change the value of a parameter of Dr.Web CloudD, run the command:

```
# drweb-ctl cfset CloudD.<parameter> <value>
```

The `[CloudD]` section contains the following parameters:

Parameter	Allowed values	Description
`LogLevel`	*Debug, Info, Notice, Warning or Error*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log`	*Auto, Syslog[:<Daemon, User, Mail or Local0..7>] or <path to file>*	Logging method of the component. Default value: `Auto`
`ExePath`	*path to file*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-cloudd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-cloudd`
`RunAsUser`	*<UID> or <username>*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`IdleTimeLimit`	*time interval*	Maximum idle time for the component. When the specified period of time expires, the component shuts down.

Parameter	Allowed values	Description
		Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `1h`

`FixedSocketPath`	*path to file*	Path to the Unix socket file of the component fixed instance.
		If this parameter is specified, the Dr.Web ConfigD configuration management daemon ensures that there is always a running component instance available to clients via this socket.
		⚠ Ensure that the following permissions are assigned to the directory with the socket file:
		`drwxr-xr-x`
		To view the permissions, run the command:
		`$ ls -ld <path to directory>`
		Default value: *(not specified)*

`PersistentCache`	*On, Yes, True, Off, No or False*	Enable or disable saving of the cache of Dr.Web Cloud responses to the disk. Default value: `Off`
`DebugSdk`	*On, Yes, True, Off, No or False*	Log or do not log detailed messages from Dr.Web Cloud at the debug level (with `LogLevel = DEBUG`). Default value: `No`

9.18. Dr.Web LookupD

The Dr.Web LookupD component is designed to query data from external storage units (text files, relational databases, directory services supporting LDAP). The data retrieved by the component can be used to scan network connections (for example, to check user privileges and authorization), block access to a URL or an email message if a certain condition is observed.

In the component settings, you can specify parameters for connection to several data sources. The Dr.Web LookupD component connects to a required data source only upon receiving a data request from a Dr.Web Gateway Security Suite component.

The Dr.Web LookupD component can query the following data sources:

- text files (in *AllMatch*, *Mask*, *Regex* or *Cidr* modes);
- relational databases (MySQL, PostgreSQL or SQLite);
- Redis databases;
- directory services (Active Directory and others that provide access via LDAP, for example, OpenLDAP).

Data can be shared via LDAP either over an insecure channel or over a secure channel using SSL/TLS. To use a secure connection, provide Dr.Web LookupD with a valid SSL certificate and key. If you need to generate keys and certificates, you can use the `openssl` utility. An example of how to use the `openssl` utility to generate a certificate and a private key is given in the Appendix E. Generating SSL Certificates section.

9.18.1. Operating Principles

In this section

- Aspects of processing text files
- MySQL connection aspects

The component is designed to request data from text files, relational databases, network storage and directory services (such as Active Directory) that support LDAP. The received data (for example, user identifiers and rights) is transferred to Dr.Web Gateway Security Suite components that requested it to be used for various scanning rules (for example, to allow the user to access a requested URL and so on).



This manual does not cover the operating principles of relational databases, a network Redis database, directory services and LDAP. If necessary, refer to the corresponding reference materials.

The Dr.Web LookupD component is started automatically by the Dr.Web ConfigD configuration daemon when required (upon receiving a request for fetching of data from storage).

Having received a request for data from a certain component of Dr.Web Gateway Security Suite, the Dr.Web ConfigD configuration daemon starts Dr.Web LookupD (if not started), then the component performs the request from a requested data source and returns a response. Depending on the request, the response is either a list of strings retrieved from the data source according to a given search criteria, or a boolean value (true or false) that indicates whether the search results contain strings that match the given condition.

You can specify an unlimited number of data sources in Dr.Web LookupD settings. When generating a request for data retrieval, the client component of Dr.Web Gateway Security Suite must specify the source of data. Once Dr.Web LookupD is started and the request is performed, it continues to operate for some time waiting for new requests. If there are no more requests, Dr.Web LookupD shuts down automatically after a delay.

Other components of Dr.Web Gateway Security Suite basically use Dr.Web LookupD for retrieving data to check the validity of custom conditions specified in the operation rules for these components. When checking the applicability of rules and the validity of such conditions, data requests to Dr.Web LookupD are performed automatically.

Aspects of processing text files

1. When processing text files, leading and trailing spaces in strings are discarded. Blank strings and strings that have the `#` character as the first non-whitespace character are ignored.
2. Text files are considered immutable data sources and their content is fully cached in memory. In addition, the results of requests to these files for value validation are also cached. Thus, if the source file has been modified, Dr.Web LookupD must re-read the configuration, which can be done by sending a HUP signal to the Dr.Web ConfigD component, for example, by running the `drweb-ctl reload` command.

MySQL connection aspects

Before establishing a connection to MySQL, the parameters from the `[client]` section of the MySQL default settings file are read (the file search is done for the following paths: `/etc/my.cnf`, `/etc/mysql/my.cnf` and `/etc/alternatives/my.cnf`).

9.18.2. Command-Line Arguments

To start the Dr.Web LookupD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-lookupd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-lookupd [<parameters>]
```

Dr.Web LookupD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-lookupd --help
```

This command outputs short help information about the Dr.Web LookupD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-lookupd` command.

9.18.3. Configuration Parameters

In this section

- Component parameters
  - URI requirements for a database connection
- Data source sections
  - Parameters used in the LDAP section
  - Parameters used in the AD section
  - Parameters used in the AllMatch, Mask, Regex and Cidr sections
  - Parameters used in the Pq, Mysql and Sqlite sections
  - Parameters used in the Redis section

- Adding sections for new data sources

The component uses configuration parameters specified in the `[LookupD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

Component parameters

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-lookupd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-lookupd`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`
`DebugLibldap` *{boolean}*	Log or do not log debug messages of the `libldap` library at the debug level (with `LogLevel = DEBUG`). Default value: `No`

Parameter	Description
`LdapCheckCertificate` *{No | Allow | Try | Yes}*	The mode of certificate verification for LDAP connections via SSL/TLS. Allowed values: • `No`—never request a server certificate. • `Allow`—request the server certificate. If the certificate is not provided, the session will continue normally. If the server certificate is provided but cannot be verified (the corresponding root certificate was not found), the server certificate will be ignored and the session will continue normally. • `Try`—request the server certificate. If the certificate is not provided, the session will continue normally. If the server certificate is provided but cannot be verified (the corresponding root certificate was not found), the session will be terminated. • `Yes`—request the server certificate. If the certificate is not provided or cannot be verified (the corresponding root certificate was not found), the session will be terminated. For `LDAP` data sources this certificate verification mode influences the way the URL is processed when the `ldaps://` scheme or the StartTLS extension is used. For `AD` data sources it influences the server connection, if `UseSSL=Yes` is specified in the corresponding section (see below). Default value: `Yes`
`LdapCaPath` *{path}*	Path to the directory or file with a list of root certificates which are trusted for sharing data using the LDAP protocol via SSL/TLS. Default value: <*path to the system list of trusted certificates*>. The path depends on your GNU/Linux distribution. • For Astra Linux, Debian, Linux Mint, SUSE Linux and Ubuntu this is usually the path `/etc/ssl/certs/`. • For CentOS and Fedora—the path `/etc/pki/tls/certs/ca-bundle.crt`. • For other distributions, the path can be determined by running the `openssl version -d` command. • If the command is unavailable or your OS distribution cannot be identified, the `/etc/ssl/certs/` value is used.
`LdapCertificatePath` *{path to file}*	Path to the SSL certificate used for secure connection to LDAP servers (Active Directory) via SSL/TLS. The certificate file and the private key file (specified by the `LdapKeyPath` parameter) must match each other. Default value: *(not specified)*

Parameter	Description
`LdapKeyPath` *{path to file}*	Path to the private key used for secure connection to LDAP servers (Active Directory) via SSL/TLS.  The certificate file (specified by the `LdapCertificatePath` parameter) and the private key file must match each other. Default value: *(not specified)*
`PqDefaultConn` *{URL}*	URI that defines default parameters for connecting to a PostgreSQL database. Allowed values: • `tcp://[<user>[:<password>]@][<host>][:<port>][/<database name>][?<parameter>=<value>[&...]]`; • `unix://[<user>[:<password>]@]<socket path>[:<database name>][?<parameter>=<value>[&...]]`.  Take notice of the URI requirements. Default value: *(not specified)*
`SqliteDefaultConn` *{path to file}*	Path to a default SQLite database file (specify the `file://` scheme prefix). Default value: *(not specified)*
`MysqlDefaultConn` *{URL}*	URI that defines default parameters for connecting to a MySQL database. Allowed values: • `tcp://[<user>[:<password>]@][<host>][:<port>][/<database name>][?<parameter>=<value>[&...]]`; • `unix://[<user>[:<password>]@]<socket path>[:<database name>][?<parameter>=<value>[&...]]`.  Take notice of the URI requirements. Default value: *(not specified)*
`RedisDefaultConn` *{URL}*	URL that defines default parameters for connecting to a Redis database. Allowed values: • `tcp://[<password>@][<host>][:<port>][/<database index>];` • `unix://[<password>@]<socket path>[:<database index>]`.

Parameter	Description
	Take notice of the URI requirements. Default value: *(not specified)*
`RedisCaPath` *{path to file}*	Path to the directory or file with a list of root certificates that are trusted for sharing data with a Redis database via SSL/TLS. Default value: *<path to the system list of trusted certificates>*. The path depends on your GNU/Linux distribution. • For Astra Linux, Debian, Linux Mint, SUSE Linux and Ubuntu this is usually the path `/etc/ssl/certs/`. • For CentOS and Fedora—the path `/etc/pki/tls/certs/ca-bundle.crt`. • For other distributions, the path can be determined by running the `openssl version -d` command. • If the command is unavailable or your OS distribution cannot be identified, the `/etc/ssl/certs/` value is used.
`RedisCertificatePath` *{path to file}*	Path to the SSL certificate used for secure connection to a Redis database via SSL/TLS. The certificate file and the private key file (specified by the `RedisKeyPath` parameter) must match each other. Default value: *(not specified)*
`RedisKeyPath` *{path to file}*	Path to the private key used for secure connection to a Redis database via SSL/TLS. The certificate file (specified by the `RedisCertificatePath` parameter) and the private key file must match each other. Default value: *(not specified)*
`DbIdleTimeout` *{time interval}*	The timeout period at the end of which an established connection to a Redis database will be closed if it is idle. Default value: `5m`

URI requirements for a database connection

1. Only the `tcp://` and `unix://` scheme prefixes (for local Unix sockets) are used. Database-specific prefixes, such as `postgresql://` or `mysql://`, are not supported. Path to an SQLite database file is specified with the `file://` scheme prefix.

2. If the <*host*> field is not specified in the URI or the `localhost` host is specified, the `127.0.0.1` host address is substituted. In this case by default the connection to MySQL and PostgreSQL databases is established *using a local Unix socket* despite that a network connection is specified.
3. If URI fields (such as <*user*>, <*password*>, <*database name*> and so on) or the connection parameter string contain special characters (a space, a colon and so on), use hex coding, for example:
   - space—%20;
   - :—%3A;
   - /—%2F;
   - @—%40;
   - %—%25.
4. For MySQL, the connection parameter string can include only the following parameters:

Parameter name	Convention in database documentation	Type	Description
`init`	*MYSQL_INIT_COMMAND*	String	SQL command to be run after connecting to the database
`compression`	*MYSQL_OPT_COMPRESS*	Boolean	Compress or do not compress data in transit
`connect-timeout`	*MYSQL_OPT_CONNECT_TIMEOUT*	Integer	Time-out for disconnecting an unused connection in seconds
`reconnect`	*MYSQL_OPT_RECONNECT*	Boolean	Allow or deny automatic reconnection
`read-timeout`	*MYSQL_OPT_READ_TIMEOUT*	Integer	Time-out for receiving packets from the server in seconds
`write-timeout`	*MYSQL_OPT_WRITE_TIMEOUT*	Integer	Time-out for sending packets to the server in seconds
`charset`	*MYSQL_SET_CHARSET_NAME*	String	Character encoding used for the default connection

Parameter name	Convention in database documentation	Type	Description
`plugin-dir`	*MYSQL_PLUGIN_DIR*	String	Path to the server directory storing plug-ins
`nonblock`	*MYSQL_OPT_NONBLOCK*	Integer	Stack size for non-blocking I/O operations
`ssl-key`	*MYSQL_OPT_SSL_KEY*	String	Path to a private key (in the PEM format) used to establish a secure connection
`ssl-cert`	*MYSQL_OPT_SSL_CERT*	String	Path to a public key certificate (in the PEM format) used to establish a secure connection
`ssl-ca`	*MYSQL_OPT_SSL_CA*	String	Path to a file (in the PEM format) containing trusted CA certificates
`ssl-capath`	*MYSQL_OPT_SSL_CAPATH*	String	Path to a directory containing trusted CA certificates (in the PEM format)
`ssl-cipher`	*MYSQL_OPT_SSL_CIPHER*	String	List of supported encryption algorithms for a secure connection
`ssl-crl`	*MYSQL_OPT_SSL_CRL*	String	Path to a file (in the PEM format) containing revoked certificates
`ssl-crlpath`	*MYSQL_OPT_SSL_CRLPATH*	String	Path to a directory containing revoked certificates (in the PEM format)
`ssl-fp`	*MARIADB_OPT_SSL_FP*	String	SHA1 hash of a trusted server certificate

Parameter name	Convention in database documentation	Type	Description
`ssl-fp-list`	*MARIADB_OPT_SSL_FP_LIST*	String	Path to a file containing SHA1 hashes of tursted server certificates
`tls-passphrase`	*MARIADB_OPT_TLS_PASSPHRASE*	String	Password for a password-protected client private key
`tls-version`	*MARIADB_OPT_TLS_VERSION*	String	List of supported TLS versions
`server-verify-cert`	*MYSQL_OPT_SSL_VERIFY_SERVER_CERT*	Boolean	Allow or deny verification of server certificates
`server-public-key-path`	*MYSQL_SERVER_PUBLIC_KEY*	String	Path to a file (in the PEM format) containing an RSA public key of the server

Read more on these parameters in the database documentation.

5. Parameter key words for the PostgreSQL DBMS.

Data source sections

In addition to the general section `[LookupD]`, the configuration file also contains sections that describe connections to data sources (one section for each connection). These sections are named using the following scheme: `[LookupD.<type>.<name>]`, where:

- <*type*>—connection type:
  - `LDAP`—for a directory service using LDAP;
  - `AD`—for a directory service using Active Directory;
  - `AllMatch`—for a text file in the *AllMatch* mode (full match);
  - `Mask`—for a text file in the *Mask* mode (mask matching);
  - `Regex`—for a text file in the *Regex* mode (matching a regular expression in the PCRE standard);
  - `Cidr`—for a text file in the *Cidr* mode (matching IP addresses and/or ranges);
  - `Pq`—for a PostgreSQL database;
  - `Mysql`—for a MySQL database;
  - `Sqlite`—for an SQLite database;

  - `Redis`—for a Redis database.
- <*name*>—a unique identifier (tag) for the connection, by which the connection can be referred to from the rules.

Example: `[LookupD.LDAP.auth1]`.

Connection-specific sections comprise a set of parameters depending on a source type. The number of sections is unlimited.

Parameters used in the LDAP section

Parameter	Description
`Url` *{string}*	URL that defines the used LDAP server and extracted data. In accordance with RFC 4516, the URL is constructed on the basis of the following scheme: *<scheme>*`://`*<host>*`[:`*<port>*`]/`*<dn>*`[?`*<attrs>*`[?`*<scope>*`[?`*<filter>*`[?`*<extensions>*`]]]]` Where: <*scheme*>—method of connection to the server (the following schemes are allowed: `ldap`, `ldaps` and `ldapi`); <*host*>`[:`<*port*>`]`—LDAP server address at which the request is sent; <*dn*>—distinguished name of an object, information about which is received; <*attrs*>—names of record attributes, the values of which must be received in the request; <*scope*>—search scope (`base`, `one`, `sub`); <*filter*>—filtering condition for the values of extracted attributes; <*extensions*>—list of LDAP extensions used in the request. **Features** • In the list of attributes <*attrs*>, it is possible to use special selection characters `*`, `+` and `1.1`. • The following automatically resolved markers can be used in the <*dn*> and <*filter*> parts of the URLs: ▫ `$u` is automatically replaced with `user`—the user name sent by a client component; ▫ `$d` is automatically replaced with `domain`—the domain sent by the client component; ▫ `$D`—a chain of <*subdomain*>.<*domain*> transformed into `dc=`<*subdomain*>`,dc=`<*domain*>; ▫ `$$`—the `$` character. • If the <*filter*> condition requires using special characters (for example: `*`, `(`, `)`, `\`, the character with code 0) as usual ones, they

	should be formatted as `\XX`. In addition, special characters in LDAP URLs are encoded using `%XX` sequences. For example, when using the `/` character in a URL in the `ldapi` scheme as a part of the path to a local LDAP server socket, this character is encoded as `%2f`.
	• As allowed extensions in <*extensions*>, only `StartTLS` and `1.3.6.1.4.1.1466.20037` are supported, they use TLS (i.e. a secure connection with an LDAP server is established, even if the use of the `ldaps` secure scheme is not explicitly indicated). If the name of the used extension is preceded by the `!` character, then the use of TLS *is required*, i.e. in case it is impossible to establish a secure connection, the request *will not* be performed. Otherwise, the request will be performed even if a secure connection is not established.
	These extensions cannot be used with the secure `ldaps` scheme. For more information, refer to RFC 4516 or `man ldap_search_ext_s`.
	Examples:
	`ldaps://ds.example.com:990/$D?givenName,sn,cn?sub?(uid=$u)`
	`ldap://ldap.local/o=org,dc=nodomain?ipNetworkNumber?sub?(objectClass=ipNetwork)?!StartTLS`
	Default value: *(not specified)*

`BindDn` *{string}*	Object in the LDAP directory with which the user is associated to get authorization. Example: `cn=admin,dc=nodomain` Default value: *(not specified)*
`BindPassword` *{string}*	Password for authentication on the LDAP server. Default value: *(not specified)*
`ChaseReferrals` *{boolean}*	Follow or do not follow references to other LDAP servers, if the current LDAP server returns them in response to requests. Default value: `No`

Parameters used in the AD section

Parameter	Description
`Host` *{string}*	Domain name (FQDN) or IP address of a host running the server of the Active Directory service to be connected to. Example: `win2012.win.local` Default value: *(not specified)*

`Port` *{integer}*	Port on the host which is listened to by the server of the Active Directory service. Default value: `389`
`Dn` *{string}*	*DN* of an object in the Active Directory (similar to the `dn` part of the LDAP URL). Example: `dc=win,dc=local` Default value: *(not specified)*
`User` *{string}*	Full identifier of a user on the server. Example: `Administrator@WIN.LOCAL` Default value: *(not specified)*
`Password` *{string}*	Password for authentication on the Active Directory server. Default value: *(not specified)*
`ChaseReferrals` *{boolean}*	Follow or do not follow references to other LDAP servers, if the current Active Directory server returns them in response to requests. Default value: `No`
`UseSSL` *{boolean}*	Use or do not use SSL/TLS for connecting to an Active Directory server. Default value: `No`

Parameters used in the AllMatch, Mask, Regex and Cidr sections

Parameter	Description
`File` *{path}*	Path to a text file containing search strings. Example: `/etc/file1` Default value: *(not specified)*

Features

- Strings from a file specified in the section of the `AllMatch` type are used in a case-insensitive search to find a full match.
- Strings of a file specified in the section of the `Mask` type are treated as masks (*wildcards*). Masks can be considered a simplified version of regular expressions containing special characters besides standard ones. Comparing the strings with the masks is case-insensitive. Masks can contain the following special characters and expressions:

  `*`—any character sequence;

  `?`—any character;

  `[`<*character set*>`]`—any character from the set (for example, `[bac]`);

`[!`<*character set*>`]`—any character that does not match any character from the set (for example, `[!cab]`);

`[[:`<*class*>`:]]`—any character from the provided POSIX character class (*alnum*, *alpha*, *ascii*, *blank*, *cntrl*, *digit*, *graph*, *lower*, *print*, *punct*, *space*, *upper*, *xdigit*).

A mask matching some substring must cover a search substring surrounded by `*` characters (e.g., `*host*`). If you need to specify one of special characters, you need to escape such character with the backslash: `\[`, `\]`, `\*`, `\?`. If needed, the backslash can be escaped as well: `\\`. There is no need to escape any other characters, e.g. the `\a\b\c\*\d\?\\` string will be converted to the `abc*d?\` string. Examples of masks:

```
#Matches the name string exactly
name

#Matches three-character strings where
#the first character is c, the second is any, and the third is t
#For example: cat, cut, cct
c?t

#Matches strings user, users, us3rr, ussr1 and so on
#(the [:alpha:] character class matches any alphabetical
#character, the special character ? matches any character)
us[[:alpha:]34]r?

#Matches strings .con, file.col, 3...co! and so on
#(before .co—any character sequence, after—
#any character except m and ?)
*.co[!m\?]

#Matches any string containing the host part,
#for example: host, localhost, hostel, ghosts
*host*
```

- Strings from a file specified in the section of the `Regex` type are interpreted as PCREs (*Perl Compatible Regular Expressions*). Matching strings with regular expressions is done case-insensitively. Examples of regular expressions:

```
#IPv4
(\d{1,3}.){3}\d{1,3}

#Email address in the .com domain
\w+@\w+\.com
```

- Strings from a file specified in the section of the *Cidr* type are interpreted as IP addresses or IP address ranges. Both IPv4 and IPv6 formats for IP addresses as well as address ranges are allowed. A subnet mask can be specified in the bit (octet) format, as well as in the CIDR (*Classless Inter-Domain Routing*) notation, for example:

```
#IPv4
192.168.0.1
192.168.0.0/12
192.168.0.0/255.255.255.224

#IPv6
fe80::c7e8/32
fe80::c7e8/255.255.255.224
```

Parameters used in the Pq, Mysql and Sqlite sections

Parameter	Description

`Conn`
{string}
	Database connection string.
	Allowed values:
	• for the `Mysql` (MySQL) and `Pq` (PostgreSQL) sections:
	`tcp://[`*<user>[:<password>]@]<host>[:<port>][/<database name>][?<parameter>=<value>[&...]]*;
	`unix://[`*<user>[:<password>]@]<socket path>[:<database name>][?<parameter>=<value>[&...]]*;
	Examples:
	`tcp://user:pwd@localhost:1234/userdb`
	`unix://user:pwd@/tmp/pgsql.sock:userdb`
	Take notice of the URI requirements.
	• for the `Sqlite` (SQLite) section:
	Path to a database file (specify the `file://` scheme prefix).
	Example: `file:///home/user/users.db`
	Default value: *Defined by* the corresponding `*DefaultConn` parameter value

`Request` *{string}*	String of an SQL query (`SELECT`) to the database. Like with `AD` and `LDAP` sources, the following automatically resolved markers can be used in the query: • `$u`, `$U` are replaced with `user`—the user name sent by a client component; • `$d`, `$D` are replaced with `domain`—the domain sent by the client component; • `$$` is replaced with the `$` character. Example: `SELECT username FROM users INNER JOIN domains ON users.domain = domains.id WHERE domains.name = $d AND users.name = $u` Default value: *(not specified)*

Only the query of the `SELECT` type can be specified as an SQL query. After making substitutions, the query is passed to the database "as is". If the query result contains more than one column, then all columns except the first will be ignored.

Parameters used in the Redis section

Parameter	Description
`Conn` *{string}*	Redis database connection string. Allowed values: • `tcp://[<password>@]<host>[:<port>][/<database index>]`; • `unix://[<password>@]<socket path>[:<database index>]`; Take notice of the URI requirements. Example: `tcp://localhost:6379` Default value: *Defined by* the `RedisDefaultConn` parameter value
`Request` *{string}*	Redis database query string. The following automatically resolved markers can be used in the query: • `$u`, `$U` are replaced with `user`—the user name sent by a client component; • `$d`, `$D` are replaced with `domain`—the domain sent by the client component; • `$$` is replaced with the `$` character. Example: `HVALS bad_users` Default value: *(not specified)*
`UseSSL` *{boolean}*	Use or do not use SSL/TLS for connecting to a Redis database. Default value: `No`

If the query result contains more than one column, then all columns except the first will be ignored.

Adding sections for new data sources

To add a new section for a data source of some type with a *<name>* tag using the Dr.Web Ctl command-line management tool for Dr.Web Gateway Security Suite (started by running the `drweb-ctl` command), run the command:

```
# drweb-ctl cfset LookupD.<type> -a <name>
```

Example:

```
# drweb-ctl cfset LookupD.AD -a WinAD1
# drweb-ctl cfset LookupD.AD.WinAD1.Host 192.168.0.20
```

The first command will add a section named `[LookupD.AD.WinAD1]` to the configuration file, and the second command will modify the value of the `Host` parameter within this section.

Alternatively, you can write the new section directly to the configuration file (for example, by adding it to the end of the file):

```
[LookupD.AD.WinAD1]
Host = 192.168.0.20
```

Both ways have the same effect; however, if you edit the configuration file directly, you will also need to apply the changed settings by reloading the configuration of Dr.Web Gateway Security Suite by running the command:

```
# drweb-ctl reload
```

9.19. Dr.Web StatD

The Dr.Web StatD component is designed for accumulating statistics on events that occur during the operation of the Dr.Web Gateway Security Suite components. The events are registered in permanent storage and can be obtained on request.

9.19.1. Operating Principles

The Dr.Web StatD component ensures accumulation and permanent storage of events obtained during the operation of Dr.Web Gateway Security Suite components. The events related to unexpected component shutdown are logged.

To view and manage events, use the `events` command of the Dr.Web Ctl utility.

9.19.2. Command-Line Arguments

To start the Dr.Web StatD component from the command line, run the command:

- for GNU/Linux:

```
$ /opt/drweb.com/bin/drweb-statd [<parameters>]
```

- for FreeBSD:

```
$ /usr/local/libexec/drweb.com/bin/drweb-statd [<parameters>]
```

Dr.Web StatD accepts the following parameters:

Parameter	Description
`--help`	Function: Output short help information about command-line parameters to the console or the terminal emulator and shut down the component. Short form: `-h` Arguments: None
`--version`	Function: Output information about the component version to the console or the terminal emulator and shut down the component. Short form: `-v` Arguments: None

Example:

```
$ /opt/drweb.com/bin/drweb-statd --help
```

This command outputs short help information about the Dr.Web StatD component.

Startup notes

The component cannot be started directly from the command line in standalone mode. The component is started automatically by the Dr.Web ConfigD configuration management daemon when necessary. To manage the operation parameters of the component, use the Dr.Web Ctl tool designed to manage Dr.Web Gateway Security Suite from the command line.



To start the Dr.Web Ctl tool, run the `drweb-ctl` command.

To get documentation on this component from the command line, run the `man 1 drweb-statd` command.

9.19.3. Configuration Parameters

The component uses configuration parameters specified in the `[StatD]` section of the unified configuration file of Dr.Web Gateway Security Suite.

The section contains the following parameters:

Parameter	Description
`LogLevel` *{logging level}*	Logging level of the component. If a parameter value is not specified, the `DefaultLogLevel` parameter value from the `[Root]` section is used. Default value: `Notice`
`Log` *{log type}*	Logging method of the component. Default value: `Auto`
`ExePath` *{path to file}*	Component executable path. Default value: • for GNU/Linux: `/opt/drweb.com/bin/drweb-statd` • for FreeBSD: `/usr/local/libexec/drweb.com/bin/drweb-statd`
`IdleTimeLimit` *{time interval}*	Maximum idle time for the component. When the specified period of time expires, the component shuts down. Allowed values: from 10 seconds (`10s`) to 30 days (`30d`). If the `None` value is set, the component will operate indefinitely; the SIGTERM signal will not be sent if the component goes idle. Default value: `10m`
`RunAsUser` *{UID | user name}*	User on behalf of whom the component is started. You can specify either a numerical user identifier (UID) or a user name (login). If the user name

Parameter	Description
	consists of numbers (that is, the name is similar to a numerical UID), specify it with the `name:` prefix, for example, `RunAsUser = name:123456`. If the user name is invalid, the component shuts down with an error upon startup. Default value: `drweb`
`MaxEventStoreSize` *{size}*	Maximum allowed size of the event database. Defined in `mb`, for example: `MaxEventStoreSize = 100mb`. Minimal value: `50mb`. Default value: `1GB`

10. Appendices

10.1. Appendix A. Types of Computer Threats

In this section

- Computer viruses
- Computer worms
- Trojans
- Hacktools
- Adware
- Jokes
- Dialers
- Riskware
- Suspicious objects

Herein, the term *"threat"* is defined as any kind of software potentially or directly capable of inflicting damage to a computer or network and compromising user information or rights (that is, malicious and other unwanted software). Broadly speaking, the term "threat" can be used to indicate any type of potential danger to a computer or network (that is, a vulnerability, which can be used in cyberattacks).

All of the program types stated below are capable of endangering user data or confidentiality. Programs that do not conceal their presence in the system (e.g. spam distribution software or traffic analyzers) are usually not considered computer threats, although they can also harm the user under certain circumstances.

Computer viruses

Computer threats of this type are capable of embedding their code in other programs. Such embedding is called *infection*. In most cases, an infected file becomes a virus carrier and the embedded code does not necessarily match the original one. A significant number of viruses is designed to damage or destroy data.

In Doctor Web classification, viruses are separated by the type of objects they infect:

- *File viruses* infect files of the operating system (usually executable files and dynamic libraries) and are activated when an infected file is started.
- *Macro-viruses* infect documents used by Microsoft® Office (and other applications supporting macros, for example, written in Visual Basic). *Macros* are embedded programs written in a fully functional programming language and can be run in specific conditions (for instance, in Microsoft® Word, macros can be automatically started upon opening, closing or saving a document).

- *Script viruses* are created using script languages and usually infect other script files (e.g. service files of an operating system). They can also infect files of other types that allow execution of scripts and can spread, for example, via vulnerable web applications.
- *Boot viruses* infect boot records of disks and partitions as well as master boot records of hard drives. They do not require much memory and remain ready to continue performing their tasks until the system is unloaded, restarted or shut down.

Most viruses employ certain mechanisms of protection against detection. They are constantly being improved, and ways to cope with them are constantly being developed. All viruses can also be classified according to their type of protection against detection:

- *Encrypted viruses* encrypt their code upon every infection to hinder their detection in a file, a boot sector or memory. All instances of such viruses contain only a short common code fragment (the decryption procedure) that can be used as a signature.
- *Polymorphic viruses* not only encrypt their code, but they also generate a special decryption procedure that is different in every instance of the virus. As a result, such viruses do not have byte signatures.
- *Stealth viruses* (invisible viruses) perform certain actions to disguise their activity and to conceal their presence in infected objects. Such viruses gather the characteristics of an object before infecting it and then pass old data when the operating system or a program scans for modified files.

Viruses can also be classified according to a programming language in which they are written (for example, a low-level language such as an assembly language or a high-level language such as Go) or according to infected operating systems.

Computer worms

Like viruses, programs of the *"computer worm"* type can copy themselves, but they do not infect other objects. A worm gets into a computer from a network (typically as an email attachment or from the internet) and sends its functioning copies to other computers. Worms either rely on user actions or spread automatically.

Worms do not necessarily consist of only one file (the worm body). Many of them have a so called infectious part (the shellcode), which loads into the computer operating memory and then downloads the worm body as an executable file over the network. Until the worm body is downloaded to the system, the worm can be avoided by rebooting the computer (at which the operating memory is reset). However, if the worm body infiltrates the system, then only an anti-virus program can cope with it.

Worms are capable of rendering entire networks inoperable even if these worms do not carry any payload (i.e. do not cause any direct damage to the system).

In Doctor Web classification, worms are separated by their distribution method (environment):

- *Network worms* spread via various network and file sharing protocols.
- *Mail worms* spread via email protocols (POP3, SMTP and so on).

- *Chat worms* spread via popular instant messaging services (ICQ, IM, IRC and so on).

Trojans

Malware of this type cannot reproduce itself. A trojan pretends to be a popular program and performs its functions (or imitates its operation). Meanwhile, it performs some malicious actions (damages or deletes data, sends confidential information and so on) or makes it possible for a hacker to access the computer without permission, for example, in order to harm a third party.

Trojan masking and malicious features are similar to those of a virus. A trojan can even be a component of a virus. However, most trojans spread as separate executable files (through file exchange servers, removable data carriers or email attachments) that are started by users or certain system processes.

It is very hard to classify trojans due to the fact that they are often spread by viruses and worms and also because many malicious actions that can be performed by other types of threats are attributed to trojans only. Here are some trojan types that are classified by Doctor Web as separate classes:

- *Backdoors* are trojans that allow to gain privileged access to a system, bypassing existing access and security measures. Backdoors do not infect files.
- *Rootkits* are used to intercept system functions of an operating system in order to conceal themselves. Furthermore, a rootkit can hide processes of other programs, directories and files. It can spread either as an independent program or as an auxiliary component of another malicious program. There are two kinds of rootkits according to their operation mode: *User Mode Rootkits—UMR* (intercept functions of user mode libraries) and *Kernel Mode Rootkits—KMR* (intercept functions at the system kernel level, which makes them harder to detect and neutralize).
- *Keyloggers* are used to log data that users enter by means of a keyboard. The aim of this is to steal personal information (i.e. network passwords, logins, primary account numbers and so on).
- *Clickers* redefine hyperlinks when they are clicked and thus redirect users to certain websites (sometimes malicious). This is usually done to increase ad traffic of websites or perform distributed denial-of-service (DDoS) attacks.
- *Proxy trojans* provide anonymous internet access to a malicious actor through a victim's computer.

In addition, trojans can also change the start page in a web browser or delete certain files. However, these actions can also be performed by other types of threats (viruses and worms).

Hacktools

Hacktools are programs designed to assist an intruder with hacking. The most common among them are port scanners that detect vulnerabilities in firewalls and other components of a

computer protection system. Besides hackers, such tools are used by administrators to test security of their networks. Occasionally, programs that use social engineering techniques are classified as hacktools.

Adware

Usually, this term refers to program code embedded in freeware programs that forces displaying of advertisements to a user. However, sometimes such code can be distributed via other malicious programs and display advertisements, for example, in web browsers. Many adware programs operate with data collected by spyware.

Jokes

Like adware, this type of malware cannot be used to inflict any direct damage to the system. Joke programs usually just generate messages about errors that never occurred and threaten to perform actions that will lead to data loss. Their purpose is to frighten or annoy users.

Dialers

These are special programs that are designed to scan a range of telephone numbers and find those where a modem answers. These numbers are then used to mark up the price of telephoning facilities or to connect the user to expensive telephone services.

Riskware

These programs were not created for malicious purposes, but due to their characteristics they can pose a threat to the system security. Riskware can accidentally damage or delete data or be used by malicious actors or other programs to harm the system. Riskware includes various remote chat and administrative tools, FTP servers and so on.

Suspicious objects

Suspicious objects include any potential threats detected by a heuristic analyzer. Such objects can potentially relate to any type of computer threats (even unknown to information security specialists) or appear to be safe in case of false positives. It is recommended that files containing suspicious objects are quarantined and sent to Doctor Web anti-virus laboratory experts for analysis.

10.2. Appendix B. Neutralizing Computer Threats

In this section

- Detection methods
  - Signature analysis
  - Origins Tracing™
  - Execution emulation
  - Heuristic analysis
  - Cloud-based threat detection technologies
- Threat-related actions

All Doctor Web anti-virus solutions use a set of methods to detect threats, which allows to scan suspicious objects thoroughly.

Detection methods

Signature analysis

This method of detection is the first to run. It is applied by scanning object contents for known threat signatures. A signature is a continuous finite sequence of bytes necessary and sufficient to identify a threat unequivocally. At that, the search for signatures of the objects being scanned is performed using checksums, which allows to reduce virus databases significantly in size having preserved, at the same time, unequivocal matching and correct detection of threats and curing infected objects. Dr.Web virus databases are composed such that the same record can cover entire classes or families of threats.

Origins Tracing™

This unique Dr.Web technology allows to detect new and modified threats using already known infecting and damaging techniques covered by virus databases. The technology is used after completion of the signature analysis and protects users using Dr.Web anti-virus solutions against such threats as the notorious Trojan.Encoder.18 ransomware (also known as gpcode). Furthermore, using the Origins Tracing™ technology allows to considerably reduce the number of false positives of the heuristic analyzer. Objects detected using Origins Tracing™ have the `.Origin` postfix added to their names.

Execution Emulation

The technology of program code emulation is used for detection of polymorphic and encrypted viruses when a search by checksums cannot be performed directly, or is considerably complicated (for example, it is impossible to generate reliable signatures). The method consists

in emulating the execution of an analyzed code with an *emulator*—a programming model of a processor and runtime environment. An emulator operates with a protected memory area (an *emulation buffer*). At that, instructions are not passed to a CPU for actual execution. If the code processed by the emulator is infected, the emulation results in restoring the original malicious code available for signature analysis.

Heuristic analysis

The operation of the heuristic analyzer is based on a set of *heuristics*—assumptions about fingerprints of both malware and safe code, which statistical significance is proved experimentally. Each fingerprint has a certain *weight* (that is, a number which determines a level of its severity and reliability). The weight can be positive if the fingerprint is indicative of malicious behavior or negative if the fingerprint is non-typical for computer threats. Depending on the total weight of contents of an object, the heuristic analyzer calculates a probability of this object containing unknown malware. If a threshold is exceeded, the heuristic analyzer returns a verdict that the analyzed object is malicious.

The heuristic analyzer also uses the FLY-CODE™ technology, which is a versatile algorithm to unpack files. This technology allows to make heuristic assumptions about the presence of malware in objects packed not only by those packers that Dr.Web developers are aware of, but also by new, previously unknown programs. While scanning packed objects, their structural entropy is being analyzed, thereby allowing to detect threats on the basis of the allocation of their code segments. This technology allows to detect multiple different threats packed by the same polymorphous packer on the basis of a single record in a virus database.

As any system of hypothesis testing under uncertainty, the heuristic analyzer can make type I or type II errors (skipping unknown threats or raising false positives correspondingly). Thus, objects detected by the heuristic analyzer are treated as “suspicious”.

While performing any of the scans, Dr.Web anti-virus solutions use the most recent information about all known malware. Threat signatures, footprints and behavioral patterns are updated and added to virus databases as soon as experts of the Doctor Web anti-virus laboratory discover new threats, occasionally several times per hour. Even if the newest malicious program passes Dr.Web real-time protection and penetrates the system, this program will be detected in the list of processes and neutralized after updating virus databases.

Cloud-based threat detection technologies

Cloud-based detection methods allow to check any object (a file, an application, a browser extension and so on) against its *hash sum*—a unique sequence of digits and letters of a given length. When checked against their hash sums, objects are searched in the existing database and then classified into categories: clean, suspicious, malicious and so on.

This technology reduces time required for file scanning and saves device resources. The verdict is almost instantaneous given that the hash sum and not the object itself is analyzed. If Dr.Web

Cloud servers are unavailable, the files are scanned locally, and the cloud scanning is resumed when the connection is restored.

Thus, the Dr.Web Cloud service collects information from numerous users and quickly updates data on previously unknown threats thereby increasing the effectiveness of device protection.

Threat-related actions

Dr.Web products implement a number of actions that can be applied to detected objects to neutralize computer threats. The user can keep default actions applied to specific types of threats automatically, adjust these actions or choose the required action manually each time upon detection. Available actions are provided below. The brackets contain a parameter denoting a corresponding action and used in the unified configuration file, traffic monitoring rules and commands of the Dr.Web Ctl tool.

- **Report** (`REPORT`)—report the threat without applying any action to the infected object.
- **Cure** (`CURE`)—attempt to cure the infected object by removing only malicious content from its body.

  This action can be applied only to certain types of threats.

- **Quarantine** (`QUARANTINE`)—put the infected object (if possible) in a specialized quarantine directory to isolate it.
- **Delete** (`DELETE`)—permanently delete the infected object.

  If a threat is detected in a file inside a container (an archive, an email message and so on), the container is quarantined and not deleted.

10.3. Appendix C. Technical Support

If you have a problem installing or using Doctor Web products, please try the following before contacting technical support:

1. Download and review the latest manuals and guides.
2. See the Frequently Asked Questions section.
3. Browse the official Doctor Web forum.

If you haven't found a solution to your problem, you can fill out a web form in the appropriate section of the official website to request direct assistance from Doctor Web technical support specialists.

For information on regional and international offices of Doctor Web, please visit the official website.

To facilitate processing of your issue, we recommend that you generate a data set for the installed product, its configuration, and system environment before contacting our technical support. To do that, you can use a special utility included in Dr.Web Gateway Security Suite.

To collect data for our technical support, run the command:

- for GNU/Linux:

```
# /opt/drweb.com/bin/support-report.sh <path to file>
```

- for FreeBSD:

```
# /usr/local/libexec/drweb.com/bin/support-report.sh <path to file>
```

where *<path to file>*—path to an archive in the `.tgz` format to which the debugging information about the product and the system environment will be written, for example, `/tmp/report.tgz`. Already existing files will not be rewritten. If the path is not specified, the archive will be stored in the directory of the superuser who started the utility (for example, `/root`) and will be named as follows:

```
drweb.report.<timestamp>.tgz
```

where *<timestamp>* is a full timestamp of creating the report, down to milliseconds, for example: 20190618151718.23625.



The utility for collecting data for our technical support must be run with superuser (*root*) privileges. To get superuser privileges, use the `su` command to change the user or the `sudo` command to run the command as another user.

During operation, the utility collects and archives the following information:

- information about your OS (name, architecture, output of the `uname -a` command);
- list of packages installed on your system, including Doctor Web packages;
- log contents:
  - Dr.Web Gateway Security Suite logs (if configured for separate components);
  - log of the `syslog` system daemon (`/var/log/syslog`, `/var/log/messages`);
  - log of a system package manager (`apt`, `yum`, etc.);
  - `dmesg` log;
- output of the commands `df`, `ip a` (`ifconfig -a`), `ldconfig -p`, `iptables-save`, `nft export xml`.
- information about settings and configuration of Dr.Web Gateway Security Suite:
  - list of downloaded virus databases (`drweb-ctl baseinfo -l`);
  - list of files from Dr.Web Gateway Security Suite directories and MD5 hash values of these files;
  - Dr.Web Virus-Finding Engine version and MD5 hash value;

- information about the user and permissions retrieved from the key file, if Dr.Web Gateway Security Suite is not running in centralized protection mode.

10.4. Appendix D. Dr.Web Gateway Security Suite Configuration File

Configuration parameters of all Dr.Web Gateway Security Suite components are managed by the Dr.Web ConfigD configuration management daemon. Changed parameters are stored in the `drweb.ini` file whose default directory is `/etc/opt/drweb.com/` for GNU/Linux or `/usr/local/etc/drweb.com/` for FreeBSD.



The text configuration file stores only those parameters whose values differ from defaults. If a parameter is absent in the configuration file, its default value is used.

View the list of all available parameters, including those that are absent in the configuration file and have default values, by running the command:

```
$ drweb-ctl cfshow
```

You can change any parameter value in two ways:

- By running the command:

```
# drweb-ctl cfset <section>.<parameter> <new value>
```



This command requires to run the Dr.Web Ctl management tool with superuser (the *root* user) privileges. To obtain superuser privileges, use the `su` or `sudo` command.

For further information about the syntax of the `cfshow` and `cfset` commands of the Dr.Web Ctl tool (the `drweb-ctl` module), refer to the Dr.Web Ctl section.

- By specifying this parameter in the configuration file (by editing the file in any text editor) and reloading the configuration of Dr.Web Gateway Security Suite to apply changes by running the command:

```
# drweb-ctl reload
```

10.4.1. File Structure

The configuration file complies with the rules indicated below.

- The file content is separated into named sections. Allowed section names are strictly set and cannot be arbitrary. A section name is specified in square brackets and is identical to the name of the Dr.Web Gateway Security Suite component that uses parameters of this section (except for the `[Root]` section, which stores parameters of the Dr.Web ConfigD configuration daemon).
- The `;` or `#` characters in the beginning of the lines of the configuration file indicate a comment. Such lines are skipped by Dr.Web Gateway Security Suite components while reading configuration parameters from the configuration file.

- Each line in the file can contain only one parameter:

  ```
  <parameter> = <value>
  ```

- All parameter names are strictly set and cannot be arbitrary.
- All section and parameter names are case-insensitive. Parameter values, except for names of directories and files in paths (for Unix-like OSes), are also case-insensitive.
- Parameter values in the configuration file must be enclosed in quotation marks if these values contain spaces.
- Some parameters can accept multiple values. If so, the values are either comma-separated or specified several times in different lines of the configuration file. In the former case, spaces around a comma are ignored. If a space character is a part of a parameter value, the entire value must be enclosed in quotation marks.

  You can specify multiple values as:

  - a comma-separated list:

    ```
    Parameter = "Value1", "Value2","Value 3"
    ```

  - a sequence of lines in the configuration file:

    ```
    Parameter = Value2
    Parameter = Value1
    Parameter = "Value 3"
    ```

  The order of values is unimportant.

  Paths to files are always enclosed in quotation marks when separated by commas, for example:

  ```
  ExcludedPaths = "/etc/file1", "/etc/file2"
  ```

For the description of the sections of the configuration file, see the description of the Dr.Web Gateway Security Suite components using it.

10.4.2. Parameter Types

Configuration parameters can be of the following types:

1. *Address*—network connection address specified as *<IPv4 address>:<port>* or *<IPv6 address>:<port>*. Use square brackets to set an IPv6 address value. In some cases the port can be omitted (in each case this is indicated in the description of the parameter).
2. *Boolean*—flag parameter accepting `On`, `Yes`, `True` (the parameter is enabled) and `Off`, `No`, `False` (the parameter is disabled).
3. *Integer*—non-negative integer.
4. *Fractional number*—non-negative number with a fractional part can be indicated as a parameter value.

5. *Time interval*—time interval consisting of a non-negative integer and a suffix character indicating the specified unit of measurement. The following suffixes representing units of measurement can be used:
   - `w`—weeks (`1w = 7d`);
   - `d`—days (`1d = 24h`);
   - `h`—hours (`1h = 60m`);
   - `m`—minutes (`1m = 60s`);
   - `s` or no suffix—seconds.

   If the time interval is specified in seconds, you can specify milliseconds after a point (no more than three digits, for example, `0.5s`—500 milliseconds). It is possible to specify multiple time intervals in different time units as a single time interval; in this case, it is counted as a sum of intervals (in fact, the configuration parameters always store a number of milliseconds covered by this time interval).

   In general terms, any time interval can be represented by the following expression: N_1wN_2dN_3hN_4mN_5[.N_6]s, where N_1, ..., N_6 is a number of the corresponding time unites included in this interval. For example, a year (365 days) can be represented as follows (all records are equivalent): `365d`, `52w1d`, `52w24h`, `51w7d24h`, `51w7d23h60m`, `8760h`, `525600m`, `31536000s`.

   The examples below show you how intervals of 30 minutes, 2 seconds, 500 milliseconds can be specified:

   - in the configuration file:

   ```
   UpdateInterval = 30m2.5s
   ```

   - by running the `drweb-ctl cfset` command:

   ```
   # drweb-ctl cfset Update.UpdateInterval 1802.5s
   ```

   - via a command-line parameter (for example, for the scanning engine):
     - for GNU/Linux:

     ```
     # /opt/drweb.com/bin/drweb-se <socket> --WatchdogInterval 1802.5
     ```

     - for FreeBSD:

     ```
     # /usr/local/libexec/drweb.com/bin/drweb-se <socket> --WatchdogInterval 1802.5
     ```

6. *Size*—the parameter value represents a size of an object (a file, a buffer, a cache, and so on), specified as a non-negative integer and a suffix representing a measurement unit. The following suffixes that specify size units can be used:
   - `GB`—gigabytes (`1GB = 1024MB`);
   - `MB`—megabytes (`1MB = 1024KB`);
   - `KB`—kilobytes (`1KB = 1024B`);
   - `B`—bytes.

If a suffix is omitted, the size is considered to be in bytes. A single size record can be specified by multiple sizes in different units; in this case, the resulting size is counted as their sum (in fact, the configuration parameters always store the size in bytes).

7. *Path to directory (file)*—the parameter value is a string representing an acceptable path to a directory (a file).



The file path must be ended with a file name.



On Unix-like operating systems, names of directories and files are case-sensitive. If it is not explicitly mentioned in a parameter description, paths cannot be represented by masks with special characters (`?`, `*`).

8. *Logging level*—level at which the Dr.Web Gateway Security Suite component events are logged. The following values are allowed:
   - `Debug`—the most detailed logging level. All messages and debug information are logged.
   - `Info`—all messages are logged.
   - `Notice`—all error messages, warnings and notifications are logged.
   - `Warning`—all error messages and warnings are logged.
   - `Error`—only error messages are logged.
9. *Log type*—the parameter value specifies the Dr.Web Gateway Security Suite component logging method. The following values are allowed:
   - `Stderr[:ShowTimestamp]`—messages are output to *stderr* (standard error stream). This value can be used *only* in the settings of the configuration daemon. At that, if it works in the background ("*daemonized*"), i.e. it is started with the `-d` parameter, this value *cannot* be used because components operating in the background cannot access input/output streams of the terminal). The additional `ShowTimestamp` parameter instructs to add a timestamp to every message.
   - `Auto`—messages for logging are passed to the Dr.Web ConfigD configuration daemon, which stores them in a single location specified in its own settings (the `Log` parameter in the `[Root]` section). This value is specified for all components *except for the configuration daemon* and is used as a default value.
   - `Syslog[:`<*facility*>`]`—the component will pass messages to the `syslog` system logging service.
   - The additional <*facility*> option is used to specify a type of a log to store messages logged by `syslog`. Allowed values:
     - `Daemon`—messages from daemons,
     - `User`—messages from user processes,
     - `Mail`—messages from mail programs,
     - `Local0`—messages from local processes 0,

    - ...
    - `Local7`—messages from local processes 7.
  - *<path to file>*—messages will be stored by the component directly in the specified log.

  Examples of how to specify a parameter value:

  - in the configuration file:

```
Log = Stderr:ShowTimestamp
```

  - by running the `drweb-ctl cfset` command:

```
# drweb-ctl cfset Root.Log /tmp/log
```

  - via a command-line parameter (for example, for the scanning engine):
    - for OSes of the GNU/Linux family:

```
# /opt/drweb.com/bin/drweb-se <socket> --Log Syslog:DAEMON
```

    - for FreeBSD:

```
# /usr/local/libexec/drweb.com/bin/drweb-se <socket> --Log Syslog:DAEMON
```

10. *Action*—action to be applied by the Dr.Web Gateway Security Suite component upon detection of certain threats or upon another event. Allowed values:
    - **Report** (`REPORT`)—only notify of threat detection without applying other actions;
    - **Cure** (`CURE`)—attempt to cure the threat (that is, remove only malicious content from the file body);
    - **Quarantine** (`QUARANTINE`)—put the infected file in quarantine;
    - **Delete** (`DELETE`)—delete the infected file.

    Some of the actions can be applied only upon certain events (for example, a "Scanning error" event cannot trigger the `CURE` action). Allowed actions are always listed in the description of each parameter of the *action* type.

Other types of parameters and their allowed values are explicitly specified in the description of these parameters.

10.4.3. Traffic Monitoring Rules

In this section

- General information
- Rule format
- Conditions
- Actions
- Variables used in the rules

- Categories of unwanted websites and threats
- Configuration parameters used in rule conditions
- Aspects of writing the rules to the configuration file

General information

The rules are represented by such constructs as IF <*conditional part*> THEN <*action part*>. The following condition types are specified in the *conditional part*: "The variable value is (not) set" or "The variable value is (not) included in the specified set". The *action part* contains a set of actions (at least one action), whereas each of them is a *final resolution* (skip or block a scanned object) or a *modifying action* such as "Change properties of the scanned object", "Set the indicated variable to the specified value" or "Add the specified value to a set of values of the indicated variable".

The actions specified in a rule are performed only if the conditional part is true. If the conditional part is false, the actions specified in this rule are not performed and the program goes to the next rule. The rules are processed from top to bottom until a final resolution is triggered. After this, all rules below (if there are any) are ignored.



When a rule is triggered, the actions in the *action part* are performed from left to right in order in which they are specified, and if there is a final resolution in the chain of actions that interrupts the rule processing, the actions following it are not performed.

Rule format

The rules have the following format:

```
[<condition>[, <condition>[, …]]] : <action>[, <action>[, …]]
```

The conditional part of the rule (before the "**:**" character) can be absent, in this case the action part is performed without any condition. If the conditional part of the rule is absent, the "**:**" separator can be omitted. The comma between conditions in the conditional part and actions in the action part performs a role of a logical conjunction (that is, logical "AND"): the conditional part is true only if all its conditions are true, and all actions specified in the action part are performed in order of their specification from left to right until a final resolution, which interrupts the rule handling, is reached. Key words, names of variables and configuration parameters used in the rules are case-insensitive.

Conditions

The following types of conditions can be used in the rules:

Condition	Comment
<*variable*> <*value*>	The variable value matches the specified one. Can be used only for variables that do not accept a set of values.
<*variable*> `[not] in` <*set of values*>	The value of the variable is contained in the specified set of values (for `not`—does not match any value from the specified set).
<*variable*> `[not] match` <*set of values*>	The value of the variable matches a regular expression from the specified set (for `not`—does not match any expression from the specified set). Regular expressions are specified using either the *POSIX* syntax (*BRE* or *ERE*) or the *Perl* syntax (*PCRE* or *PCRE2*).
<*variable*> `[not] gt` <*value*>	The value of the variable is (not) greater than the specified value. Can be used only for variables that have a single numerical value.
<*variable*> `[not] lt` <*value*>	The value of the variable is (not) less than the specified value. Can be used only for variables that have a single numerical value.

*) The optional key word `not` means negation.

The part <*set of values*> to which the variable is compared can be specified in the following ways:

Syntax	Meaning
`(`<*value 1*>`[, `<*value 2*>`[, ...]])`	A set of values to check against (no less than one value) is listed directly in the parentheses. In case there is only one value and the `in` condition is used, you can omit the

Syntax	Meaning
	parentheses (and you will end up with <*variable*> <*value*>).
"<*section*>.<*parameter*>"	A set of configuration parameters, where a parameter whose value (or a set of values) is to be checked is put in quotation marks (a section to which the parameter belongs is also specified). Lists of parameters that can be used in a condition depend on the component for which the rules are set and are provided below.
`file("`<*file name*>`")`	A list of values is read from a text file <*file name*> (one line per a list item, leading and trailing spaces in strings are ignored). A path to the file must be absolute. If the <*file name*> contains quotation marks and apostrophes, they must be escaped with a backslash ('\').  The file size must not exceed 64 MB. The file content is read and inserted into the rules once—when the configuration file is being loaded. If the file does not exist or the file size is exceeded, error x102 is returned when loading the settings. If you have edited the file, reload the Dr.Web Gateway Security Suite configuration to apply the changes by running the command: `# drweb-ctl reload` The file does not provide a set of values for all variables. It is indicated for each variable below whether you can check its values against a set of values from the file.
<*LOOKUP_query_type*>`@`<*tag*>`[@`<*value*>`]`	A set of values is queried via Dr.Web LookupD from an external data source, where <*LOOKUP_query_type*> is a type of the source, <*tag*> is a name of the section describing a connection for checked parameter sampling, and optional <*value*> is a value that must be in the set of the values retrieved from the data source.

Syntax	Meaning
	Values from Dr.Web LookupD are not available for all variables. In addition, the <*check*> condition is not used for all variables. It is indicated below for each variable whether its values can be checked against the values received via Dr.Web LookupD.
`dnsxl(`<*DNSxL server 1*>`[: [mask]` <*IP*>`][, …])`	The brackets contain a list of DNSxL servers (DNSBL and so on) that must check whether their lists of IP addresses include an IP address or FQDN (resolved to the IP address in advance). If the IP address in check is included in the lists of one of the DNSxL servers put in brackets, the response of this server contains one or more DNS records of type A; furthermore, a fake IP address returned by the server can contain a reason why the IP address in check was included in the lists of the server (generally, a type of the reason is determined by a value of the last octet of the returned fake IP address). For each DNSxL server in the list, it is possible to assign check of the expected returned value of the fake IP address. The check is specified after the colon as follows: • <*DNSxL server*>`:` <*IP address*> • <*DNSxL server*>`: mask` <*IP address*> The first case requires the fake IP address returned by the server <*DNSxL server*> to exactly match the specified address <*IP address*>. The second case requires the fake IP address returned by the server <*DNSxL server*> to exactly match the specified mask in its nonnull octets. If the check parameters are not specified, the condition is triggered if <*DNSxL server*> returns any fake IP address as a response to the request. Examples: <*IP*> `in dnsxl("dnsxl.server.org")`—for the IP address contained in the <*IP*> variable, the server must return any fake IP address; <*IP*> `in dnsxl("dnsxl.server.org": 127.0.0.2)`—for the IP address contained in the <*IP*> variable, the server must return the fake IP address of `127.0.0.2`; <*IP*> `in dnsxl("dnsxl.server1.org": mask 0.0.0.8, "dnsxl.server2.org": 127.0.0.3, "dnsxl.server3.org")`—for the IP address contained in the <*IP*> variable, either the first server returns a fake IP address with low octet `8`, or the

Syntax	Meaning
	second one—the fake IP address of `127.0.0.3`, or the third one—any fake IP address. Use of the check instruction <*variable*> `in dnsxl(`<*server list*>`)` is allowed only if <*variable*> is an IP address or a domain name that can be resolved by the DNS service to an IP address (FQDN). Thus, only the `src_ip` and `url_host` variables can be used as a variable for this condition (see further).

If a variable is multiple-valued, the condition <*variable*> `in` <*set of values*> is true, if an intersection of the set of current values of the specified variable <*variable*> with the indicated set <*set of values*> is not empty. Otherwise, the condition `not in` is true.

For example, suppose `X` is a variable whose current value is a set containing values *a*, *b* and *c*. Then:

- `X in (a, b)` is true, because values *a* and *b* are present in both sets;
- `X in (a, d, e)` is true, because value *a* is present in both sets;
- `X in (d, e)` is false, because none of the values of variable (*a*, *b*, *c*) is in set (*d*, *e*);
- `X in ()` is false, because the set of variable values is not empty;
- `X not in ()` is true, because the set of variable values is not empty;
- `X not in (d, e)` is true, because none of the values of variable (*a*, *b*, *c*) is in set (*d*, *e*);
- `X not in (a, d, e)` is false because value *a* is present in both sets.

The description of each variable below indicates whether it accepts a set of values.

Actions

Actions are grouped into *final resolutions*, which determine whether to allow or deny passing an object, and *actions changing variable values*, which can be used when checking conditions in further rules.

Final resolutions

Resolution	Description (meaning)
General resolutions	
`PASS`	Pass the traffic (allow establishing a connection and sending the object to the recipient). The further rules (if there are any) are not used.

Resolution	Description (meaning)
`BLOCK as <reason>`	Block the traffic (deny establishing a connection and sending the object to the recipient). Further rules (if there are any) are not used. The reason <*reason*> for blocking is logged. The same reason is used to determine which page with the notification should be displayed to the user in a browser. Two common reasons can be used as <*reason*> for `BLOCK`: • `BlackList`—the data is blocked because it has been blacklisted by the user. • `_match`—blocking occurs because a web resource or a file containing a threat belongs to a category that triggers a rule (for conditions `*_category in (...)`). The `_match` variable contains a list of blocked categories for which a match has been found.

Aspects of processing final resolutions:

- `BLOCK as BlackList` is always processed as *"is blacklisted"* (irrespective of the condition specified in the rule with this resolution).
- `BLOCK as _match`, if `_match` is not empty, is processed as *"belongs to the _match category(-ies)"*.
- `BLOCK as _match`, if `_match` is empty, is processed as *"is blacklisted"* (irrespective of a condition specified in a rule with this resolution).
- If all rules have been processed, and none of the rules with the resolution has triggered (or the rules do not have resolutions), this is equivalent to applying the `PASS` action to the connection.

Changing a variable value

To change a variable value, use the following instruction:

```
SET <variable> = ([<value 1>[, <value 2>[, …]]])
```

If there are no values in parentheses, the list of the variable values will be cleared:

```
SET <variable> = ()
```

If the variable has only one value, the parentheses are not used:

```
SET <variable> = <value>
```

Variables used in the rules

When indicating variables in the rules, the character case is ignored. Multiword variables can be named with or without an underscore to separate words. Thus, a variable can be named

variable_name, VariableName or variablename. In this section, all variables contain the underscore (that is, the variable_name variant is used).

Variable	Description	Can be used in	
		condition	action (SET)

protocol	Network protocol type used by the connection.
	
	The variable accepts multiple values.
	
	Allowed values: HTTP, SMTP, IMAP and POP3.
	
	Usage aspects
	• The variable value is defined only if SSL/TLS is not used or SSL scanning is enabled.
	• Dr.Web ICAPD rules accept only HTTP as its value.
	• A set of values for checking the variable value can be read from a file.
	
	Examples:
	`protocol in (HTTP, SMTP)`
	`protocol in (POP3)`
	`protocol in file("/etc/file")`
		Yes	No

sni_host	SNI (address) of the host to which the connection is established via SSL/TLS. **Usage aspects** • If SSL is not used, the variable value is not defined; therefore, the condition is false. • Dr.Web ICAPD rules do not use this variable (this component does not process SSL; therefore, the condition is always false). • A set of values for checking the variable value can be read from a file. • You can use this variable together with the proc variable (see below).	Yes	No

Variable	Description	Can be used in	
		condition	action (SET)

	Examples:
	`sni_host not in ('vk.com', 'ya.ru')`
	`sni_host in "LinuxFirewall.BlackList"`
	`sni_host in file("/etc/file")`		

`sni_category`	List of categories (`AdultContent` and so on) to which the host (identified by SNI) belongs in accordance with the databases of web resource categories; the connection to the host is established via SSL/TLS.
	
	The variable accepts multiple values.
	
	Usage aspects
	• If SSL is not used, the variable value is not defined; therefore, the condition is false.
	• There is no point in using the variable in Dr.Web ICAPD rules (this component does not process SSL, so the condition is always false).
	• A condition with `not in` is *true* in the rules used by Dr.Web ICAPD, even if the host does not belong to any of the predetermined categories ("safe" host). The condition is *false* in this case for Dr.Web Firewall for Linux (SpIDer Gate) rules.
	• If databases of web resource categories are not installed, do not use the variable in the rules (attempts to check if a condition in the rule is true will cause error x112).
	• A set of values for checking the variable value can be read from a file.
	
	Examples:
	`sni_category not in (AdultContent, Chats)`
	`sni_category in "LinuxFirewall.BlockCategory"`
	`sni_category in (FreeEmail)`
	`sni_category not in file("/etc/file")`
		Yes	No

Variable	Description	Can be used in	
		condition	action (SET)

`url`	URL requested by the client. Can be compared with a string or a regular expression.
	
	Usage aspects
	• Can be used only in Dr.Web ICAPD rules.
	• You can use Dr.Web LookupD to check the value of this variable.
	• A set of values for checking the variable value can be read from a file.
	• You can use this variable together with the `proc` variable (see below).
	
	Examples:
	`url match ("drweb.com", "example\..*", "aaa\.ru/")`
	`url match "ICAPD.Adlist"`
	`url not match LDAP@BadURLs`
	`url match file("/etc/file")`
		Yes	No

`url_host`	URL or host to which the connection is established.
	
	Usage aspects
	• The variable value is defined only if SSL/TLS is not used or SSL scanning is enabled.
	• You can use Dr.Web LookupD to check the value of this variable.
	• A set of values for checking the variable value can be read from a file.
	
	Examples:
	`url_host in ('vk.com', 'ya.ru')`
	`url_host not in "ICAPD.Whitelist"`
	`url_host in LDAP@hosts`
	`url_host not in file("/etc/file")`
		Yes	No

`url_category`	List of categories to which the URL or the host to which the connection is established belongs (in accordance with the databases of web resource categories or the Dr.Web Cloud response).	Yes	No

Variable	Description	Can be used in	
		condition	action (SET)

	The variable accepts multiple values.
	
	Usage aspects
	
	• The variable value is defined only if SSL/TLS is not used or SSL scanning is enabled.
	• A condition with `not in` is *true* in the rules used by Dr.Web ICAPD, even if the URL or the host does not belong to any of the predetermined categories ("safe" URL or host). The condition is *false* in this case for Dr.Web Firewall for Linux (SplDer Gate) rules.
	• If databases of web resource categories are not installed, do not use the variable in the rules (attempts to check if a condition in the rule is true will cause error x112).
	• A set of values for checking the variable value can be read from a file.
	
	Examples:
	
	`url_category not in (AdultContent, Chats)`
	`url_category in "LinuxFirewall.BlockCategory"`
	`url_category in (FreeEmail)`
	`url_category in file("/etc/file")`		

`threat_category`	List of categories to which a threat found in transmitted data is related in accordance with the virus databases. The variable accepts multiple values. **Usage aspects** • The variable value is defined only if SSL/TLS is not used or SSL scanning is enabled. • A condition with `not in` is *true* for the rules used by Dr.Web ICAPD, even if the object does not contain threats from any of the predetermined categories ("safe" object). The condition is *false* in this case for Dr.Web Firewall for Linux (SplDer Gate) rules.	Yes	No

Variable	Description	Can be used in	
		condition	action (SET)

	• A set of values for checking the variable value can be read from a file.
	Examples:
	`threat_category in "LinuxFirewall.BlockThreat"`
	`threat_category not in (Joke)`
	`threat_category in file("/etc/file")`		

`user`	User on behalf of whom a process for sending (or receiving) traffic is started.
	Usage aspects
	• In the Dr.Web ICAPD rules, the name of the user who has authenticated on a proxy server is implied (if the proxy server supports authentication). If the proxy server does not support user authentication, the variable has an empty value.
	• You can use Dr.Web LookupD to check the value of this variable. If you need to find out whether a user belongs to a certain group, use an LDAP or Active Directory data source that returns a list of groups. The query must also contain a condition for comparing the name of the group that includes the user with the required group name (use the following format: *<type of the LookupD source>@<source of groups>@<required group>*). Queries to Active Directory (AD@) return only lists of groups; therefore, it is mandatory to use the *@<required group>* part for these queries.
	• A set of values for checking the variable value can be read from a file.
	Examples:
	`user in ('user1', 'user2')`
	`user in AD@Winusergroups@Admins`
	`user in LDAP@AllowedUsers`
	`user not in file("/etc/file")`
		Yes	No

`src_ip`	IP address of the host establishing the connection.	Yes	No

Variable	Description	Can be used in	
		condition	action (SET)

	Usage aspects
	• You can use Dr.Web LookupD to check the value of this variable.
	• A set of values for checking the variable value can be read from a file.
	Examples:
	`src_ip not in (127.0.0.1, 10.20.30.41, 198.126.10.0/24)`
	`src_ip in LDAP@AllowedAddresses`
	`src_ip not in file("/etc/file")`		

`proc`	Process establishing the connection (the full executable path).
	Usage aspects
	• There is no point in using this variable in Dr.Web ICAPD rules (this component does not have information about processes, so the condition is always false).
	• A set of values for checking the variable value can be read from a file.
	• You can use this variable together with `sni_host`, `url` and `dst_address` (see below).
	Examples:
	`proc in ('/usr/bin/ls')`
	`proc not in ('/home/user/myapp', '/bin/bin1')`
	`proc in "LinuxFirewall.ExcludedProc"`
	`proc in file("/etc/file")`
		Yes	No

`direction`	Connection traffic type. Allowed values: `request` (client request), `response` (server response). This variable does not accept multiple values; conditions of the `match` and `in` type cannot be applied.	Yes	No

Variable	Description	Can be used in	
		condition	action (SET)

	Examples:
	`direction request`
	`direction not response`		

`divert`	Connection direction.
	
	Allowed values: `input` (incoming—created or initiated outside of the local host), `output` (outgoing—created or initiated on the local host).
	
	This variable does not accept multiple values; conditions of the `match` and `in` type cannot be applied.
	
	Examples:
	`divert input`
	`divert not output`
		Yes	No

`content_type`	MIME type of data transmitted via connection.
	
	Usage aspects
	• The variable value is defined only if SSL/TLS is not used or SSL scanning is enabled.
	• The expression "`*/*`" matches data of any MIME type and HTTP responses without the `Content-Type` header.
	• You can use Dr.Web LookupD to check the value of this variable.
	• A set of values for checking the variable value can be read from a file.
	
	Examples:
	`content_type in ("multipart/byteranges", "application/octet-stream")`
	`content_type not in ("text/*", "image/*")`
	`content_type not in ("audio/*")`
	`content_type in ("*/*")`
	`content_type in LDAP@BlockedContent`
	`content_type not in file("/etc/file")`
		Yes	No

Variable	Description	Can be used in	
		condition	action (SET)

(proc, <*variable*>)	Network activity of the process, where proc is the full process executable path (see above), and <*variable*> defines a type of the network activity and can accept one of the following values:
	• sni_host—SNI (address) of the host to which the connection is established via SSL/TLS (see above);
	• url—URL requested by a client (see above);
	• dst_address—network address (<*IP address*>:<*port*>) at which the process establishes the connection.
	Usage aspects
	• Used only with the condition match ({<*Proc_reg*>, <*Var_reg*>} [, …]), where <*Proc_reg*> is a regular expression for proc, and <*Var_reg*> is a regular expression for <*variable*>.
	• There is no point in using this variable in Dr.Web ICAPD rules (this component does not have information about processes, so the condition is always false).
	Examples:
	`(proc, url) match ({"/usr/bin/wget", "www\.ya\.*"})`
	`(proc, dst_address) match ({"/usr/bin/.*", "192\.168\.1\.\d+:12345"})`
		Yes	No

unwrap_ssl	Scan or do not scan secure traffic passing via SSL/TLS. Allowed values: true, false. **Usage aspects** • The value of the variable cannot be empty. The instruction SET unwrap_ssl = () is illegal. • The variable cannot be used in conditions and is necessary only to manage SSL scanning (for example, to	No	Yes

Variable	Description	Can be used in	
		condition	action (SET)

	display a webpage containing notification about blocking).
	• There is no point in using this variable in Dr.Web ICAPD rules (this component does not process SSL, changing the variable value does not influence rule processing).
	Examples:
	`SET unwrap_ssl = TRUE`
	`set Unwrap_SSL = false`		

`http_templates_dir`	Path to the directory storing block page templates.
	If the path starts with the "`/`" character, it is an absolute path; if it starts with any other character, it is a relative path. In the latter case, the root directory is defined by a path specified in the `TemplatesDir` parameter.
	Usage aspects
	• Can be used only for HTTP(S).
	Examples:
	`SET http_templates_dir = "/etc/mytemplates"`
	`set http_templates_dir = "templates_for_my_site"`
		No	Yes

Categories of unwanted websites and threats

1. Categories of unwanted websites (for the variables `sni_category` and `url_category`)

Convention	Website category
`InfectionSource`	Websites containing malware ("infection sources")
`NotRecommended`	Fraudulent websites (that use "social engineering") visiting which is not recommended
`AdultContent`	Websites that contain pornographic or erotic materials, dating websites and so on

Convention	Website category
Violence	Websites that encourage violence or contain materials about various fatal accidents and so on
Weapons	Websites that describe weapons and explosives or provide information on their manufacturing and so on
Gambling	Websites that provide access to gambling, online casinos, auctions, including websites for placing bets, and so on
Drugs	Websites that promote use, production or distribution of drugs and so on
ObsceneLanguage	Websites that contain obscene language (in section titles, articles and so on)
Chats	Websites that offer real-time exchange of text messages
Terrorism	Websites that contain aggressive and propaganda materials or description of terrorist attacks and so on
FreeEmail	Websites that offer free email registration
SocialNetworks	Various social networking services: general, professional, corporate, interest-based; special dating websites
DueToCopyrightNotice	Websites, links to which are provided by copyright holders of some copyrighted work (movies, music and so on)
OnlineGames	Websites that provide access to games using permanent internet connection
Anonymizers	Websites that allow the user to hide personal information and grant access to blocked web resources
CryptocurrencyMiningPool	Websites that grant access to services uniting users to mine cryptocurrencies
Jobs	Job search websites

As values of the variables `sni_category` and `url_category`, you can also use names of the parameters that control blocking (see below).

2. Threat categories (for the `threat_category` variable)

Convention	Threat categories
KnownVirus	Known threat
VirusModification	Modification of a known threat
UnknownVirus	Unknown threat, suspicious object

Convention	Threat categories
`Adware`	Adware
`Dialer`	Dialer
`Joke`	Joke program
`Riskware`	Riskware
`Hacktool`	Hacktool

As a value of the `threat_category` variable, you can also use names of the parameters that control blocking (see below).

Configuration parameters used in rule conditions

Parameters used in the Dr.Web Firewall for Linux component rules and indicated with the `LinuxFirewall.` prefix:

Parameter	Description and usage example

`Whitelist`	White list, i.e. a list of domains the access to which is allowed, even if these domains are included in the database of web resource categories.
	
	Examples:
	`sni_host in "LinuxFirewall.Whitelist" : PASS`
	`url_host not in "LinuxFirewall.Whitelist" : BLOCK as _match`

`Blacklist`	Black list contains a list of domains the access to which is blocked by the user or the administrator.
	
	Examples:
	`sni_host in "LinuxFirewall.Blacklist" : SET Unwrap_SSL = FALSE`
	`url_host in "LinuxFirewall.Blacklist" : BLOCK as BlackList`

`BlockCategory`	"Metaparameter": a list of categories (`Chats`, `AdultContent` and so on) for which the `Block*` parameters in the `[LinuxFirewall]` section are set to `Yes`.
	
	Examples:
	`url_category in "LinuxFirewall.BlockCategory" : BLOCK as _match`
	`sni_category in "LinuxFirewall.BlockCategory" : BLOCK as BlackList`

`BlockThreat`	"Metaparameter": a list of names of threat types (`KnownVirus`, `Joke` and so on) for which the `Block*` parameters in the `[LinuxFirewall]` section are

Parameter	Description and usage example
	set to `Yes`. Examples: `threat_category in "LinuxFirewall.BlockThreat" : BLOCK as _match`
`ExcludedProc`	List of trusted processes whose traffic must be excluded from scanning. Examples: `proc in "LinuxFirewall.ExcludedProc" : PASS`

Parameters used in the Dr.Web ICAPD component rules and indicated with the `ICAPD.` prefix:

Parameter	Description and usage example
`Whitelist`	White list contains a list of domains the access to which is allowed, even if these domains are included in the database of web resource categories. Examples: `url_host not in "ICAPD.Whitelist" : BLOCK as BlackList`
`Blacklist`	Black list contains a list of domains the access to which is blocked by the user or the administrator. Examples: `url_host in "ICAPD.Blacklist" : BLOCK as BlackList`
`Adlist`	Advertisement list contains a list of regular expressions that cover advertising websites. The list is defined by the user or the administrator. Examples: `url match "ICAPD.Adlist" : BLOCK as BlackList`
`BlockCategory`	"Metaparameter": a list of categories (`Chats`, `AdultContent` and so on) for which the `Block*` parameters in the `[ICAPD]` section are set to `Yes`. Examples: `url_category in "ICAPD.BlockCategory" : BLOCK as _match`
`BlockThreat`	"Metaparameter": a list of names of threat types (`KnownVirus`, `Joke`, etc.) for which the `Block*` parameters in the `[ICAPD]` section are set to `Yes`. Examples: `threat_category in "ICAPD.BlockThreat" : BLOCK as _match`

Aspects of writing the rules to the configuration file

- The rules are stored in the configuration file, in the settings sections of components that use rules, in such variables as `RuleSet`, each of them is a set (sequence) of an unlimited number of rules. In addition, rules in each set are processed sequentially (from top to bottom) until the final resolution is reached.
- When writing an unconditional rule (rule that contains only actions without a conditional part) to the configuration file, an empty conditional part and a separator '`:`' will be added to it.

  For example, the rule that does not contain a conditional part and *consists only of an action*:

  ```
  BLOCK as _match
  ```

  will be written to the configuration file as follows:

  ```
  : BLOCK as _match
  ```

- When writing a rule whose action part contains a set of *multiple* actions to the configuration file, it will be written as a sequence of rules with the same conditional part and one action in the action part in the same order as the actions are listed.

  For example, the rule that contains *two actions* in the action part:

  ```
  user in ('user1', 'user2') : SET http_templates_dir = "/etc/mytemplates", BLOCK as
  _match
  ```

  will be written to the configuration file as the *sequence of two rules*:

  ```
  user in ('user1', 'user2') : SET http_templates_dir = "/etc/mytemplates"
  user in ('user1', 'user2') : BLOCK as _match
  ```

- Since the rule structure does not provide for a disjunction (logical “OR”) of conditions in the conditional part, in order to implement logical “OR”, write a chain of rules with each rule having a single disjunctive condition in its condition.

  For example, the following two rules are equal to the rule “Block upon detection of a malicious object of the `KnownVirus` type or a URL from the `Terrorism` category”:

  ```
  threat_category in (KnownVirus) : BLOCK as _match
  url_category in (Terrorism) : BLOCK as _match
  ```

  since the following records are logically equivalent: $(a \rightarrow x, b \rightarrow x)$; $((a \rightarrow x) \wedge (b \rightarrow x))$; $((a \vee b) \rightarrow x)$.

As for any configuration parameter, values of parameters of the `RuleSet` type (i.e. rules) can be viewed and changed using the `cfshow` and `cfset` commands of the Dr.Web Ctl management tool (the `drweb-ctl` module). For further information about the syntax of these commands, refer to the Dr.Web Ctl section.

10.5. Appendix E. Generating SSL Certificates

For the Dr.Web Gateway Security Suite components that use a secure SSL/TLS data channel and application protocols, such as HTTPS, LDAPS, SMTPS and so on, to exchange data, it is necessary to provide private SSL keys and the corresponding certificates. Keys and certificates for some components are generated automatically; as for the others, they should be provided by a Dr.Web Gateway Security Suite user. All the components use certificates in the PEM format.

To generate private keys and certificates used for connections via SSL/TLS, including verification certificates of Certification Authority (CA) and signed certificates, you can use the `openssl` command-line utility (included in the OpenSSL cryptographic package).

Consider a sequence of actions required for generating a private key and the corresponding SSL certificate together with an SSL certificate signed with a CA verification certificate.

To generate a private SSL key and a certificate

1. To generate a private key (the RSA algorithm, the key length is 2048 bits), run the command:

   ```
   $ openssl genrsa -out keyfile.key 2048
   ```

   If you want to password protect the key, use the `-des3` option. The generated key is in the `keyfile.key` file located in the current directory.

   To view the generated key, run the command:

   ```
   $ openssl rsa -noout -text -in keyfile.key
   ```

2. To generate a certificate for a specified time period based on the existing private key (in this case, for 365 days), run the command:

   ```
   $ openssl req -new -x509 -days 365 -key keyfile.key -out certificate.crt
   ```

   This command will request data (a name, an organization and so on) that identify the certified object. The generated certificate will be located in the `certificate.crt` file.

   To scan the contents of the generated certificate, run the command:

   ```
   $ openssl x509 -noout -text -in certificate.crt
   ```

To register a certificate as a trusted CA certificate

1. Move or copy the certificate file to the system trusted certificate directory (`/etc/ssl/certs` on Debian or Ubuntu).
2. In the trusted certificate directory, create a symbolic link to the certificate, where the name of the link is the hash value of the certificate.
3. Reindex the contents of the system directory containing certificates.

The example commands provided below perform all these three actions. It is assumed that the current directory is the trusted certificate directory `/etc/ssl/certs` and the certificate that is registered as a trusted one is located in the `/home/user/ca.crt` file:

```
# cp /home/user/ca.crt .
# ln -s ca.crt `openssl x509 -hash -noout -in ca.crt`.0
# c_rehash /etc/ssl/certs
```

To create a signed certificate

1. Generate a request file for signing a certificate *(Certificate Signing Request—CSR)* based on an existing private key. If the key is absent, generate it.

   The request for signing is created by running the command:

   ```
   $ openssl req -new -key keyfile.key -out request.csr
   ```

   This command, as well as the command for certificate creation, requests data that identifies the certified object. Here, `keyfile.key` is the existing file of the private key. The received request will be saved to the `request.csr` file.

   To check the result of request creation, run the command:

   ```
   $ openssl req -noout -text -in request.csr
   ```

2. To create a signed certificated based on the request and the existing CA certificate, run the command:

   ```
   $ openssl x509 -req -days 365 -CA ca.crt -CAkey ca.key -set_serial 01 -in request.csr
   -out sigcert.crt
   ```

   To create a signed certificate, you must have the following three files: the file of the root certificate `ca.crt` and its private key `ca.key` (the `certificate.crt` certificate and the `keyfile.key` key may be used istead of `ca.crt` and `ca.key`, then the obtained certificate will be self-signed), as well as the request for signing `request.csr`. The created signed certificate will be saved to the file `sigcert.crt`.

   To check the result, run the command:

   ```
   $ openssl x509 -noout -text -in sigcert.crt
   ```

Repeat the procedure of creating a key and a certificate (or a signed certificate, if necessary) as many times as the number of unique certificates you need to create. For example, from a security point of view, every agent for distributed file scanning by Dr.Web Network Checker within a scanning cluster should have its own key/certificate pair.

To convert a signed certificate

Some browsers or mail clients may require to convert the signed certificate used for authorization to the `PKCS12` format.

Perform such conversion by running the command:

```
# openssl pkcs12 -export -in sigcert.crt -out sigcert.pfx -inkey keyfile.key
```

Here, `sigcert.crt` is the existing file of the signed certificate, `keyfile.key` is the file of the corresponding private key. The resulting converted certificate is saved to the `sigcert.pfx` file.

10.6. Appendix F. Known Errors

In this section

- Recommendations for error identification
- Error codes
- Errors without a code

If the occurred error is not present in this section, it is recommended that you contact our technical support. Be ready to provide the error code and describe steps to reproduce the issue.

Recommendations for error identification

- To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.
- To identify errors, we recommend that you store output of the log in a separate file and enable output of detailed debug information. For that, run the commands:

```
# drweb-ctl cfset Root.Log <log path>
# drweb-ctl cfset Root.DefaultLogLevel DEBUG
```

- To reset the logging settings to defaults, run the commands:

```
# drweb-ctl cfset Root.Log -r
# drweb-ctl cfset Root.DefaultLogLevel -r
```

Error codes

Error message: *Error on monitor channel*

Error code: `x1`

Internally used name: `EC_MONITOR_IPC_ERROR`

Description: One or several components cannot connect to the Dr.Web ConfigD configuration daemon.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Restart the configuration daemon:

   ```
   # service drweb-configd restart
   ```

2. Check whether the authentication mechanism for PAM is installed, configured and operates correctly. If necessary, install and configure it (for details refer to administration manuals for your OS distribution).

3. If PAM is configured correctly and restarting the configuration daemon does not help, reset Dr.Web Gateway Security Suite settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```

   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

4. If it is not possible to start the configuration daemon, reinstall the `drweb-configd` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Operation is already in progress*

Error code: `x2`

Internally used name: `EC_ALREADY_IN_PROGRESS`

Description: The operation is already in progress.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Wait for the operation to finish. If necessary, repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Operation is in pending state*

Error code: `x3`

Internally used name: `EC_IN_PENDING_STATE`

Description: An operation requested by the user is in a pending state (possibly, a network connection is currently being established or one of the components is starting and initializing, which may take a long time).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less`

`/var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Wait for the operation to start. If necessary, repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Interrupted by user*

Error code: `x4`

Internally used name: `EC_INTERRUPTED_BY_USER`

Description: The action has been terminated by the user (possibly, it was taking a long time).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Operation canceled*

Error code: `x5`

Internally used name: `EC_CANCELED`

Description: The action has been canceled.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action.

If the error persists, contact our technical support and provide the error code.

Error message: *IPC connection terminated*

Error code: `x6`

Internally used name: `EC_LINK_DISCONNECTED`

Description: An IPC connection to one of the Dr.Web Gateway Security Suite components has been terminated (possibly, the component was shut down due to being idle or by a user request).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security

Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

If the operation did not finish, repeat it later. Otherwise, the connection termination is not an error.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid IPC message size*

Error code: `x7`

Internally used name: `EC_BAD_MESSAGE_SIZE`

Description: A message of an invalid size has been received during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Gateway Security Suite:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid IPC message format*

Error code: `x8`

Internally used name: `EC_BAD_MESSAGE_FORMAT`

Description: A message of an invalid format has been received during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Gateway Security Suite:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Not ready*

Error code: x9

Internally used name: EC_NOT_READY

Description: The required action cannot be performed because the necessary component or device has not been initialized yet.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Component is not installed*

Error code: x10

Internally used name: EC_NOT_INSTALLED

Description: The component necessary for running the required function is not installed.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Install or reinstall the necessary component. If you do not know the component name, try to determine it from the log file.
2. If the installation or reinstallation of the necessary component does not help, reinstall Dr.Web Gateway Security Suite.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected IPC message*

Error code: x11

Internally used name: EC_UNEXPECTED_MESSAGE

Description: An invalid message has been received during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Gateway Security Suite by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *IPC protocol violation*

Error code: `x12`

Internally used name: `EC_PROTOCOL_VIOLATION`

Description: A protocol violation has occurred during component inter-process communication (IPC).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Gateway Security Suite by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Subsystem state is unknown*

Error code: `x13`

Internally used name: `EC_UNKNOWN_STATE`

Description: The required operation cannot be performed because a subsystem of Dr.Web Gateway Security Suite is in an unknown state.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the operation.
2. If the error persists, restart Dr.Web Gateway Security Suite by running the command:

   ```
   # service drweb-configd restart
   ```

   and repeat the operation afterwards.

If the error persists, contact our technical support and provide the error code.

Error message: *Path must be absolute*

Error code: x20

Internally used name: EC_NOT_ABSOLUTE_PATH

Description: A relative path to a file or a directory has been specified, whereas an absolute path is required.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Make the file or directory path absolute and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Not enough memory*

Error code: x21

Internally used name: EC_NO_MEMORY

Description: Not enough memory to complete the requested operation.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Increase the size of the memory available for Dr.Web Gateway Security Suite processes (for example, by changing the limits using the `ulimit` command), restart Dr.Web Gateway Security Suite and repeat the operation.



In some cases the `systemd` system service can ignore the specified limit changes. In this case, edit (or create if it does not exist) the file `/etc/systemd/system/drweb-configd.service.d/limits.conf` and indicate the changed limit value in it, for example:

```
[Service]
LimitDATA=32767
```

Available `systemd` limits can be determined using the `man systemd.exec` command.

Restart Dr.Web Gateway Security Suite by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *I/O error*

Error code: x22

Internally used name: `EC_IO_ERROR`

Description: An input/output error has occurred (for example, a disk device has not been initialized yet or a partition of the file system is not available anymore).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Ensure the availability of the required I/O device or partition of the file system and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *No such file or directory*

Error code: `x23`

Internally used name: `EC_NO_SUCH_ENTRY`

Description: An attempt to access a non-existent file or directory has been made.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the indicated path. If necessary, correct it and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Permission denied*

Error code: `x24`

Internally used name: `EC_PERMISSION_DENIED`

Description: Insufficient privileges to access the specified file or directory.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Ensure that the path is correct and the component has the required privileges. If access to the object is denied, change its permissions or elevate component privileges and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Not a directory*

Error code: x25

Internally used name: EC_NOT_A_DIRECTORY

Description: The specified object of the file system is not a directory.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the object path. Specify the correct path and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Data file corrupted*

Error code: x26

Internally used name: EC_DATA_CORRUPTED

Description: The requested data is corrupted.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the operation.
2. If the error persists, restart Dr.Web Gateway Security Suite:

   ```
   # service drweb-configd restart
   ```

   and repeat the operation afterwards.

If the error persists, contact our technical support and provide the error code.

Error message: *File already exists*

Error code: x27

Internally used name: EC_FILE_EXISTS

Description: A file with the same name already exists.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the file path. If necessary, correct it and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Read-only file system*

Error code: `x28`

Internally used name: `EC_READ_ONLY_FS`

Description: The file system is read-only.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the object path. Change it so as to indicate a writable partition of the file system and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Network error*

Error code: `x29`

Internally used name: `EC_NETWORK_ERROR`

Description: A network error has occurred (possibly, a remote host stopped responding unexpectedly or required connection failed).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check that the network is available and network settings are correct. If necessary, change the network settings and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Not a drive*

Error code: `x30`

Internally used name: `EC_NOT_A_DRIVE`

Description: The input/output device being accessed is not a disk device.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified device name. Correct the path so as to indicate a disk device and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected EOF*

Error code: `x31`

Internally used name: `EC_UNEXPECTED_EOF`

Description: The end of the file has been reached enexpectedly while reading data.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified file name. If necessary, correct the path so as to indicate the correct file and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *File was changed*

Error code: `x32`

Internally used name: `EC_FILE_WAS_CHANGED`

Description: The file being scanned has been modified.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat scanning.

If the error persists, contact our technical support and provide the error code.

Error message: *Not a regular file*

Error code: `x33`

Internally used name: `EC_NOT_A_REGULAR_FILE`

Description: The file system object being accessed is not a regular file (it may a directory, a socket and so on).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified file name. If necessary, change the path so as to indicate a regular file and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Name already in use*

Error code: `x34`

Internally used name: `EC_NAME_ALREADY_IN_USE`

Description: A file with the same name already exists.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the specified path. Correct it and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Host is offline*

Error code: `x35`

Internally used name: `EC_HOST_OFFLINE`

Description: A remote host cannot be accessed over the network.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check that the required host is available. If necessary, correct the host address and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Resource limit reached*

Error code: `x36`

Internally used name: `EC_LIMIT_REACHED`

Description: A limit on resource usage has been reached.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less`

`/var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check the availability of the required resource. If necessary, raise the limit on the usage of this resource and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Mounting points are different*

Error code: `x37`

Internally used name: `EC_CROSS_DEVICE_LINK`

Description: Restoring the file implies moving it between two different mount points.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Choose another path for restoring the file and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Unpacking error*

Error code: `x38`

Internally used name: `EC_UNPACKING_ERROR`

Description: Unable to unpack an archive (it may be password-protected or corrupted).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Make sure that the archive is not corrupted. If the archive is protected with a password, remove the protection having entered the correct password and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Virus database corrupted*

Error code: `x40`

Internally used name: `EC_BASE_CORRUPTED`

Description: Virus databases are corrupted.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Non-supported virus database version*

Error code: `x41`

Internally used name: `EC_OLD_BASE_VERSION`

Description: Current virus databases are intended for an earlier program version.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset Root.VirusBaseDir -r
```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Empty virus database*

Error code: `x42`

Internally used name: `EC_EMPTY_BASE`

Description: The virus database is empty.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir -r
   ```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Object cannot be cured*

Error code: `x43`

Internally used name: `EC_CAN_NOT_BE_CURED`

Description: The **Cure** action has been applied to an incurable object.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Select an action that can be applied to this object and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Non-supported virus database combination*

Error code: `x44`

Internally used name: `EC_INVALID_BASE_SET`

Description: Incompatible virus databases.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Scan limit reached*

Error code: `x45`

Internally used name: `EC_SCAN_LIMIT_REACHED`

Description: The specified limits have been exceeded during object scanning (for example, on the size of an unpacked file, on the nesting depth and so on).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Change the scanning limits (in the component settings) using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands.
2. After changing the settings, repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Authentication failed*

Error code: `x47`

Internally used name: `EC_AUTH_FAILED`

Description: Invalid user credentials have been used for authentication.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Provide valid credentials of a user having required privileges and try to authenticate again.

If the error persists, contact our technical support and provide the error code.

Error message: *Authorization failed*

Error code: `x48`

Internally used name: `EC_NOT_AUTHORIZED`

Description: The current user has insufficient privileges for performing the requested operation.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Provide valid credentials of a user having required privileges and try to authorize again.

If the error persists, contact our technical support and provide the error code.

Error message: *Access token is invalid*

Error code: `x49`

Internally used name: `EC_INVALID_TOKEN`

Description: A component of Dr.Web Gateway Security Suite has provided an invalid authorization token in an attempt to execute an operation requiring elevated privileges.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Authenticate by providing valid credentials of a user having required privileges and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *DBMS general error*

Error code: `x50`

Internally used name: `EC_DB_COMMON_ERROR`

Description: A request to the DBMS made by Dr.Web LookupD was not successfull.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Refer to DMBS server logs as well.

If the error persists, contact our technical support and provide the error code.

Error message: *Cannot open database*

Error code: `x51`

Internally used name: `EC_DB_OPEN_ERROR`

Description: The database to which Dr.Web LookupD is trying to connect is unavailable.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Refer to DMBS server logs as well.

If the error persists, contact our technical support and provide the error code.

Error message: *Connection closed by DBMS*

Error code: x52

Internally used name: EC_DB_CONN_CLOSED

Description: The connection has been closed by the DBMS server.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Refer to DMBS server logs as well.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid argument*

Error code: x60

Internally used name: EC_INVALID_ARGUMENT

Description: Unable to run the command since an invalid argument has been provided.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the command syntax and format.
2. Repeat the required action having indicated a valid argument.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid operation*

Error code: x61

Internally used name: EC_INVALID_OPERATION

Description: An attempt to run an invalid command has been made.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action using a valid command.

If the error persists, contact our technical support and provide the error code.

Error message: *Superuser privileges required*

Error code: x62

Internally used name: EC_ROOT_ONLY

Description: Superuser privileges are required to perform the required action.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Elevate you privileges to superuser ones and repeat the required action. To elevate the privileges, use the `su` or `sudo` command.

If the error persists, contact our technical support and provide the error code.

Error message: *Not allowed in centralized protection mode*

Error code: x63

Internally used name: EC_STANDALONE_MODE_ONLY

Description: The required action can be performed only when Dr.Web Gateway Security Suite operates in standalone mode.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Switch Dr.Web Gateway Security Suite to the standalone mode and repeat the operation.

To switch Dr.Web Gateway Security Suite to the standalone mode:

- clear the **Enable centralized protection mode** check box on the **Centralized protection** of the management web interface;
- or run the command:

```
# drweb-ctl esdisconnect
```

If the error persists, contact our technical support and provide the error code.

Error message: *Non-supported OS*

Error code: x64

Internally used name: EC_NON_SUPPORTED_OS

Description: Dr.Web Gateway Security Suite does not support the operating system installed on the host.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Install an operating system from the list provided in system requirements.

If the error persists, contact our technical support and provide the error code.

Error message: *Feature not implemented*

Error code: `x65`

Internally used name: `EC_UNKNOWN_OPTION`

Description: The required features of the component are not implemented in the current version.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Try to reset the settings of Dr.Web Gateway Security Suite to defaults.

To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

- for GNU/Linux:

```
# cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
# echo "" > /etc/opt/drweb.com/drweb.ini
```

- for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Unknown option*

Error code: `x66`

Internally used name: `EC_UNKNOWN_OPTION`

Description: The configuration file contains parameters that are unknown to or not supported by the current version of Dr.Web Gateway Security Suite.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Open the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD in any text editor, remove the line containing the invalid parameter. Save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

2. If this does not help, reset Dr.Web Gateway Security Suite settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```

   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

If the error persists, contact our technical support and provide the error code.

Error message: *Unknown section*

Error code: `x67`

Internally used name: `EC_UNKNOWN_SECTION`

Description: The configuration file contains sections unknown to or not supported by the current version of Dr.Web Gateway Security Suite.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Open the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD in any text editor, remove the unknown (unsupported) section. Save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

2. If this does not help, reset Dr.Web Gateway Security Suite settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

```
# cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
# echo "" > /etc/opt/drweb.com/drweb.ini
```

- for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid option value*

Error code: `x68`

Internally used name: `EC_INVALID_OPTION_VALUE`

Description: One or more parameters in the configuration file have invalid values.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Set the parameter value using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands.

   If you do not know which value is valid for the parameter, refer to a man page for the component which uses this parameter. You can also restore a default value of the parameter.
2. You can also directly edit the configuration file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD. To do this, open it in any text editor, find the line containing an invalid parameter value, set a valid value, then save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```
3. If the previous steps did not help, reset Dr.Web Gateway Security Suite settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:
   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```
   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the

command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid state*

Error code: `x69`

Internally used name: `EC_INVALID_STATE`

Description: Dr.Web Gateway Security Suite or one of its components is in an invalid state and cannot complete the required operation.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the required action later.
2. If the error persists, restart Dr.Web Gateway Security Suite by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Only one value allowed*

Error code: `x70`

Internally used name: `EC_NOT_LIST_OPTION`

Description: A list of values is set for a single-valued parameter in the configuration file.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Set the parameter value using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands.

   If you do not know which value is valid for the parameter, refer to a man page for the component which uses this parameter. You can also restore a default value of the parameter.
2. You can also directly edit the configuration file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD. To do this, open it in any text editor, find the line containing an invalid parameter value, set a valid value, then save the file and restart the Dr.Web ConfigD configuration daemon by running the command:

```
# service drweb-configd restart
```

3. If the previous steps did not help, reset Dr.Web Gateway Security Suite settings to defaults.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

     ```
     # cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
     # echo "" > /etc/opt/drweb.com/drweb.ini
     ```

   - for FreeBSD:

     ```
     # cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
     # echo "" > /usr/local/etc/drweb.com/drweb.ini
     ```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

   ```
   # service drweb-configd restart
   ```

If the error persists, contact our technical support and provide the error code.

Error message: *Tag value is invalid*

Error code: `x71`

Internally used name: `EC_INVALID_TAG`

Description: An invalid or inexistent tag has been used in the name of a data source with which the Dr.Web LookupD component interacts.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check that the tag is spelled correctly. If you find an error, edit the appropriate section of the configuration file.

If the error persists, contact our technical support and provide the error code.

Error message: *Record not found*

Error code: `x80`

Internally used name: `EC_RECORD_NOT_FOUND`

Description: The threat record is missing (it may have already been proccessed by another component).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update the threat list later.

If the error persists, contact our technical support and provide the error code.

Error message: *Record is in process now*

Error code: `x81`

Internally used name: `EC_RECORD_BUSY`

Description: The threat is already being processed by another component.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update the threat list later.

If the error persists, contact our technical support and provide the error code.

Error message: *File has already been quarantined*

Error code: `x82`

Internally used name: `EC_QUARANTINED_FILE`

Description: The file is already in quarantine. Probably, another component has already processed the threat.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update the threat list later.

If the error persists, contact our technical support and provide the error code.

Error message: *Update zone is not provided by cloud*

Error code: `x83`

Internally used name: `EC_NO_ZONE_IN_CLOUD`

Description: An attempt to update using Dr.Web Cloud was unsuccessful.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Repeat the required action later.

If the error persists, contact our technical support and provide the error code.

Error message: *Update zone is not provided on disk*

Error code: `x84`

Internally used name: `EC_NO_ZONE_ON_DISK`

Description: An attempt to update the virus bases in offline mode was unsuccessful.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Make sure that the path to the device used for updating is correct.
2. Make sure that the user as whom you are trying to update has read permissions for the directory with updates.

If the error persists, contact our technical support and provide the error code.

Error message: *Cannot backup before update*

Error code: `x89`

Internally used name: `EC_BACKUP_FAILED`

Description: Prior to downloading updates from an update server, an attempt to make a backup copy of the files to be updated has failed.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the directory that stores backup copies of the files to be updated. Change the path, if necessary (the `BackupDir` parameter in the `[Update]` section of the configuration file).
   - To view and change the path, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Update.BackupDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Update.BackupDir <new path>
   ```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset Update.BackupDir -r
```

2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```
3. If the error persists, check whether the user as whom the component is running has write permissions for a directory specified in the `BackupDir` parameter. The name of this user is specified in the `RunAsUser` parameter. If necessary, change the user specified in the `RunAsUser` parameter or grant the lacking permissions in the directory attributes.
4. If the error persists, reinstall the `drweb-update` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid DRL file*

Error code: `x90`

Internally used name: `EC_BAD_DRL_FILE`

Description: The integrity of one of the files storing a list of update servers is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the file storing the list of servers and change the path if necessary (parameters containing `*DrlDir` in the `[Update]` section of the configuration file).
   - To view and change the path, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To view the current parameter value, run the command (replace <**DrlDir*> with a specific parameter name. If the parameter name is unknown, view all parameter values in the section, skipping the command part in square brackets):

   ```
   $ drweb-ctl cfshow Update[.<*DrlDir>]
   ```

   To set a new parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

   ```
   # drweb-ctl cfset Update.<*DrlDir> <new path>
   ```

   To restore the default parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

   ```
   # drweb-ctl cfset Update.<*DrlDir> -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;

- or run the command:

```
$ drweb-ctl update
```

3. If the error persists, reinstall the `drweb-update` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid LST file*

Error code: `x91`

Internally used name: `EC_BAD_LST_FILE`

Description: The integrity of the file storing a list of virus databases to be updated is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Update virus databases again after some time:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

```
$ drweb-ctl update
```

2. If the error persists, reinstall the `drweb-update` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid compressed file*

Error code: `x92`

Internally used name: `EC_BAD_LZMA_FILE`

Description: The integrity of the downloaded file containing updates is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update virus databases again after some time:

- click **Update** on the **Main** page of the management web interface;
- or run the command:

```
$ drweb-ctl update
```

If the error persists, contact our technical support and provide the error code.

Error message: *Proxy authentication error*

Error code: `x93`

Internally used name: `EC_PROXY_AUTH_ERROR`

Description: Unable to connect to update servers via the proxy server specified in the settings.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the parameters used to connect to the proxy server (set with the `Proxy` parameter in the `[Update]` section of the configuration file).
   - To view and set the connection parameters, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Update.Proxy
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Update.Proxy <new parameters>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Update.Proxy -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *No update servers available*

Error code: `x94`

Internally used name: `EC_NO_UPDATE_SERVERS`

Description: Unable to connect to any of the update servers.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check if the network is available. Change network settings if necessary.
2. If you can access the network only using a proxy server, configure connection to the proxy server (the `Proxy` parameter in the `[Update]` section of the configuration file).
   - To view and set the connection parameters, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Update.Proxy
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Update.Proxy <new parameters>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Update.Proxy -r
   ```
3. If the network connection parameters (including the parameters of the proxy server) are correct, but the error persists, make sure that you use a relevant list of update servers. The list of used update servers is set in `*DrlDir` parameters in the `[Update]` section of the configuration file.

   If `*CustomDrlDir` parameters indicate an existing valid file storing a list of servers, the servers specified there will be used instead of the servers of the standard update zone (the value specified in the corresponding `*DrlDir` parameter is ignored).

   - To view and set the connection parameters, go to the **Updater** page of the management web interface.
   - You can also use the command-line management tool.

   To view the current parameter value, run the command (replace <**DrlDir*> with a specific parameter name. If the parameter name is unknown, view all parameter values in the section, skipping the command part in square brackets):

   ```
   $ drweb-ctl cfshow Update[.<*DrlDir>]
   ```

   To set a new parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

   ```
   # drweb-ctl cfset Update.<*DrlDir> <new path>
   ```

   To restore the default parameter value, run the command (replace <**DrlDir*> with a specific parameter name):

   ```
   # drweb-ctl cfset Update.<*DrlDir> -r
   ```
4. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid key file format*

Error code: x95

Internally used name: EC_BAD_KEY_FORMAT

Description: The key file format is violated.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check that you have a key file and a correct path to it. You can specify the path to the key file in the `KeyPath` parameter in the `[Root]` section of the configuration file.
   - To view and set the path to the key file, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.KeyPath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.KeyPath <path to file>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.KeyPath -r
   ```
2. If you do not have a key file or the key file being used is corrupted, purchase and install it. For more details on the key file, purchasing and installing it, refer to the Licensing section.
3. To install the key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
4. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *License is already expired*

Error code: x96

Internally used name: EC_EXPIRED_KEY

Description: The license has expired.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Purchase a new license and install a received key file. For more details on how to purchase the license and install the key file, refer to the Licensing section.
2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Network operation timed out*

Error code: `x97`

Internally used name: `EC_NETWORK_TIMEDOUT`

Description: A network connection timed out (possibly, a remote host has stopped responding unexpectedly or the required connection has failed).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Check that the network is available and network settings are correct. If necessary, change the network settings and repeat the operation.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid checksum*

Error code: `x98`

Internally used name: `EC_BAD_CHECKSUM`

Description: The checksum of the downloaded file with updates is invalid.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Update virus databases again after some time:

- click **Update** on the **Main** page of the management web interface;
- or run the command:

```
$ drweb-ctl update
```

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid trial license*

Error code: `x99`

Internally used name: `EC_BAD_TRIAL_KEY`

Description: The demo key file is invalid (for example, it was received for another computer).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security

Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Request a new demo period for this computer or purchase a new license and install a received key file. For more details on how to purchase the license and install the key file, refer to the Licensing section.
2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Blocked license key*

Error code: `x100`

Internally used name: `EC_BLOCKED_LICENSE`

Description: The current license is blocked (the terms of use of Dr.Web Gateway Security Suite may be violated).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Purchase a new license and install a received key file. For more details on how to purchase the license and install the key file, refer to the Licensing section.
2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid license*

Error code: `x101`

Internally used name: `EC_BAD_LICENSE`

Description: The license is intended for another product or does not cover Dr.Web Gateway Security Suite components.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Purchase a new license and install a received key file. For more details on how to purchase the

license and install the key file, refer to the Licensing section.

2. To install the received key file, you can use the license activation form at the bottom of the **Main** page of the management web interface.
3. You can view current license options on the **My Dr.Web** user webpage.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid configuration*

Error code: `x102`

Internally used name: `EC_BAD_CONFIG`

Description: A component of Dr.Web Gateway Security Suite cannot operate because of invalid configuration settings.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. If you do not know the name of the component that causes the error, try to determine it by viewing the log file.

If the error persists, contact our technical support and provide the error code.

Error message: *Invalid executable file*

Error code: `x104`

Internally used name: `EC_BAD_EXECUTABLE`

Description: Failed to start the component. The executable file is corrupted or the file path is invalid.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. If you do not know the name of the component that causes the error, try to determine it by viewing the log file.
2. Check the executable path to the component in the Dr.Web Gateway Security Suite configuration file (the `ExePath` parameter in the component section) by running the command (replace <*component section*> with the name of the corresponding section of the configuration file):

   ```
   $ drweb-ctl cfshow <component section>.ExePath
   ```

3. Reset the path to its default value by running the command (replace <*component section*> with the name of the corresponding section of the configuration file):

   ```
   # drweb-ctl cfset <component section>.ExePath -r
   ```

4. If the previous steps did not help, reinstall the package of the corresponding component.

For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Core engine is not available*

Error code: `x105`

Internally used name: `EC_NO_CORE_ENGINE`

Description: The file of Dr.Web Virus-Finding Engine is missing or unavailable.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the anti-virus engine file `drweb32.dll` and, if necessary, correct it (the `CoreEnginePath` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.CoreEnginePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.CoreEnginePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.CoreEnginePath -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```
3. If the path is correct and the error persists after updating virus databases, reinstall the `drweb-bases` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *No virus databases*

Error code: `x106`

Internally used name: `EC_NO_VIRUS_BASES`

Description: Virus databases are missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the virus database directory. Change the path, if necessary (the `VirusBaseDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.VirusBaseDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.VirusBaseDir -r
   ```
2. Update virus databases:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

     ```
     $ drweb-ctl update
     ```

If the error persists, contact our technical support and provide the error code.

Error message: *Process terminated by signal*

Error code: `x107`

Internally used name: `EC_APP_TERMINATED`

Description: A component has been shut down (possibly, owing to being idle or by a user request).

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. If the operation has not finished, repeat it. Otherwise, the shutdown is not an error.
2. If the component shuts down constantly, reset its settings to default using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands;
   - edit the configuration file manually (delete all parameters from the component section).
3. If this did not help, reset Dr.Web Gateway Security Suite settings to default.

To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

- for GNU/Linux:

```
# cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
# echo "" > /etc/opt/drweb.com/drweb.ini
```

- for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected process termination*

Error code: `x108`

Internally used name: `EC_APP_CRASHED`

Description: A component has been shut down because of a failure.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Repeat the terminated operation.
2. If the component constantly shuts down abnormally, reset its settings to default using any of the following methods:
   - on the page with the component settings using the management web interface;
   - using the `drweb-ctl cfshow` and `drweb-ctl cfset` commands;
   - edit the configuration file manually (delete all parameters from the component section).
3. If this did not help, reset Dr.Web Gateway Security Suite settings to default.

   To do this, clear the contents of the file `/etc/opt/drweb.com/drweb.ini` for GNU/Linux or `/usr/local/etc/drweb.com/drweb.ini` for FreeBSD (it is recommended that you make a backup of the configuration file), for example, by running the commands:

   - for GNU/Linux:

```
# cp /etc/opt/drweb.com/drweb.ini /etc/opt/drweb.com/drweb.ini.save
# echo "" > /etc/opt/drweb.com/drweb.ini
```

   - for FreeBSD:

```
# cp /usr/local/etc/drweb.com/drweb.ini /usr/local/etc/drweb.com/drweb.ini.save
# echo "" > /usr/local/etc/drweb.com/drweb.ini
```

   After clearing the contents of the configuration file, restart the configuration daemon by running the command:

```
# service drweb-configd restart
```

4. If the error persists after resetting Dr.Web Gateway Security Suite settings, reinstall the component package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Incompatible software detected*

Error code: `x109`

Internally used name: `EC_INCOMPATIBLE`

Description: One or several components of Dr.Web Gateway Security Suite cannot operate properly. Their operation is impeded by software in your system.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. If this error is caused by SpIDer Gate, the issue may be related to software generating rules for the NetFilter system firewall, which prevent SpIDer Gate from operating properly, such as Shorewall or SuseFirewall2 (on SUSE Linux). Such applications recurrently check the integrity of rule sets specified by them and rewrite these rules.

   Reconfigure conflicting software so that it does not interfere in SpIDer Gate operation. If unsuccessful, disable the software so that it does not load at the operating system startup. You can try to configure SuseFirewall2 (on SUSE Linux) as follows:

   1) open the configuration file of SuseFirewall2 (by default, it is `/etc/sysconfig/SuSEfirewall2`);

   2) find the following lines in the file:

```
## Type: yesno
#
# Install NOTRACK target for interface lo in the raw table. Doing so
# speeds up packet processing on the loopback interface. This breaks
# certain firewall setups that need to e.g. redirect outgoing
# packets via custom rules on the local machine.
#
# Defaults to "yes" if not set
#
FW_LO_NOTRACK=""
```

   3) Set the `FW_LO_NOTRACK` parameter value to `no`:

```
FW_LO_NOTRACK="no"
```

   4) Restart SuseFirewall2:

```
# rcSuSEfirewall2 restart
```

> If there is no `FW_LO_NOTRACK` parameter in SuseFirewall2 settings, stop this application and disable its launch at system startup to deter conflict.

After reconfiguring or disabling the conflicting application, restart SpIDer Gate.

2. If the error is caused by another component, disable or reconfigure the conflicting software such as to prevent any interference in the Dr.Web Gateway Security Suite operation.

If the error persists, contact our technical support and provide the error code.

Error message: *No web resource databases*

Error code: `x112`

Internally used name: `EC_NO_DWS_BASES`

Description: Databases of web resource categories are missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path directory to a database of web resource categories. Change the path if necessary (the `DwsDir` parameter in the `[Root]` section of the configuration file).
   - To view and change the path, go to the **General Settings** page of the management web interface.
   - You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Root.DwsDir
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Root.DwsDir <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Root.DwsDir -r
   ```

2. Update databases of web resource categories:
   - click **Update** on the **Main** page of the management web interface;
   - or run the command:

   ```
   $ drweb-ctl update
   ```

If the error persists, contact our technical support and provide the error code.

Error message: *MeshD is not available*

Error code: `x114`

Internally used name: `EC_NO_MESHD`

Description: The Dr.Web MeshD component required for load balancing during file scanning is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-meshd` executable file. Change the path if necessary (the `ExePath` parameter in the `[MeshD]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow MeshD.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset MeshD.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset MeshD.ExePath -r
   ```

2. If the configuration does not contain Dr.Web MeshD settings or the error persists after entering the correct path, install or reinstall the `drweb-meshd` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *LookupD is not available*

Error code: `x115`

Internally used name: `EC_NO_LOOKUPD`

Description: The Dr.Web LookupD component required for retrieving data from external sources is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

1. Check the path to the `drweb-lookupd` executable file. Change the path if necessary (the `ExePath` parameter in the `[LookupD]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow LookupD.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset LookupD.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset LookupD.ExePath -r
   ```

2. If the configuration does not contain Dr.Web LookupD settings or the error persists after entering the correct path, install or reinstall the `drweb-lookupd` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *UrlCheck is not available*

Error code: `x116`

Internally used name: `EC_NO_URL_CHECK`

Description: The Dr.Web URL Checker component required for checking URLs for belonging to blocked or potentially dangerous categories is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-urlcheck` executable file. Change the path if necessary (the `ExePath` parameter in the `[URLCheck]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow URLCheck.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset URLCheck.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset URLCheck.ExePath -r
   ```

2. If the configuration does not contain Dr.Web URL Checker settings or the error persists after entering the correct path, install or reinstall the `drweb-urlcheck` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *GateD is not available*

Error code: `x117`

Internally used name: `EC_NO_GATED`

Description: The SpIDer Gate component required for scanning network connections is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-gated` executable file. Change the path if necessary (the `ExePath` parameter in the `[GateD]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow GateD.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset GateD.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset GateD.ExePath -r
   ```

2. If the configuration does not contain SpIDer Gate settings or the error persists after entering the correct path, install or reinstall the `drweb-gated` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *ScanEngine is not available*

Error code: `x119`

Internally used name: `EC_NO_SCAN_ENGINE`

Description: The Dr.Web Scanning Engine component required for threat detection is missing or has failed to start.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-se` executable file. Change the path if necessary (the `ExePath` parameter in the `[ScanEngine]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow ScanEngine.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset ScanEngine.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset ScanEngine.ExePath -r
   ```

2. If the error persists after entering the correct path:

   - Run the command:

     ```
     $ drweb-ctl rawscan /
     ```

If the line "`Error: No valid license provided`" is output, a valid key file is missing. Register Dr.Web Gateway Security Suite and receive a license. After receiving the license, check whether the key file is available and install it if necessary.

- If your operating system uses SELinux, configure the security policy for the `drweb-se` module (see the section Configuring SELinux Security Policies).

3. If the configuration does not contain Dr.Web Scanning Engine settings or the previous steps did not help, install or reinstall the `drweb-se` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *FileCheck is not available*

Error code: `x120`

Internally used name: `EC_NO_FILE_CHECK`

Description: The Dr.Web File Checker component is missing or has failed to start.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-filecheck` executable file. Change the path if necessary (the `ExePath` parameter in the `[FileCheck]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow FileCheck.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset FileCheck.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset FileCheck.ExePath -r
   ```

2. If the error persists after entering the correct path:
   - If your operating system uses SELinux, configure the security policy for the `drweb-filecheck` module (see the section Configuring SELinux Security Policies).
3. If the configuration does not contain Dr.Web File Checker settings or the previous steps did not help, install or reinstall the `drweb-filecheck` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *ESAgent is not available*

Error code: x121

Internally used name: EC_NO_ESAGENT

Description: The Dr.Web ES Agent component required to connect to a centralized protection server is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-esagent` executable file. Change the path if necessary (the `ExePath` parameter in the `[ESAgent]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow ESAgent.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset ESAgent.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset ESAgent.ExePath -r
   ```

2. If the configuration does not contain Dr.Web ES Agent settings or the error persists after entering the correct path, install or reinstall the `drweb-esagent` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Firewall is not available*

Error code: x122

Internally used name: EC_NO_FIREWALL

Description: The Dr.Web Firewall for Linux component required for scanning network connections is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-firewall` executable file. Change the path if necessary (the `ExePath` parameter in the `[LinuxFirewall]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow LinuxFirewall.ExePath
   ```

To set a new parameter value, run the command:

```
# drweb-ctl cfset LinuxFirewall.ExePath <new path>
```

To reset the parameter to its default value, run the command:

```
# drweb-ctl cfset LinuxFirewall.ExePath -r
```

2. If the configuration does not contain Dr.Web Firewall for Linux settings or the error persists after entering the correct path, install or reinstall the `drweb-firewall` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *NetCheck is not available*

Error code: `x123`

Internally used name: `EC_NO_NETCHECK`

Description: The Dr.Web Network Checker component required to scan files over the network is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-netcheck` executable file. Change the path if necessary (the `ExePath` parameter in the `[Netcheck]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow Netcheck.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset Netcheck.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset Netcheck.ExePath -r
   ```

2. If the configuration does not contain Dr.Web Network Checker settings or the error persists after entering the correct path, install or reinstall the `drweb-netcheck` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *CloudD is not available*

Error code: x124

Internally used name: EC_NO_CLOUDD

Description: The Dr.Web CloudD component required for making requests to the Dr.Web Cloud service is missing.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

1. Check the path to the `drweb-cloudd` executable file. Change the path if necessary (the `ExePath` parameter in the `[CloudD]` section of the configuration file).

   You can also use the command-line management tool.

   To get the current parameter value, run the command:

   ```
   $ drweb-ctl cfshow CloudD.ExePath
   ```

   To set a new parameter value, run the command:

   ```
   # drweb-ctl cfset CloudD.ExePath <new path>
   ```

   To reset the parameter to its default value, run the command:

   ```
   # drweb-ctl cfset CloudD.ExePath -r
   ```

2. If the configuration does not contain Dr.Web CloudD settings or the error persists after entering the correct path, install or reinstall the `drweb-cloudd` package.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support and provide the error code.

Error message: *Unexpected error*

Error code: x125

Internally used name: EC_UNEXPECTED_ERROR

Description: An unexpected error has occurred in operation of one or several components.

To identify a possible cause and the circumstances of the error, refer to the Dr.Web Gateway Security Suite log (by default, it can be viewed with the `# journalctl` command for GNU/Linux or the `# less /var/log/messages` command for FreeBSD). You can also run the `# drweb-ctl log` command.

Troubleshooting

Restart Dr.Web Gateway Security Suite by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support and provide the error code.

Errors without a code

Symptoms: One of the following errors occurs on FreeBSD 13, 14 or 15:

- Dr.Web ConfigD component fails to start.
- Paths to sockets specified by parameters `Root.AdminSocketPath` and `Root.AdminSocketPath` (by default, `/var/run/.com.drweb.public` and `/var/run/.com.drweb.admin` correspondingly) do not exist or these sockets are unavailable.

Description: Dr.Web Gateway Security Suite operation is impossible because the `compat12x` compatibility package is not installed.

Troubleshooting

Install the `compat12x` compatibility package. An example of how to do this is provided below.

1. Start a console or a terminal emulator and run the commands:

   ```
   # pkg update
   # pkg search compat12x
   ```

   The output of the last command will provide the full package name from your OS repository.
2. Specify the package name and install it, for example:

   ```
   # pkg install compat12x-amd64-12.2.1202000.20210406_2
   ```

If the error persists, contact our technical support.

Symptoms: Such components as Dr.Web ClamD, SpIDer Gate, Dr.Web ICAPD do not scan messages; Dr.Web Gateway Security Suite log indicates `Too many open files`.

Description: Owing to a heavy load while scanning data, Dr.Web Network Checker has reached the limit on the number of available file descriptors.

Troubleshooting

1. Raise the limit of the number of open file descriptors available to the application by running the `ulimit -n` command (the default limit on the number of descriptors for Dr.Web Gateway Security Suite is 16384).

   The `systemd` system service can ignore the specified limit changes in some cases.

   In these situations, edit the file `/etc/systemd/system/drweb-configd.service.d/limits.conf` (or create it if it does not exist) and specify the changed limit value, for example:

   ```
   [Service]
   LimitNOFILE=16384
   ```

   Available `systemd` limits can be determined using the `man systemd.exec` command.

2. Once the limit is changed, restart Dr.Web Gateway Security Suite by running the command:

```
# service drweb-configd restart
```

If the error persists, contact our technical support.

Symptoms: A web browser cannot establish connection to the Dr.Web management web interface, Dr.Web components are not in the list of running processes (`ps ax | grep drweb`), an attempt to run any command such as `drweb-ctl <command>`, except for `drweb-ctl rawscan`, results in one of the following errors:

`Error: connect: No such file or directory: "<path>/.com.drweb.public"`

or

`Error: connect: Connection refused: "<path>/.com.drweb.public".`

Description: Dr.Web Gateway Security Suite cannot start as the Dr.Web ConfigD configuration daemon is not available.

Troubleshooting

1. Run the command:

```
# service drweb-configd restart
```

   to restart Dr.Web ConfigD and Dr.Web Gateway Security Suite in general.
2. If this command returns an error or has no effect, install the `drweb-configd` package separately.

   This also may mean that PAM authentication is not used in the system. If so, install and configure PAM since Dr.Web Gateway Security Suite cannot operate correctly without it.

3. If the error persists, uninstall Dr.Web Gateway Security Suite entirely and then reinstall it.

   For details on how to install and uninstall Dr.Web Gateway Security Suite or its components, refer to sections Installing Dr.Web Gateway Security Suite and Uninstalling Dr.Web Gateway Security Suite.

If the error persists, contact our technical support.

Symptoms

1. After disabling SpIDer Gate, all network connections (both outgoing and, possibly, incoming via the SSH and FTP) stop working.
2. Searching through NetFilter (`iptables`) rules using the command:

```
# iptables-save | grep "comment --comment --comment"
```

   returns a non-empty result.

Description: This error is related to incorrect operation of NetFilter (`iptables`) earlier than 1.4.15. The error causes the rules having a unique tag (comment) to be added incorrectly, as a result of which,

when shutting down, SpIDer Gate cannot remove its rules for forwarding network connections.

Troubleshooting

1. Re-enable SpIDer Gate.
2. If you need to keep SpIDer Gate disabled, remove incorrect NetFilter (`iptables`) rules by running the command:

```
# iptables-save | grep -v "comment --comment --comment" | iptables-restore
```



Running the `iptables-save` and `iptables-restore` commands requires superuser privileges. To get superuser privileges, use the `su` or `sudo` command.

This command removes all rules with an incorrectly added comment from the list of rules (for example, those that were added by other applications adjusting traffic routing).

Additional information

- To prevent this issue in the future, it is recommended to upgrade your operating system (or at least NetFilter to version 1.4.15 or later).
- You can also switch SpIDer Gate to a manual mode of forwarding connections by specifying the required rules manually using the `iptables` utility (not recommended).
- For details, refer to the following `man` pages: `drweb-firewall(1)`, `drweb-gated(1)`, `iptables(8)`.

If the error persists, contact our technical support.

10.7. Appendix G. List of Abbreviations

The following abbreviations were used in this manual without further interpretation:

Convention	Complete form
AD	Microsoft Active Directory
DN	(LDAP) Distinguished Name
FQDN	Fully Qualified Domain Name
GID	Group ID (system user group identifier)
GNU	GNU project (GNU is Not Unix)
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure (via SSL/TLS)
ICAP	Internet Content Adaptation Protocol
ID	Identifier
IMA/EVM	Integrity Measurement Architecture and Extended Verification Module
IP	Internet Protocol
LDAP	Lightweight Directory Access Protocol
LKM	Loadable Kernel Module
MBR	Master Boot Record
OID	(SNMP) Object ID
OS	Operating System
PID	Process ID (system process identifier)
PAM	Pluggable Authentication Modules
RPM	Red Hat Package Manager
RRA	Round-Robin Archive
RRD	Round-Robin Database
SNI	Server Name Indication

Convention	Complete form
SNMP	Simple Network Management Protocol
SP	Service Pack
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UID	User ID (system user identifier)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VBR	Volume Boot Record

© Doctor Web, 2026