# Table of Contents

Version history ............................................................................................................. 2

1. Overview .................................................................................................................. 3
   Packages ................................................................................................................... 3
   Business products ................................................................................................... 3

2. Installation ................................................................................................................. 4
   Debian/Ubuntu ....................................................................................................... 4
   RHEL/CentOS ......................................................................................................... 5
   Installation of virus definitions ............................................................................... 6
   Avast public key ...................................................................................................... 6

3. Licensing ................................................................................................................... 7

4. Operation .................................................................................................................. 8
   Systemd units ......................................................................................................... 8
      avast.service ...................................................................................................... 8
      avast.socket ....................................................................................................... 9
      avast-vpsupdate.service ............................................................................... 9
      avast-vpsupdate.timer .................................................................................. 9
      avast-fss.service ............................................................................................ 9
      avast-rest.service ........................................................................................... 9
      avast.target ....................................................................................................... 9

5. Virus definitions updates ........................................................................................ 10
   Local virus definitions mirrors ........................................................................... 10
   Security considerations ......................................................................................... 10

6. REST API ................................................................................................................ 11
   Specification ......................................................................................................... 11
   Bash client ............................................................................................................. 11
   NGINX integration (RHEL/CentOS) ...................................................................... 12

7. AMaViS integration ................................................................................................. 14

Appendix A: scan(1) manual page ........................................................................... 15
Appendix B: avast(1) manual page .......................................................................... 18
Appendix C: avast-protocol(5) manual page ............................................................... 21
Appendix D: avast-fss(1) manual page .................................................................... 25
Appendix E: avast-rest(1) manual page .................................................................. 28
Appendix F: avastlic(1) manual page ...................................................................... 34
Appendix G: REST API specification ....................................................................... 35
Appendix H: Avast public key .................................................................................. 39
Appendix I: Third-party libraries ............................................................................ 40
Version history

The following table lists significant changes made to this documentation and to the product itself. The full changelog is available in /usr/share/doc/avast/ChangeLog after installing the avast package.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2022-01-31</td>
<td>4.0</td>
<td>Added systemd support.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added REST API component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added support: Debian 11, Ubuntu 20.04 LTS, RHEL 7, RHEL 8.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No longer supported: Debian 9, Ubuntu 16.04 LTS, RHEL 6, OpenSUSE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed repo URL and GPG key. See Migration from 3.x to 4.x</td>
</tr>
<tr>
<td>2022-07-31</td>
<td>4.1</td>
<td>Added support: Ubuntu 22.04 LTS, RHEL 9.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added note about Amazon Linux 2 support.</td>
</tr>
</tbody>
</table>

Migration from 3.x to 4.x

The program update URL and the GPG key has been changed.

Please update the URL according to the Installation section. For example:

- **DEB:**
  - deb http://deb.avast.com/lin/repo debian-buster release
  - ⇒ deb https://repo.avcdn.net/linux-av/deb debian-buster release

- **RPM:**
  - baseurl=http://rpm.avast.com/lin/repo/dists/rhel/release
  - ⇒ baseurl=https://repo.avcdn.net/linux-av/rpm/el$releasever/release

See Debian/Ubuntu for the supported Debian and Ubuntu releases.
See RHEL/CentOS for the supported RHEL-based distributions.

The Avast public key has been also changed, please install the new key according to Installation section.
1. Overview

The Avast Antivirus for Linux products include the following components which are distributed as standard software packages—DEB for Debian and Ubuntu systems, and RPM for RedHat systems. Software repositories are also provided so that all of the standard system management tools can be used to keep the Avast programs up to date.

Packages

avast

The avast package provides the core scanner service (avast) and a command line scan utility (scan). Additionally, it contains a command line license utility (avastlic). The package allows for on-demand scanning and mail server integration using AMaViS as described in section AMaViS integration.

The avast package is required by the avast-fss packages.

avast-fss

The avast-fss package provides a fanotify(7) based "on write" file system shield designed for file server usage. The typical use case for avast-fss are SMB/NFS file servers.

avast-rest

The avast-rest package contains an HTTP server which provides REST API for avast scanner service.

Business products

The Avast components are available as the following business products:

Avast Security Suite for Linux

License for all packages.
2. Installation

The Avast Linux server product is installed in two steps:

- Add the Avast repository to the system repositories.
- Install the desired packages from the repository.

The instructions differ depending on target distribution.

Debian/Ubuntu

1. Add the Avast repository to the system repositories.

   Supported distributions ($DIST variable in the following command):
   
   - `debian-buster` - Debian 10 “buster”
   - `debian-bullseye` - Debian 11 “bullseye”
   - `ubuntu-bionic` - Ubuntu 18.04 LTS (Bionic Beaver)
   - `ubuntu-focal` - Ubuntu 20.04 LTS (Focal Fossa)
   - `ubuntu-jammy` - Ubuntu 22.04 LTS (Jammy Jellyfish)

   ```
   root# DIST=$(.
   /etc/os-release; echo "$ID-
   VERSION_CODENAME")
   root# echo "deb https://repo.avcdn.net/linux-av/deb $DIST release"
   > /etc/apt/sources.list.d/avast.list
   ```

   For example, Debian 11 “bullseye” will have this line in `/etc/apt/sources.list.d/avast.list`:

   ```
   deb https://repo.avcdn.net/linux-av/deb debian-bullseye release
   ```

   Debian 9 “stretch” is no longer supported. The old Avast packages (version 3.x) are still available at the old URL:

   ```
   deb http://deb.avast.com/lin/repo debian-stretch release
   ```

2. Install Avast public key and update package manager state:

   ```
   root# cp /path/to/avast-gpg-key.asc /etc/apt/trusted.gpg.d/
   root# apt update
   ```

3. Install the `avast` package and optionally the `avast-fss`, `avast-rest` packages.
RHEL/CentOS

1. Add the Avast repository to the system repositories.

Supported distributions:

- **el7** - RHEL 7, CentOS 7, or compatible
- **el8** - RHEL 8, AlmaLinux 8, Rocky Linux 8, or compatible
- **el9** - RHEL 9, AlmaLinux 9, Rocky Linux 9, or compatible

Note that `$releasever` is a variable known to YUM, so it doesn't need to be replaced manually in the `avast.repo` file, as long as the actual `$releasever` matches one of the above versions.

For Amazon Linux 2, please replace `el$releasever` by `el7` manually. The distribution is RHEL 7 compatible, but uses its own versioning scheme, so the variable would evaluate to `el2`.

```
root# echo '[[avast]
name=Avast
baseurl=https://repo.avcdn.net/linux-av/rpm/el$releasever/release
enabled=1
gpgcheck=1
]' > /etc/yum.repos.d/avast.repo
root#
```

The content of `/etc/yum.repos.d/avast.repo` will be:

```
[avast]
name=Avast
baseurl=https://repo.avcdn.net/linux-av/rpm/el$releasever/release
enabled=1
gpgcheck=1
```

RHEL 6 is no longer supported. The old Avast packages (version 3.x) are still available at the old URL:

```
baseurl=http://rpm.avast.com/lin/repo/dists/rhel/release
```

2. Install **Avast public key**:

```
root# rpm --import /path/to/avast-gpg-key.asc
```
3. Install the `avast` package and optionally the `avast-fss, avast-rest` packages.

```
root# yum install avast
root# yum install avast-fss
root# yum install avast-rest
```

### Installation of virus definitions

The current virus definitions database (VPS) is downloaded during the installation of the `avast` package, so the installation may take some time.

### Avast public key

All packages and the RPM / DEB repository metadata are signed with Avast key. The public part of that key needs to be imported into APT or RPM package manager according to above instructions, to allow verification of the signatures.

The public key file is named `avast-gpg-key.asc`. Full listing of that file can be found in appendix: [Avast public key](https://repo.avcdn.net/linux-av/doc/avast-gpg-key.asc). For convenience, the file can be downloaded directly from [https://repo.avcdn.net/linux-av/doc/avast-gpg-key.asc](https://repo.avcdn.net/linux-av/doc/avast-gpg-key.asc). Please make sure the content of the downloaded file matches with the copy in the appendix.
3. Licensing

Access to the program repositories is not restricted in any way. The latest packages are always available, but they require a license file to run the components. The product license is stored in a file named `license.avastlic`. When you have the license file, copy it into the `/etc/avast` directory:

```
root# cp /path/to/license.avastlic /etc/avast
```

In case you have received an activation code instead of a license file, use `avastlic(1)` tool to download the license. Please note that for some activation codes this can only be done a limited number of times. Also, some activation codes require customer information to be entered and as such the tool is by default interactive.

```
$ avastlic -f new_license_file_path -c avast_activation_code
$ sudo cp new_license_file_path /etc/avast/license.avastlic
```

In case the downloaded license is valid for multiple machines, it is recommended to download the license once and then distribute the license file to all other machines.

Note that it's fine to run `avastlic` tool on a different machine than where the licence will be used.
4. Operation

All Avast packages provide systemd unit files for starting/stopping the services. For example starting the core Avast service is done by running

```
root# systemctl start avast
```

and stopping the service is done by running

```
root# systemctl stop avast
```

Reloading the configuration and the VPS is done by

```
root# systemctl reload avast
```

Or by sending SIGHUP to the `avast` process.

Restarting all installed avast daemons (avast, avast-fss), and recreating sockets in one transaction is done by

```
root# systemctl restart avast.target
```

All Avast services use the system logger (syslog) to create log files and the location is dependent on the host system. The most common log file paths are `/var/log/messages` and `/var/log/syslog`.

**Systemd units**

**avast.service**

The main service of the scanning engine.

This service may not be running right after the installation due to following reasons:

- A license is missing. See Licensing.
- Socket activation. By default, the service is started on-demand by `avast.socket`.

Listens on three Unix domain sockets:

- `/run/avast/scan.sock` - end-user protocol, see `avast-protocol(5)` manual page
- `/run/avast/emsg.sock` - a proprietary scan protocol, used by REST API and `[cliscan]`
- `/run/avast/dinfo.sock` - a proprietary info protocol, used by REST API and `[cliscan]`
avast.socket

Provides socket activation for avast.service.

This unit will create the two sockets for avast.service and start it when a client program connects to one of the sockets. This allows to save some resources when the service is not used continuously and also shields the client program from avast.service's restarts or crashes.

Note that this feature is optional — by starting avast.service manually, the socket activation can be skipped.

avast-vpsupdate.service

Updates the virus database (aka VPS). This unit launches the update script. It's not supposed to be running persistently, only when updating.

avast-vpsupdate.timer

Provides a timer for avast-vpsupdate.service. Runs the update every three hours.

avast-fss.service

This is an additional File Server Shield service, installed by avast-fss package.

Needs avast.service to be running or avast.socket ready to start it.

avast-rest.service

This is an additional REST API service, installed by avast-rest package.

Needs avast.service to be running or avast.socket ready to start it.

avast.target

A meta unit linked in all avast services. Restarting this unit restarts all avast daemons, and recreates sockets. Enabling it enables everything Avast antivirus needs to operate.

See [systemd.target(5)] for more information on systemd targets.
5. Virus definitions updates

Regularly updating the virus definitions database (VPS) is necessary to keep your antivirus protection up to date. Avast antivirus provides a shell script which checks for, downloads and installs the latest VPS. The update script is installed and enabled by default and executed every three hours as a systemd timer.

Avast uses incremental updates, so the average update data size is less than 0.5MB.

Local virus definitions mirrors

It is possible to use a local, mirrored, VPS repository. This is useful when you are running several Avast installations on your local network.

To set up a local VPS mirror, all you need is a local HTTP server that can serve a copy of the official public repository. To get your local repository copy, use the following command [1]:

```
$ wget --mirror --no-host-directories --cut-dirs=2 \ 
"http://linux-av.u.avcdn.net/linux-av/avast/x86_64/vps9/"
```

To change the VPS repository URL that Avast uses for VPS updates edit the `/etc/avast/vps.conf` configuration file.

Security considerations

The update files are signed by Avast, and the application verifies the signature before applying an update.

[1] Replace x86_64 with i386 for 32b systems
6. REST API

The REST API is available after installing `avast-rest` package.

The configuration and API is documented in `avast-rest(1)` man page.

**Specification**

The REST API specification in OpenAPI 3.0.0 format is installed together with other documentation: `/usr/share/doc/avast-rest/api.yaml`.

This file can be used to generate API clients in various programming languages using an open-source tool `openapi-generator`.

For example, to generate a Python client, install the tool and run:

```
openapi-generator generate -i /usr/share/doc/avast-rest/api.yaml -g python -o /tmp/rest-api-python
```

This generates a library that can be used to request the scans. An example program is generated to `README.md`.

This example shows a program which uses the Python library generated by `openapi-generator` 5.3.0:

```python
import openapi_client
from openapi_client.api import default_api
from pprint import pprint

configuration = openapi_client.Configuration(host = "http://127.0.0.1:8080")

filename = "eicar.com.txt"
content = "X5O!P%@AP%[4PZX54(P^)7CC)7}EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*"

with openapi_client.ApiClient(configuration) as api_client:
    api_instance = default_api.DefaultApi(api_client)
    api_response = api_instance.v1_scan_post(filename, body=content, full=True, archives=True, pup=False, heuristics=80)
    pprint(api_response)
```

**Bash client**

Following example shows a simple client written in Bash which scans files given in its arguments:
The client is also installed to `/usr/share/avast/scan-rest.sh`.

**NGINX integration (RHEL/CentOS)**

This example shows how to add HTTPS layer to the REST API.

For this example, generate a self-signed certificate:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
   -keyout /etc/pki/tls/private/example.key \
   -out /etc/pki/tls/certs/example.crt
```

Install `nginx` package, create a config file for avast backend `/etc/nginx/conf.d/avast-rest.conf`:

```
upstream avast_backend {
    server 127.0.0.1:8080;
    keepalive 8;
}

server {
    server_name example.local;
    listen 443 ssl;
    ssl_certificate /etc/pki/tls/certs/example.crt;
    ssl_certificate_key /etc/pki/tls/private/example.key;
    client_max_body_size 100M;

    location /avast/ {
        proxy_pass http://avast_backend/;
        proxy_http_version 1.1;
    }
}
```

Reload NGINX with `systemctl reload nginx`.

Test the HTTPS layer with `CURL`:
Note that --insecure is only needed in this example because we use a self-signed certificate.
7. AMaViS integration

AMaViS is an interface between mailer (MTA) and content checkers, which is already prepared for integration with mail scanners. This section describes how to integrate avast into AMaViS.

Integration of Avast into AMaViS includes AMaViS configuration updates and enabling access to emails going through AMaViS for Avast to scan. This can be divided into three steps:

- Integrating Avast antivirus

Open the AMaViS configuration file (e.g. `/etc/amavis/conf.d/50-user`) and insert the following lines into the file:

```bash
@av_scanners = (  
    ### http://www.avast.com  
    ['Avast', '/usr/bin/scan', '{}', [0], [1], qr/\t(.+)/m]  
);
```

- Enabling Virus Scanning

Then open the AMaViS content filter configuration file (e.g. `/etc/amavis/conf.d/15-content_filter_mode`) and enable antivirus checking mode by uncommenting the 'bypass_virus_checks' lines.

- Updating Access Permissions

Finally, enable the Avast scan service to scan emails going through AMaViS:

```
root# usermod -G amavis -a avast
```
Appendix A: scan(1) manual page

Name

scan - Avast command line scan utility

Synopsis

scan [-e PATH] [-abEfipuxJ] [PATH]...
scan [-a] -U [URL]...
scan -V
scan -h | -v

Description

Scan is the basic command line scanner that comes with Avast Antivirus for Linux. It searches the given PATH(s) for infected files and reports such files to the standard output. If no PATH is given, the scan paths are read from the standard input, line by line.

The scan tool is a client that connects to the Avast scan service. It cannot work separately, without a running scan service.

Options

-h
  Print short usage info and exit.

-v
  Print program version and exit.

-V
  Print the virus definitions (VPS) version and exit. The VPS version is retrieved from the scan service.

-U
  Check URLs. Checks whether an URL is malicious. Note: the URL is checked against a blacklist, no network request to the given URL is done.

-e PATH
  Exclude PATH from the scan. Use this option multiple times when more than one exclude path is required.

-a
  Print all scanned files / URLs, not just infected.
-b
  Report decompression bombs as infections. When set, files suspected of being decompression bombs are reported as infected, not as errors.

-E
  Scan e-mail files. Enables email-specific detections.

-f
  Scan full files. When set, the entire file contents are scanned, not just the relevant file parts.

-i
  Print verbose infection info. When set, verbose info about all infections found in the scanned file is printed.

-J
  Output information in JSON format.

-l LEVEL
  Set heuristics level to LEVEL (0-100).

-p
  Print archive content. When set, the files in an archive are listed separately, with the scan status for each shown.

-u
  Report potentially unwanted programs (PUP). When set, PUP files are reported as infected.

-x
  Do not extract archives. When set, compressed files are not extracted during scan.

**Output Format**

Every detected malicious file is reported on a separate line in the format:

```
PATH    INFECTION
```

where PATH and INFECTION are separated by a TAB character. If all files are printed using the -a option, then the clean files have a "[OK]" string as the infection name and files that could not be scanned (insufficient permissions, corrupted archives, ...) have an "[ERROR]" string as the infection name. Files, that were excluded from the scan using the -e option have a "[EXCLUDED]" string as the infection name.

If the -p option is set, PATH contains the archive path delimited by a "|>" delimiter in case of an archive.

If the -J option is set, the output format is JSON. Each report is printed as a single-line JSON fragment. Other options still affect which fields are present and what they contain.
Access Rights

It is the scan service that is accessing the files being scanned, not the scan utility itself, therefore the scan service must have access rights to the scanned files. Connections to the scan service may be restricted to clients with the same UID/GID in the scan service configuration, for details see avast(1).

Exit Status

The exit status is 0 if no infected files are found and 1 otherwise. If an error occurred, the exit status is 2. Infected status takes precedence over error status, thus a scan where some file could not be scanned and some infection was found returns 1.

See Also

avast(1)
Appendix B: avast(1) manual page

Name

avast - Avast antivirus scanner

Synopsis

avast [OPTIONS]

Description

avast is an antivirus scan service for OS X and Linux. Clients (shields, command line scan tool, ...) connect to the service's UNIX socket and perform scan requests and receive scan results.

Options

-h
  Print usage info and exit.

-V
  Print the program version and exit.

-l
  Check license and exit (exit status 12 = missing license).

-q
  Quiet mode: do not log to stderr.

-v
  Verbose mode: increase log level (-v = INFO, -vv = DEBUG).

-J
  Switch stderr log to JSON format.

-d DIR
  Verify that DIR is a valid data directory and contains a valid VPS. If the exit code is nonzero, than the VPS is missing or invalid. The check may generate some data files in the VPS directory if they are missing but can be generated from the corresponding "source" files.

-c FILE
  Set configuration file path to FILE. The default configuration file is /etc/avast/avast.conf.

Configuration
The configuration file format is INI file format, i.e. it consists of KEYWORD = VALUE entries, each on a separate line. Lines beginning with ';' are treated as comments and are ignored. Keys may be grouped into arbitrarily named sections. The section name appears on a line by itself, in square brackets ([ and ]).

The following example is an avast configuration file with explicitly defined default options:

```
; Avast configuration file

RUN_DIR = "/run/avast"
TEMP_DIR = "/tmp"
DATA_DIR = "/var/lib/avast"
SOCKET = "/run/avast/scan.sock"
LICENSE = "/etc/avast/license.avastlic"
WHITELIST = "/etc/avast/whitelist"
SUBMIT = "/usr/libexec/avast/submit"

[OPTIONS]
CREDENTIALS = 0
STATISTICS = 1
HEURISTICS = 1
STREAMING_UPDATES = 1
```

The configuration file is re-read on HUP signal by the program, but only the entries in the Options section are reloaded, changes to the global parameters are ignored.

**Global parameters**

**RUN_DIR**
- Run directory. The PID file is stored here.

**TEMP_DIR**
- Temporary directory. The program temporary files are stored here.

**DATA_DIR**
- Data directory. Contains the virus definitions database and various other data files used by avast.

**SOCKET**
- Path to the UNIX socket used by the clients to connect to the scan service. The socket is created by avast at service start.

**LICENSE**
- Path to the license file.

**WHITELIST**
- Path to the whitelist file. The whitelist file contains sha256 hashes of files, that shall not be reported as infections even though detected by the scan engine. The file format is one sha256 hash in text mode per line. Hash mark (#) prefixed comments can be used in the file.
SUBMIT
Path to the submit utility. If enabled (see the Options section), the submit utility creates and sends reports about infected and suspicious files to the avast virus lab.

Options

CREDENTIALS
If enabled, avast performs a UNIX socket credentials check, whenever a new client is connecting. If the client’s effective UID does not match the effective UID of the avast process or the client’s effective GID does not match the avast effective GID or any avast supplementary group GID, the connection is refused.

STATISTICS
If enabled, avast sends statistics submits about malicious files detected during the scan to the Avast virus lab.

HEURISTICS
If enabled, avast sends heuristics submits about suspicious files detected during the scan to the Avast virus lab.

STREAMING_UPDATES
If enabled, the scan service establishes a permanent network connection to the avast cloud and retrieves virus definitions updates instantly as they are released. Streaming updates are an addition to the regular virus database updates, they do not replace them (you always get all the streamed updates in the next regular virus definitions database update).

See Also
scan(1), avast-protocol(5)
Appendix C: avast-protocol(5) manual page

Name
avast-protocol - Avast UNIX socket communication protocol

Synopsis
nc -U /run/avast/scan.sock
socat /run/avast/scan.sock -

Description
avast(1) uses a text based protocol for communication with the scan service daemon over the UNIX socket. This manual page briefly describes the protocol.

General Protocol Rules
The communication consists of command-response pairs and is line-based. The new line terminator is CRLF. The general command syntax is:
<command>[<space><parameter>]]...

Responses may be numerical only, or may contain additional output data. Numerical responses have the format:
<code><space><command><space><msg>

The output data format is:
<command><space><data>

Output data are always encapsulated between numerical responses 210 (DATA) and the final numerical response for the command. Delimiters such as <space>, <tab> or CR/LF are backslash escaped, when present in the data or command argument.

The communication from the service starts with a numeric welcome message, 220. The protocol commands are case-insensitive.

Response Codes
200 OK
210 DATA
220 Welcome message
451 Engine error
466 License error
501 Syntax error
520 URL blocked
Commands

**SCAN**

Scan a file/directory.

**Synopsis:** SCAN <path>

The format of the data message lines is:

```
<path><tab><status>[<tab><info>]
```

The <status> has a format of ""'[<X>]'<depth>.0"", where <X> can be one of: "+" - file is OK, "E" - error during scan and "L" - infection found. <depth> is the depth when scanning inside archives (0 for common non-archive files).

The <info> follows the "E" and "L" cases. The "L" case info has the format "0<space><infection>". The "E" case info has the format "Error<space><errno><space><errstr>".

**Example:**

```
> scan /etc

210 SCAN DATA
SCAN /etc/fstab [+]0.0
SCAN /etc/shadow [E]0.0  Error 13 Permission\ denied
SCAN /etc/eicar.com [L]0.0  0 EICAR\ Test-NOT\ virus!!
...
200 SCAN OK
```

**VPS**

Get the virus definitions (VPS) version.

**Synopsis:** VPS

**Example:**

```
> VPS

210 VPS DATA
VPS 15051301
200 VPS OK
```
**PACK**  
Get/set packer options.

**Synopsis:** PACK [+-<packer>…]

Use +<packer> to enable a specific packer and -<packer> to disable it. When invoked without an argument, the packer set is displayed, but not changed. The same mechanism applies to the FLAGS and SENSITIVITY commands.

**Example:**

```
> PACK -zip -iso
210 PACK DATA
PACK +mime -zip +arj +rar ... +7zip -iso +dmg
200 PACK OK
```

**FLAGS**  
Get/set scan flags.

**Synopsis:** FLAGS [+-<flag>…]

**Example:**

```
> FLAGS +fullfiles
210 FLAGS DATA
FLAGS +fullfiles +allfiles -scandevices
200 FLAGS OK
```

**SENSITIVITY**  
Get/set scan sensitivity.

**Synopsis:** SENSITIVITY [+-<sensitivity>…]

**Example:**

```
> SENSITIVITY +pup
210 SENSITIVITY DATA
SENSITIVITY +worm +trojan +adware +spyware ... +pup
200 SENSITIVITY OK
```
**EXCLUDE**  Exclude path from scans.

**Synopsis:** EXCLUDE <path>

Paths omitted by exclusion are reported with error 42019 - Skipped due to exclusion settings. <path> is matched as a string prefix thus e.g. "/usr/lib/" excludes nothing because the "/usr/lib/" scan path does not match and any "/usr/lib/anything" subpath also does not match. <path> may contain wild cards ("*").

**Example:**

```
> EXCLUDE /tmp
210 EXCLUDE DATA
EXCLUDE /tmp
200 EXCLUDE OK
```

**CHECKURL**  Check whether a given URL is malicious.

**Synopsis:** CHECKURL <url>

**Example:**

```
> CHECKURL http://www.google.com
200 CHECKURL OK

> CHECKURL http://www.avast.com/eng/test-url-blocker.html
520 CHECKURL URL blocked
```

**See Also**

avast(1), nc(1), socat(1)
Appendix D: avast-fss(1) manual page

Name

avast-fss - Avast file server shield

Synopsis

avast-fss [OPTIONS]

Description

avast-fss (FSS), a part of Avast antivirus for Linux suite, provides real-time scanning of files written to any of the monitored directories. FSS is based on the fanotify(7) access notification system available on Linux kernels 2.6.37+.

The directories that should be monitored by FSS need to be configured in the config file (see below). By default, FSS does not monitor any directories. Note that the monitoring is always limited to the same mount point. If you want to monitor a mounted subdirectory of a monitored directory, add it explicitly to the config.

FSS monitors only write events. Access to already infected files is not monitored. Any write operation in a monitored directory triggers a scan by Avast engine. If it founds an infection, it moves the infected file to the chest directory and reports the finding in the virus log file (see Files).

Options

-h
  Print short usage info and exit.

-v
  Print the program version and exit.

-c FILE
  Set configuration file path to FILE. The default configuration file is /etc/avast/fss.conf.

-n
  Do not daemonize.

Configuration

The default configuration file is /etc/avast/fss.conf. Its format is INI as described in the avast(1) manual page.

The configuration consists of two parts - the global configuration options and the monitoring configuration. The sample configuration below shows all available global configuration options and
their default values followed by some examples of monitoring (and monitoring exclude) entries.

```ini
; Avast fileserver shield configuration file
RUN_DIR = "/run/avast"
SOCKET = "/run/avast/scan.sock"
LOG_FILE = "/var/log/avast/fss.log"
CHEST = "/var/lib/avast/chest"
SCANNERS = 4
UNLIMITED_QUEUE = 0

[MONITORS]
SCAN = "/some/directory/to/monitor"
SCAN = "/another/directory/to/monitor"
EXCL = "/path/to/exclude/from/scan"
```

**Global parameters**

**RUN_DIR**

Run directory. The PID file is stored here.

**SOCKET**

Path to the avast service UNIX socket.

**LOG_FILE**

Path to the virus log file.

**CHEST**

Path to the chest directory. The chest directory is where the detected malicious files are moved. If the chest directory is located in a monitored directory, it is automatically added to the excluded paths on startup.

**SCANNERS**

Number of parallel running scans. Set this option to the number of CPU cores to get the best performance.

**UNLIMITED_QUEUE**

If set to 1, FSS disables the limit on the fanotify event queue size. For more info, see FAN_UNLIMITED_QUEUE in fanotify_init(2).

**Monitors**

**SCAN**

A path that shall be monitored by FSS.

**EXCL**

A path to be excluded from monitoring.
Files

/etc/avast/fss.conf
Default configuration file. See Configuration.

/var/log/avast/fss.log
Default virus log file. Note that this is distinct from the syslog, which is used by FSS for normal logging.

/var/lib/avast/chest
Default directory for quarantined infected files.

Caveats

Files created via bound directories (mount --bind) or namespaces (unshare) may circumvent the fanotify(7) notification system, even if the file ultimately appears in the monitored directory. To work around this issue, add also the other directory to the list of monitored directories.

See Also

avast(1), fanotify(7)
Appendix E: avast-rest(1) manual page

Name

avast-rest - Avast Antivirus REST API

Synopsis

avast-rest [-qvsJ] (-V | -h) [-LP] [-I ADDRESS] [-p PORT] [-c SCANNERS]

Description

avast-rest is an HTTP server which provides a REST API for avast(1) daemon.

The REST API is documented (below) and will provide backward compatibility. Internally, the scanning requests are translated and forwarded to emsg.sock, which is avast(1) daemon’s proprietary communication protocol. Unlike avast-protocol(5) and the REST API, emsg.sock protocol can’t be used directly by end-users and is subject to change.

The service is managed by systemd, as avast-rest.service.

REST API

Currently, there is a single command, which can be used with GET or POST methods.

Scan local file by path

HTTP GET /v1/scan?path=PATH_TO_FILE

Scanning local paths is disabled by default. To enable it, either set server_side_paths in config file (see Configuration), or add --server-side-paths option to avast-rest program.

Scan uploaded file

HTTP POST /v1/scan?filename=FILE_NAME

The original filename should be given in filename parameter.

The file content is posted in request body as raw binary data. The http method can be POST or PUT, both work in the same way.

Required headers:

Content-Type: application/octet-stream

Content-Length: FILE_SIZE
Additional request parameters

Supported by both GET and POST methods. All parameters can be set globally in the config file (see Configuration). The value in URL overrides the value from config file.

email=0|1|false|true
This option hints the scanner that the file represents an email message. Enables email-specific detections. Can be used in an email service integration.

full=0|1|false|true
Scan full files. By default, the scanner chooses which parts of each file should be scanned and skips the rest as optimization.

archives=0|1|false|true
Unpack archives during scan. Enabled by default.

pup=0|1|false|true

heuristics=0..100
Level of heuristics: 0=disabled, 40=low, 80=medium, 100=high (maximum) Default is 80.

detections=0|1|false|true
Verbose information about all virus detections found in each path. When enabled, virus info is reported in detections array, instead of a single virus name (which is equivalent to detections[0].virus). This option is useful mainly for investigating problems with the scanner (e.g. false positives). Disabled by default.

Response

Examples:

• clean file:

```
{"issues":[],"vps_version":"21081902"}
```

• virus:

```
{"issues":[]{"path":"eicar.com.txt","virus":"EICAR Test-NOT virus!!!"},"vps_version":"21091404"}
```

• zip bomb:

```
{"issues":[]{"path":[]10G.gz","10G"},"warning_id":42110,"warning_str":"The file is a decompression bomb"},"vps_version":"21091404"}
```
Schema:

ScanResponse

• **issues**: array of ScanRecord, required
  The array is empty when scan did not find any virus or warning.

• **vps_version**: string, required
  Version of VPS (virus database) that was used to scan the file.

ScanRecord

• **path**: array of string, required
  Infected paths. The first part is the path or filename as given in the request. Each other part is a path inside an archive. Multiple archive paths are possible in case of wrapped archives.

• **virus**: string
  A name of a virus found in the path. This string is a unique ID of the virus.

• **warning_id**: integer
  Unique warning ID. Warnings are generated for other (non-virus) issues.

• **warning_str**: string
  Textual description of the warning.

Each ScanRecord contains either virus or warning_id, warning_str.

REST API specification

The REST API specification in OpenAPI 3.0.0 format is available in a file which is installed to /usr/share/doc/avast-rest/api.yaml.

Options

HTTP Server Options

- **-l, --listen ADDRESS**
  Listen on local interface with IP ADDRESS.

- **-p, --port PORT**
  Listen on TCP PORT.

- **-t, --threads THREADS**
  Number of server threads. See CONFIGURATION.
-L, --dont-route
Prevent routing (SO_DONTROUTE).

Scanner Options

-c, --scanners NUM
Number of scanner connections in pool. This is maximum number of concurrent scans. Exceeding requests will be queued.

-P, --server-side-paths
Allow scanning of server-side paths. See CONFIGURATION.

General Options

-q, --quiet
Do not log to stderr.

-v, --verbose
Increase log level. Affects both stderr and log file (syslog). Default log level is INFO. Single -v increases to DEBUG.

-s, --silent
Decrease log level. Multiple -s args will decrease further: -s to INFO, -ss to WARNING etc.

-J, --json-log
Switch stderr log to JSON format.

-V, --version
Print program version and exit.

-h, --help
Print help on program options and exit.

Configuration

The config file is /etc/avast/rest.conf. It's INI format.

After changing the configuration, restart avast-rest.service:

```
systemctl restart avast-rest.service
```

Note that reload (without restarting) is not supported.

[server]

listen = ADDRESS
Listen on local interface with IP ADDRESS. Can be overridden by command-line option: --listen
port = PORT
Listen on TCP PORT. Can be overridden by command-line option: --port

threads = THREADS
Number of server threads. Should not exceed number of CPUs - the server is asynchronous (event based), it can handle multiple requests concurrently in a single thread. Can be overridden by command-line option: --threads

dont_route = true|false
Prevent routing (SO_DONTROUTE).

[engine]

conn_pool = SCANERS
Number of scanner connections in pool. This is maximum number of concurrent scans. Exceeding requests will be queued. Can be overridden by command-line option: --scanners

max_queued = REQUESTS
Maximum number of requests that can be waiting for an available scanner.

[scanner]

server_side_paths = true|false
Allow scanning of server-side paths. This is useful when the server and client runs on the same machine - the client then provides only path to a file, server reads and scans the file. Can be overridden by command-line option: --server-side-paths

full_files = true|false
Scan full files. By default, the scanner chooses which parts of each file should be scanned and skips the rest as optimization. Can be overridden by request parameter: full

archives = true|false
Unpack archives during scan. Enabled by default. Can be overridden by request parameter: archives

pup = true|false
Scan sensitivity: Report potentially unwanted programs. Disabled by default. Can be overridden by request parameter: pup

heuristics = 0-100
Level of heuristics: 0=disabled, 40=low, 80=medium, 100=high (maximum) Can be overridden by request parameter: heuristics

detections = true|false
Verbose information about all virus detections found in each path. Can be overridden by request parameter: detections
See Also

avast(1)
Appendix F: avastlic(1) manual page

Name

avastlic - obtains license for Avast Antivirus for Linux

Synopsis

avastlic -f new_license_file_path -c avast_activation_code [-n]
avastlic -h

Description

The avastlic(1) command can be used to convert Avast Antivirus activation code into a license file. Please note that for some codes this can only be done limited number of times. Also, some activation codes require a customer information to be entered and as such the tool is by default interactive when the activation code mandates this information.

After downloading a license file with avastlic(1), install it by copying it to /etc/avast/license.avastlic.

In case the downloaded license is valid for multiple machines, it is recommended to download the license once and then distribute the license file to all the other machines.

Options

-f
   File path to store the obtained license into.

-c
   Activation code valid for Avast Antivirus for Linux

-n
   Non-interactive mode. If the activation code requires customer information, the activation will fail.

-h, --help
   Print help.

See Also

avast(1), scan(1), avast-protocol(5)
Appendix G: REST API specification

openapi: 3.0.0
info:
  description: Avast Linux AV Scan API
  version: "1.1.0"
  title: Scan API
  contact:
    email: linux-av@avast.com
paths:
  /v1/scan:
    get:
      summary: Scan server-side file
      description: Scan a file by path. The file is opened by the server on it's local filesystem. This is mostly useful for clients running on the same machine as the server. Scanning server-side paths is disabled by default.
      parameters:
        - in: query
          name: path
          description: Server-side path to be scanned
          required: true
          schema:
            type: string
      responses:
        '200':
          description: scan results
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ScanResponse'
        '400':
          description: client error - invalid parameters
        '500':
          description: scan engine error
    post:
      summary: Scan uploaded file
      description: Scan an uploaded file. The file content is received in body (raw binary). The original filename must be given in query parameters.
      parameters:
        - in: query
          name: filename
          description: The original filename. The filename will be used in the response to identify main file. The extension may be used by scanning engine as a hint.
          required: true
          schema:
            type: string
      responses:
        '200':
          description: scan results
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ScanResponse'
        '400':
          description: client error - invalid parameters
description: client error - invalid parameters
'500':
  description: scan engine error
requestBody:
  content:
    application/octet-stream:
      schema:
        type: string
        format: binary
  description: File content
parameters:
- in: query
  name: email
  description: 'This option hints the scanner that the file represents an email message. Enables email-specific detections. Can be used in an email service integration.'
  schema:
    type: boolean
    default: false
- in: query
  name: full
  description: 'Scan full files. By default, the scanner chooses which parts of each file should be scanned and skips the rest as an optimization.'
  schema:
    type: boolean
    default: false
- in: query
  name: archives
  description: 'Unpack archives during scan.'
  schema:
    type: boolean
    default: true
- in: query
  name: pup
  description: 'Scanning sensitivity: Report potentially unwanted programs.'
  schema:
    type: boolean
    default: false
- in: query
  name: heuristics
  description: 'Level of heuristics: 0=disabled, 40=low, 80=medium, 100=high'
  schema:
    type: integer
    format: int32
    minimum: 0
    maximum: 100
    default: 80
- in: query
  name: detections
  description: 'Enable verbose information about all virus detections ('detections' in scan response)'
  schema:
    type: boolean
    default: false
- in: header
  name: X-Correlation-Id
  description: 'A tag that will be attached to log messages related to this request'
components:
schemas:
  ScanResponse:
    type: object
    required:
      - issues
      - vps_version
    properties:
      issues:
        type: array
        items:
          $ref: '#/components/schemas/ScanRecord'
      vps_version:
        description: Version of VPS (virus database) that was used to scan the file.
        type: string
        example: 21091404

ScanRecord:
  type: object
  required:
    - path
  properties:
    path:
      description: Infected paths. The first part is the path or filename as given in the request. Each other part is a path inside an archive. Multiple archive paths are possible in case of wrapped archives.
      type: array
      items:
        type: string
        example: "[/path/to/archive.zip", "test/eicar.txt"]
    virus:
      description: A name of a virus found in the path. This string is a unique ID of the virus.
      type: string
      example: "EICAR Test-NOT virus!!!"
    detections:
      description: Verbose information about all virus detections found in the path. When enabled in config or by query parameter, this replaces the single 'virus' field by an array, where detections[0].virus is the reported virus and further items may contain additional detections. This is useful mainly for investigating problems with the scanner (e.g. false positives).
      type: array
      items:
        $ref: '#/components/schemas/VirusDetection'
    warning_id:
      description: Unique warning ID. Warnings are generated for other (non-virus) issues.
      type: integer
      format: uint32
      example: 42110
    warning_str:
      description: Textual description of the warning.
      type: string
      example: The file is a decompression bomb
VirusDetection:
  type: object
  required:
  - virus
  properties:
    virus:
      description: A name of a virus found in the path. This string is a unique ID of the virus.
      type: string
      example: "EICAR Test-NOT virus!!"
    algo:
      description: Detection algorithm that found the virus.
      type: string
      example: "troj"
    aux:
      description: Auxiliary information about the detection.
      type: string
      example: "PE3-C669AF050002E7759F7320603981C3F0"
Appendix H: Avast public key

-----BEGIN PGP PUBLIC KEY BLOCK-----
mQINBGHpNpwBEACqSWhJ3aE3taRzti+iElMFP0pxKlzPsUSwSN5Pbl/ie2eCJh
n2Q7QnCTX7Vxtwp4tp0PfMDDy/toPDIJx3+3kdU6ErT6v9fkrk8P71DGt+Bi7d56A6v
cnECoWYxtXLFokk5RqijNfISnD/qiYbmDTPY1BQZ5S0COWh80LAjbgzqHZNT
YTXT5WkYqUp0C3oD797hB52suAhW0OIXwb+U7BVXWZu4Q1D/wI/KykYtrNEbWOM3
P6taps/WSpPt7uTaC8BL5GQMHx3mkMld7ok0GqTEf19jGD2EXBH1/VFZECHwDzB
H6U00uxuYe6ecRl9+T416Azas044jZjpekR/YTkXrFG5v1NmpDrxNSH88eR01DR5/
+TVE6/6OSII02ufnvzsp4qDN1ACUmigum1tQbetaFUGT9gBF6eGxwyr5xcj
mpcsJPSicTpwNkDlW6gC7xorUBVXv6wKE20v9uNObi8Zp0n+Jyxdw6f8j/7ur3OGc6
fVRe891v2jg6f2QkA9yb0yBkxW2Z89UlAYG67rGratOb6WvHFcq/gozE6C5F15
/+o661Iz3dR0ip1jUYZvrzRHFzqPZqP1gw5OEveE13bJHem4xvdzmnRJEqgJ3aP5W
Jnd2qC4jXU3Vh/x3PlxX0qPqR3EvL599IW/x3VMUnzH1Z1sDqKpHfEQJQAQAB
bTda5W1eCl1AhcBmVYX73c0uY29pokCTqTAq0ABYhB8H1FEB8goAeVQ3
tQlnkwI+97P418QH96atsABCQCAhBUlCQQLAqAoWQeMBAh4BAheAAAOJEBKF
sI+i97P4162oQA3C36EGInLh+Vv+hAV4HD6vNvXe89x9YQ1Mule4r4KyQHtKgRZ/Ayz
gBjNZUzlO6G3FH4/88ajPmkeE7Y1meRkrukFDuC5qpsb5orDoA5P10/WkzISawo8wv
T8D0hFvii+Pd0eqrvjzq965ukx4H4d0YPMW33a5u+nsD71CRK7rzUw00KH
3ABC5ruZo6egzU3iBSnB8W00871Re6Rnr/3pRZ8QzncRACS5wjr2y2KGM+zzy
+yZRS+3J2kD5g7kW2h3xnhtXsY2KUq4qt/Ffepc8kOe2BD8EwZB8HYwV5P1
95s1sXVdDhMh8cO36UirnpK9myQaRQIP4G8dolp+dh/QcCllamto6e+M61PpFQYe
FoTglUIuWni9m3a57YyVumBB615I0EAGWtqI2QbIcLAP64wWgajowqinfIs0V
4e4DlN7t8Y/K8TSMezi8ULWvNhCr1b9C8bHR4qLNNbevB4BKC8p모+y1kz
b8x8Fd6pnFMPFQ9q14u1e1eUrtdncRZ5hbn6PwQSTs7GulJaMoYhlU+UH
4ENfHeY3+TH2GeEQmpM14ng4Ps8kmjHoAY1388G70s8KHMVVtv+vp2F68gyX1h
HuqksU65NfjcxQQFYwR48s6y+Vl+F/1/M9GwZvMIXAeWmZpY3YqV15pmuQIN6GH8
NpwBEAC1RoL5FD8xe9jUKefwUL1noMnPtac90iTY1hwmx2B0ghx3GJ3/pb+h
/Pj3jH3m36bI1nxCTzS14e75s59F90podovovc+MmUr6h4fTYGNgxaojv1U65+Y9
DwQ0ouUXM9FCfq3K3YUumK0Ufom+OwaSccF5PS6IFv7EmhHy6GnKPPY86Q3w6v/xk5
A1k9aP7jz+cruUezG2JN5T7+J3KUpxy0SMK-acJcbxLT90fzut18l+8Eneszc/zgh
bfw2yHlz2rmsLxMqErgG22FRowauEHpk3eapvZnmyNe1mnyExy8sdlZvfkQJlYJ
Qnuo/Gw1Rfxh3s8g80P8Q5skDvuUyBDNrrvNymEAASSM1AL4U4MQioY1itfymBtK
C51U0f3jbybzyU8h4pctCm0MrFtTb5N7pyBsr1T CePb7w4Ay
MhZvHodvqdyum0ydmzvAov90t/PCvCwV3k6Q3tMC5d6sEgNYFHFHqXqGf
nraMeI039LkkHuM5Q0rT905x1DCiOuZfVhhaR5P85fjyqsz0x0DjDATXcpBb6c
q6k6D7f5QAgWzVp3ZKkXZ43YLD8xYve0lP717jE/3EcElt/jouwU+OeGegWnWzHq
BNCb80TQPXG5s8xG4op4BmAncAXTWKXYAFaL14XyPa0a14QARAOAbiQ12BB8B
CAAgfIEEOOUGUEQag8A5V5De1Eo+wp1s/jUFAmHnpQnwpQcAgkQePwp+j73s/3jVy
CA/2QtdC3j24yaphQwSaeSw6i+iKTeFoH06iqu3BTSUCIOLsRsWu735Ro/nY3m
e6GzQ3C1ip3NC94ipKiqd9s6IDACs5rMDHpfVzuhGN73g966RY1+Zy08/oyWap
FwMnn1uoTPlijtSQk0XjsxPXMzHR8MBK2DuDA2MNAC5v4FhWbY82pWkkJ3g6LSpce19
DscIsrQRKh729iWkd83foYosjihWae0smjnI+rW9XDnPIAHE6Jc2fxoS0reuNz
BSHH0wO5BO4T5Tm3vY1chZ3zhp0UHPBOR/ReCbu9WSADj/LP/6uB5e5imidqy9
nuJrNA/HtDNTqqG5ScbB3HTHE5W+NnLuAbOBORq+1LMwppdRyYrwATXr5Z79FtL3Qoj
8r0k16RSN6C7qM1Q3Dnn9OtszYn5/1sMWppwNN1uVz9OdPvA201/3RbteYy1Y7D7+
kReeDuuE3ay/aML7CqifdXW4fIKLQqQw1yv2y43j7u33VtKuyZufPPPqScj0/U
5uO8KQRLKE06uH2BRTF0nUX0f5pJJe8w6qKo55icEgqg0L4c6YBHZfCuFW8BB
CsKnAz7PoHEqgK6Lly92aBzIIT2C1QGNL8jwprQRMJL4BWS/SuxZ0phH50zy3Mk
yp+Rq3aUCEqAq0KQXPQ0dav/Vh/Pdhxoc3Anu+RQhTeL0e=0kVw
-----END PGP PUBLIC KEY BLOCK-----
Appendix I: Third-party libraries

This software uses third-party libraries. Most of them are open-source software, some are commercial.

The libraries are either statically linked into the binaries, or provided by the OS and linked dynamically.

The list is split into three parts:

- External dependencies. These are provided by the OS, the RPM/DEB packages depend on them.
- Internal static dependencies. Linked into the programs distributed in RPM/DEB packages.
- VPS static dependencies. Linked into the VPS, which is distributed in VPS update channel.

External dependencies

The versions of these libraries depend on the specific Linux distribution.

- **avast**: systemd, curl, uuid
- **avast-fss**: systemd
- **avast-rest**: systemd, uuid

Internal static dependencies

These libraries are linked statically, thus embedded in the binaries.

- **boost** 1.78.0
- **bspatch** FreeBSD rev. 352742 ported to Linux
- **fmt** 8.1.1
- **libev** 4.33
- **pegtl** 3.2.7
- **protobuf** 3.19.1

VPS static dependencies

These libraries are linked statically by VPS engine (downloaded and updated together with virus definitions).

- 7-Zip
- aPLib
- BriefLZ
- bzip2
- Frozen
• libmspack
• LZFSE
• Nanopb
• PCRE
• SQLite
• UnRAR
• UPX
• xmlParser
• YARA
• zlib