



Revision A

Command Line Scanner for LINUX

(Product Guide – Version 7.0.0)

COPYRIGHT

Copyright © 2021 Musarubra US LLC.

TRADEMARK ATTRIBUTIONS

McAfee and the McAfee logo, McAfee Active Protection, ePolicy Orchestrator, McAfee ePO, McAfee EMM, Foundstone, McAfee LiveSafe, McAfee QuickClean, Safe Eyes, McAfee SECURE, SecureOS, McAfee Shredder, SiteAdvisor, McAfee Stinger, True Key, TrustedSource, VirusScan are trademarks or registered trademarks of McAfee, LLC or its subsidiaries in the US and other countries. Other marks and brands may be claimed as the property of others.

LICENSE INFORMATION

License Agreement

NOTICE TO ALL USERS: CAREFULLY READ THE APPROPRIATE LEGAL AGREEMENT CORRESPONDING TO THE LICENSE YOU PURCHASED, WHICH SETS FORTH THE GENERAL TERMS AND CONDITIONS FOR THE USE OF THE LICENSED SOFTWARE. IF YOU DO NOT KNOW WHICH TYPE OF LICENSE YOU HAVE ACQUIRED, PLEASE CONSULT THE SALES AND OTHER RELATED LICENSE GRANT OR PURCHASE ORDER DOCUMENTS THAT ACCOMPANY YOUR SOFTWARE PACKAGING OR THAT YOU HAVE RECEIVED SEPARATELY AS PART OF THE PURCHASE (AS A BOOKLET, A FILE ON THE PRODUCT CD, OR A FILE AVAILABLE ON THE WEBSITE FROM WHICH YOU DOWNLOADED THE SOFTWARE PACKAGE). IF YOU DO NOT AGREE TO ALL OF THE TERMS SET FORTH IN THE AGREEMENT, DO NOT INSTALL THE SOFTWARE. IF APPLICABLE, YOU MAY RETURN THE PRODUCT TO MCAFEE OR THE PLACE OF PURCHASE FOR A FULL REFUND.

Contents

Preface	5
About this guide	5
Audience	5
Conventions	5
What's in this guide	6
Find product documentation	6
1 Introducing Command Line Scanner for UNIX	7
Product features	7
Getting product information	7
Contact information	8
2 Installing Command Line Scanner for UNIX	9
About the distributions	9
Installation requirements	10
Other recommendations	10
Installing the software	10
Troubleshooting during installation	11
Testing your installation	12
Troubleshooting when scanning	12
Removing the program	13
3 Using Command Line Scanner for UNIX	15
Running an on-demand scan	15
Command-line conventions	16
General hints and tips	16
Configuring scans	17
Example 1	17
Example 2	17
Scheduling scans	18
Examples	18
Handling viruses	18
Example 1	19
Example 2	19
Example 3	19
Example 4	19
Using heuristic analysis	19
Handling an infected file that cannot be cleaned	20
Producing reports	20
Example	20
XML reports	21
Choosing the options	21
Scanning options	22
Response options	25
General options	26

Options in alphabetic order	26
Exit codes	29
4 Preventing Infections	31
Detecting new and unidentified viruses	31
Why do I need new DAT files?	31
Updating your DAT files	31
A Schema for the XML reports	37
Index	41

Preface

Contents

- ▶ *About this guide*
- ▶ *Find product documentation*

About this guide

This information describes the guide's target audience, the typographical conventions and icons used in this guide, and how the guide is organized.

Audience





McAfee documentation is carefully researched and written for the target audience.

The information in this guide is intended primarily for:

- **Administrators** — People who implement and enforce the company's security program.
- **Users** — People who use the computer where the software is running and can access some or all of its features.

Conventions

This guide uses these typographical conventions and icons.

<i>Italic</i>	Title of a book, chapter, or topic; a new term; emphasis
Bold	Text that is emphasized
Monospace	Commands and other text that the user types; a code sample; a displayed message
Narrow Bold	Words from the product interface like options, menus, buttons, and dialog boxes
Hypertext blue	A link to a topic or to an external website
	Note: Extra information to emphasize a point, remind the reader of something, or provide an alternative method
	Tip: Best practice information
	Caution: Important advice to protect your computer system, software installation, network, business, or data
	Warning: Critical advice to prevent bodily harm when using a hardware product

What's in this guide

This guide is organized to help you find the information you need.

This release of Command Line Scanner for UNIX includes the following new features or enhancements:

- McAfee Anti-Malware Scan Engine version 6300.

Find product documentation

On the **ServicePortal**, you can find information about a released product, including product documentation, technical articles, and more.

Task

- 1 Go to the **ServicePortal** at <https://support.mcafee.com> and click the **Knowledge Center** tab.
- 2 In the **Knowledge Base** pane under **Content Source**, click **Product Documentation**.
- 3 Select a product and version, then click **Search** to display a list of documents.

1

Introducing Command Line Scanner for UNIX

Command Line Scanner for UNIX detects and removes viruses on UNIX-based systems. This section describes:

- Product features
- Getting product information
- Contact information

Contents

- *Product features*
- *Getting product information*
- *Contact information*

Product features

The scanner runs from a command-line prompt, and provides an alternative to scanners that use a graphical user interface (GUI). Both types of scanner use the same anti-virus software. The scanner acts as an interface to the powerful scanning engine — the engine common to all our security products.

Although a few years ago, the UNIX operating system was considered a secure environment against potentially unwanted software, it is now seeing more occurrences of software specifically written to attack or exploit security holes in UNIX-based systems. Increasingly, UNIX-based systems interact with Windows-based computers, and although viruses written to attack Windows-based systems do not directly attack UNIX systems, the UNIX system can unknowingly harbor these viruses, ready to infect any client that connects to it.

When installed on your UNIX systems, Command Line Scanner for UNIX becomes an effective solution against viruses, Trojan-horse programs, and other types of potentially unwanted software.

The command-line scanner enables you to search for viruses in any directory or file in your computer on demand — in other words, at any time. The command-line scanner also features options that can alert you when the scanner detects a virus or that enable the scanner to take a variety of automatic actions.

When kept up-to-date with the latest virus-definition (dat files), the scanner is an important part of your network security. We recommend that you set up a security policy for your network, incorporating as many protective measures as possible.

Getting product information

Unless otherwise noted, product documentation comes as Adobe Acrobat .PDF files, or from the McAfee download site.

- **Product Guide** — Introduction to the product and its features; detailed instructions for configuring the software; information on deployment, recurring tasks, and operating procedures.
- **Help** — Product information in the Help system that is accessed from within the application on its main pages.
- **Release Notes** — ReadMe. Product information, resolved issues, any known issues, and last-minute additions or changes to the product or its documentation.
- **License Agreement** — The McAfee License Agreement booklet that includes all of the license types you can purchase for your product. The License Agreement presents general terms and conditions for use of the licensed product.
- **Contacts** — Contact information for McAfee services and resources: technical support, customer service, Security Headquarters (McAfee Labs), beta program, and training.

See also

Contact information on page 8

Contact information

Threat Center: McAfee Labs	<p>McAfee Labs Threat Library http://www.mcafee.com/us/mcafee-labs/threat-intelligence.aspx</p> <p>McAfee Labs DAT Notification Service https://sns.secure.mcafee.com/signup_login</p>
Download Site	<p>http://www.mcafee.com/us/downloads/ Product Upgrades (Valid grant number required) Security Updates (DATs, engine) HotFix and Patch Releases</p> <ul style="list-style-type: none"> • For Security Vulnerabilities (Available to the public) • For Products (ServicePortal account and valid grant number required) <p>Product Evaluation McAfee Beta Program</p>
Technical Support	<p>http://www.mcafee.com/us/support.aspx KnowledgeBase Search https://kc.mcafee.com/ McAfee Technical Support ServicePortal (Logon credentials required) https://mysupport.mcafee.com/Eservice/Default.aspx https://platinum.mcafee.com</p>
Customer Service	<p>Web https://secure.mcafee.com/apps/support/customer-service/request-form.aspx?region=us</p> <p>Phone — US, Canada, and Latin America toll-free: +1-888-VIRUS NO or +1-888-847-8766 Monday – Friday, 8 a.m. – 8 p.m., Central Time</p>
Professional Services	<p>Enterprise: http://www.mcafee.com/us/services.aspx Small and Medium Business:http://www.mcafee.com/us/small/index.aspx http://www.mcafee.com/us/medium/index.aspx</p>

2

Installing Command Line Scanner for UNIX

We distribute the Command Line Scanner for UNIX software as an archived file that you can download from our website or from other electronic services.

Review the Installation requirements in this chapter to verify that the software will run on your system, then follow the installation steps.

Contents

- ▶ *About the distributions*
- ▶ *Installation requirements*
- ▶ *Installing the software*
- ▶ *Testing your installation*
- ▶ *Removing the program*

About the distributions

Command Line Scanner for UNIX software comes in several distribution versions, one for each supported operating system.

- Linux for Intel 32-bit distributions shipping with version 2.6, 3.x or 4.x production kernels with libstdc++.so.5.0.5 or later installed.
- Linux for Intel 64-bit distributions shipping with version 2.6, 3.x or 4.x production kernels, with libstdc++.so.6 installed.

For current information about the distribution versions, refer to the Release Notes.

Installation requirements

To install and run the software, you need the following:

- The correct version of the UNIX distribution that you require, installed and running correctly on the target computer.
- At least 512 MB of free hard disk space
- At least an additional 512 MB of free hard disk space reserved for temporary files
- At least 512 MB of RAM for scanning operations (1024 MB recommended)
- At least 1024 MB of RAM for updating operations

See also

[About the distributions on page 9](#)

Other recommendations

- To install the software and perform on-demand scan operations of your file system, we recommend that you have root account permissions.
- To take full advantage of the regular updates to DAT files from our website, you need an Internet connection, either through your local area network, or via a high-speed modem and an Internet Service Provider.

Installing the software

This section shows how to install the software on any distribution. To install a specific distribution, substitute the correct file name for the distribution file.

To start the installation script:

Task

- 1 Download the appropriate Command Line Scanner for UNIX software distribution from our website.
- 2 Copy the distribution file to a directory on your system.



We recommend that you use a separate (possibly a temporary) directory — not the directory where you intend to install the software.

- 3 Change the directory to that containing the distribution file. Use `cd`.

- 4 Type this line at the command prompt to decompress the file:

```
gunzip distribution-file | tar -xf -
```

Here, *distribution-file* is the file you copied in Step 2

- 5 Type this line at the command prompt to execute the installation script:

```
./install-uvscan installation-directory
```

If you do not specify an installation directory, the software is installed in `/usr/local/uvscan`.

If the installation directory does not exist, the installation script asks whether you want to create it. If you do not create the installation directory, the installation cannot continue.

- 6 The installation script asks whether you want to create symbolic links to the executable, the shared library and the man page. Type **Y** to create each link, or **N** to skip the step.

We recommend that you create these links. Otherwise, you will need to set one of the following environment variables to include the installation directory:

Distribution	Variable
Linux	LD_LIBRARY_PATH
Linux 64-bit	LD_LIBRARY_PATH



The program also looks in the `/usr/lib` or `/lib` directory or the current directory for the shared library.

The installation program copies the program files to your hard disk, then scans your home directory.

See also

[Handling viruses on page 18](#)

[Troubleshooting during installation on page 11](#)

Troubleshooting during installation

The following table lists the most common error messages returned if the installation fails. The table also suggests a likely reason for the error and recommends any solutions.

Table 2-1 Error messages

Error	Cause or action
Failed to create <code>install_dir</code>	Verify that you have permission to create the installation directory.
Cannot write to <code>install_dir</code>	Verify that you have permission to write to the installation directory.
The <code>install_dir</code> exists, but is not a subdirectory	Choose another installation directory.
<code><file></code> is missing	The file might not exist.
<code><file></code> is not correct	The file did not install correctly.
Bad string	The error is generated by the <code>tr</code> command used by the <code>install-uvscan</code> and <code>uninstall-uvscan</code> scripts. For more information, see KB52240 .

Testing your installation

After it is installed, the program is ready to scan your computer for infected files. You can run a test to determine that the program is installed correctly and can properly scan for viruses. The test was developed by the European Institute of Computer Anti-virus Research (EICAR), a coalition of anti-virus vendors, as a method of testing any anti-virus software installation.

To test your installation:

Task

- 1 Open a standard text editor, then type the following line:

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



The line must appear as one line in the window of your text editor.

- 2 Save the file with the name EICAR.COM. The file size will be 68 or 70 bytes.
- 3 Type the following command to scan the EICAR.COM file:

```
uvscan -v EICAR.COM
```

When the program examines this file, it reports finding the EICAR test file, but you will not be able to clean or rename it.



This file is *not a virus* — it cannot spread or infect other files, or otherwise harm your computer. Delete the file when you have finished testing your installation to avoid alarming other users. Please note that products that operate through a graphical user interface do not return this same EICAR identification message.

- 4 When you have finished testing your installation, delete the test file to avoid alarming other users.

If the software appears not to be working correctly, check that you have Read permissions on the test file.

Troubleshooting when scanning

The following table lists the most common error messages returned if the uvscan program fails when scanning. The table also suggests a likely reason for the error and recommends possible solutions.

Table 2-2 Program messages

Program message	Remedy
Unable to find shared library	Set the appropriate environment variable: <ul style="list-style-type: none"> • For Linux, use <code>LD_LIBRARY_PATH</code>.
Cannot execute: permission denied	Check the file permissions. Incorrect file permissions can prevent the program running correctly. All executables (including the shared libraries) must have read and execute permissions (<code>r_x</code>), but we recommend <code>rwxr_xr_x</code> All DAT files must have read permissions.

Table 2-2 Program messages (continued)

Program message	Remedy
Missing or invalid DAT files	Re-install the DAT files.
The program has been altered; please replace with a good copy	Re-install from the original media; the program might be infected.

Removing the program

A script is installed at the same time as the Command Line Scanner for UNIX software, which enables you to remove the product quickly and easily.

To remove the product from your system:

Task

- 1 Run the script `uninstall-uvscan`, which is in the Command Line Scanner for UNIX program directory. For example, type the following command at the command prompt:

```
/usr/local/uvscan/uninstall-uvscan
```

- 2 Delete the script `uninstall-uvscan` from the program directory to remove the program completely from your system.

If you created your own links to the program and a shared library path when you installed the software, you must remove those links yourself.



Removing the software leaves your computer unprotected against threats. Remove the product only when you are sure that you can upgrade quickly to a new version.

If you are an administrator, ensure that your users cannot accidentally remove their Command Line Scanner for UNIX software.

3

Using Command Line Scanner for UNIX

Command Line Scanner for UNIX provides virus scanning from a command line. This section describes how to use its features and customize the program to meet your needs.

The following features offer optimum protection for your computer and network:

- On-demand scanning options let you start a scan immediately or schedule automatic scans.
- Advanced heuristic analysis detects previously unknown macro viruses and program viruses.
- Updates to virus definition files and to program components ensure that the program has the most current scanning technology to deal with viruses as they emerge.

Later sections in this guide describe each of these features in detail.

Contents

- ▶ *Running an on-demand scan*
- ▶ *Configuring scans*
- ▶ *Scheduling scans*
- ▶ *Handling viruses*
- ▶ *Using heuristic analysis*
- ▶ *Handling an infected file that cannot be cleaned*
- ▶ *Producing reports*
- ▶ *Choosing the options*
- ▶ *Exit codes*

Running an on-demand scan

You can scan any file or directory on your file system from the command line by adding options to the basic command.

Only the Intel-based Linux distributions of the Command Line Scanner for UNIX program can scan for boot-sector viruses.

When executed without options, the program displays a brief summary of its options. When executed with only a directory name specified, the program scans every file in that directory only, and issues a message if any infected files are found. The options fall into the following main groups:

- **Scanning options** — determine how and where the scanner looks for infected files.
- **Response options** — determine how the scanner responds to any infected files.
- **General options** — determine how the scanner reports its activities.

Each group of options appears in its own table with a description of its function.

See also

[Scanning options on page 22](#)

[Response options on page 25](#)

[General options on page 26](#)

[Choosing the options on page 21](#)

Command-line conventions

Use the following conventions to add options to the command line:

- Follow the syntax correctly. The UNIX operating system is case-sensitive.
- Type each option in lower case and separate each with spaces.
- Do not use any option more than once on the command line.
- Type single consecutive options as one option. For example, instead of typing this:

```
-c -r --one-file-system
```

you can type this:

```
-cr --one-file-system
```

- To start the program, at the command prompt, type:

```
uvscan
```

(This example assumes that the scanner is available in your search path.)

- To have the program examine a specific file or list of files, add the target directories or files to the command line after `uvscan`. You can also create a text file that lists your target files, then add the name of the text file to the command line.

By default, the program examines all files, no matter what their extensions. You can limit your scan by adding only those extensions you want to examine to the command line after the `--extensions` option, or you may exclude certain files from scans with the `--exclude` option.

See also

[Configuring scans on page 17](#)

[Choosing the options on page 21](#)

General hints and tips

The following examples assume that the scanner is available in your search path.

- To display a list of all options, with a short description of their features, type the command:

```
uvscan --help
```

- To display a list of all the viruses that the program detects, type the command:

```
uvscan --virus-list
```

- To display information about the version of the program, type the command:

```
uvscan --version
```

- To scan all subdirectories within a directory with maximum security, type the command:

```
uvscan -r --secure target
```

- To ensure maximum protection from virus attack, you must regularly update your DAT files.

See also

[Preventing Infections on page 4](#)

Configuring scans

Instead of running each scan with all its options directly from the command line, you can keep the options in a separate text file, known as a *task file*. In the file, you can specify the actions that the scanner must take when a virus is detected. This allows you to run complete scans with ease, and at any time; you need only specify the files or directories that you want to scan.

To configure a scan:

Task

- 1 Choose the command options that you want to use.
- 2 Type the command options into a text editor just as you might on the command line.
- 3 Save the text as a file — the task file.
- 4 Type one of these lines at the command prompt:

```
uvscan --load file target
```

```
uvscan --config file target
```

Here, *file* is the name of the task file you created, and *target* is the file or directory you want to scan.

If the scanner detects no virus infections, it displays no output.

The following examples show how you can configure scans using task files. The examples assume the scanner is available in the search path.

See also

[Choosing the options on page 21](#)

[Command-line conventions on page 16](#)

Example 1

To scan files in the `/usr/docs` directory according to the settings you stored in the task file, `/usr/local/config1`, type the command:

```
uvscan --load /usr/local/config1 /usr/docs
```

The contents of the task file `/usr/local/config1`, are:

```
-m /viruses --ignore-compressed --maxfilesize 4
```

They instruct the scan to move any infected files to `/viruses`, to ignore any compressed files in the target directory, and to examine only files smaller than 4mb.

As an alternative, you can arrange the contents of the task file as separate lines:

```
-m /usr/local/viruses
```

```
--ignore-compressed
```

```
--maxfilesize 4
```

Example 2

To scan only files smaller than 4mb and to ignore any compressed files in three separate directories, type the command:

```
uvscan --load /usr/local/config1 --file mylist
```

The contents of the task file `/usr/local/config1`, are:

```
--ignore-compressed
```

```
--maxfilesize 4
```

The contents of the other file, `mylist`, are:

```
/usr/local/bin
```

```
/tmp
```

```
/etc
```

Scheduling scans

You can use the UNIX `cron` scheduler to run automated scans. `Cron` stores the scheduling commands in its `crontab` files. For further information about `cron` and `crontab`, refer to your UNIX documentation or view the Help text, using the commands, `man cron` or `man crontab`.

Examples

To schedule a scan to run at 18:30 (6:30 p.m.) every weekday, add the following to your `crontab` file:

```
30 18 * * 1-5 /usr/local/bin/uvscan
```

To schedule a scan to run and produce a summary at 11:50 p.m. every Sunday, add the following to your `crontab` file:

```
50 23 * * 0 /usr/local/bin/uvscan --summary
```

To schedule a scan to run on the work directory at 10:15 a.m. every Saturday in accordance with options specified in a configuration file `conf1`, add the following to your `crontab` file:

```
15 10 * * 6 /usr/local/bin/uvscan --load conf1 /work
```

To schedule a scan to run at 8:45 a.m. every Monday on the files specified in the file `mylist`, add the following to your `crontab` file:

```
45 8 * * 1 /usr/local/bin/uvscan --f /usr/local/mylist
```

Handling viruses

If the scanner discovers a virus while scanning, it returns exit code number 13.

To clean infected files or directories, or move them to a quarantine location on your network, you can configure your scanner using one or more response options, which are described in Response options.

The following examples show how you can use these options to respond to a virus attack. The examples assume that the scanner is available in your search path.

See also

[Exit codes on page 29](#)

[Response options on page 25](#)

Example 1

To scan and clean all files in the `/usr/docs` directory and all of its subdirectories, type the command:

```
uvscan -cr /usr/docs
```

Example 2

To scan and clean all files in the `/usr/docs` directory and its subdirectories, but ignore any other file systems that are mounted, type the command:

```
uvscan -cr --one-file-system /usr/docs
```

Example 3

To scan all files except compressed files in the `/usr/docs` directory and its subdirectories, and to move any infected files to `/viruses`, type the command:

```
uvscan -m /viruses -r --ignore-compressed /usr/docs
```

Example 4

To scan a file with a name prefixed with "-", type the command:

```
uvscan -c -v - -myfile
```

The program scans the named file. It cleans any detected viruses and issues a progress message. This format avoids confusion between the names of the options and the name of the target. Without the "-" option, the `uvscan` command appears to have three options and no target:

```
uvscan -c -v -myfile
```

Using heuristic analysis

A scanner uses two techniques to detect viruses: signature matching and heuristic analysis.

A *virus* signature is simply a binary pattern that is found in a virus-infected file. Using information in the DAT files, the scanner searches for those patterns. However, this approach cannot detect a new virus because its signature is not yet known, therefore the scanner uses another technique — *heuristic analysis*.

Programs, documents or e-mail messages that carry a virus often have distinctive features. They might attempt unprompted modification of files, invoke mail clients, or use other means to replicate themselves. The scanner analyzes the program code to detect these kinds of computer instructions. The scanner also searches for legitimate non-virus-like behavior, such as prompting the user before taking action, and thereby avoids raising false alarms.

In an attempt to avoid being detected, some viruses are encrypted. Each computer instruction is simply a binary number, but the computer does not use all the possible numbers. By searching for unexpected numbers inside a program file, the scanner can detect an encrypted virus. By using these two techniques, the scanner can detect both known viruses and many new viruses and variants. Options that use heuristic analysis include `---analyze`, `--manalyze`, `--panalyze`.

See also

[Scanning options on page 22](#)

Handling an infected file that cannot be cleaned

If the scanner cannot clean an infected file, it renames the file to prevent its use. When a file is renamed, only the file extension (typically three letters) is changed. The following table shows the method of renaming.

Table 3-1 Renaming infected files

Original	Renamed	Description
Not v??	v??	File extensions that do not start with v are renamed with v as the initial letter of the file extension. For example, myfile.doc becomes myfile.voc.
v??	.vir	File extensions that start with v are renamed as .vir. For example, myfile.vbs becomes myfile.vir.
vir, v01-v99		These files are recognized as already infected, and are not renamed again.
<blank>	.vir	Files with no extensions are given the extension, .vir.

For example, if an infected file called `bad.com` is found, the scanner attempts to rename the file to `bad.vom`. However, if a file of that name already exists in the directory, the scanner attempts to rename the file to `bad.vir`, `bad.v01`, `bad.v02`, and so on.

For file extensions with more than three letters, the name is usually not truncated. For example, `notepad.class` becomes `notepad.vlass`. However, an infected file called `water.vapor` becomes `water.vir`.

Producing reports

The program might take some time to complete a scan, particularly over many directories and files. However, the scanner can keep you informed of its progress, any viruses it finds, and its response to them.

The program displays this information on your screen if you add the `--summary` or `--verbose` option to the command line.

The `--verbose` option tells you which files the program is examining.

When the scan finishes, the `--summary` option identifies the following:

- How many files were scanned.
- How many files were cleaned.
- How many files were not scanned.
- How many infected files were found.

See also

[Response options on page 25](#)

Example

In the report below, both the `--summary` and `--verbose` options were used for scanning files in the `/usr/data` directory.

```
$ uvscan --summary --verbose /usr/data
Scanning /usr/data/*
Scanning file /usr/data/command.com
Scanning file /usr/data/grep.com
Summary report on /usr/data/*
```

```
File(s)
Total files: ..... 2
Clean: ..... 2
Not scanned: ..... 0
Possibly Infected: ..... 0
```

To determine the time taken for the scan, you may use the UNIX `time` command.

XML reports

You can generate an XML format report using the `XMLPATH` switch. For example, run the following command from the install directory:

```
uvscan . --XMLPATH=report.xml --RPTALL
```

This will generate a file called `report.xml` with the following content.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Scan Results -->
<uvscan>
  <Preamble>
    <Product_name value="McAfee Command Line Scanner Command Line for Linux32" />
    <Version value="6.0.4.564" />
    <License_info value="(408) 988-3832 LICENSED COPY - May 09 2013" />
    <AV_Engine_version value="5600.1067" />
    <Dat_set_version value="7057" />
  </Preamble>
  <Date_Time value="2013-May-09 13:38:16" />
  <Options value=". --XMLPATH=report.xml --RPTALL " />
  <File name="/usr/local/uvscan/avvclean.dat" status="ok" />
  <File name="/usr/local/uvscan/avvnames.dat" status="ok" />
  <File name="/usr/local/uvscan/avvscan.dat" status="ok" />
  <File name="/usr/local/uvscan/config.dat" status="ok" />
  <File name="/usr/local/uvscan/liblnxfv.so.4" status="ok" />
  <File name="/usr/local/uvscan/report.xml" status="ok" />
  <File name="/usr/local/uvscan/runtime.dat" status="ok" />
  <File name="/usr/local/uvscan/uvscan" status="ok" />
  <File name="/usr/local/uvscan/vcl604upg.pdf" status="ok" />
  <summary On-Path="/usr/local/uvscan/" Total-files="14" Clean="10" Not-Scanned="4"
  Possibly-Infected="0" />
  <Time value="00:00.01" />
</uvscan>
```

See also

[Schema for the XML reports on page 4](#)

Choosing the options

The following sections describe the options you can use to target your scan:

- Scanning options.
- Response options.
- General options.
- Options in alphabetic order.

The descriptions use the following conventions to identify the options or required parameters:

- Short versions of each command option appear after a single dash (-).
- Long versions of each command option, if any, appear after two dashes (--).
- Variables, such as file names or paths, appear in italics within brackets (< >).

See also

Scanning options on page 22

Response options on page 25

General options on page 26

Options in alphabetic order on page 26

Command-line conventions on page 16

Scanning options

Scanning options describe how and where each scan looks for infected files. You can use a combination of these options to customize the scan to suit your needs.

Table 3-2 Scanning options

Option	Description
--afc <size>	Specify the file cache size. By default, the cache size is 12MB. A larger cache size can improve scanning performance in some cases, unless the computer has low memory. The range of sizes allowed is 8mb to 512mb. Specify the size in megabytes. For example, <code>--afc 64</code> specifies 64MB of cache.
--alole	Check every file for OLE objects.
--analyze --analyse	Use heuristic analysis to find possible new viruses in clean files. This step occurs after the program has checked the file for other viruses. For macro viruses only, use <code>--manalyze</code> . For program viruses only, use <code>--panalyze</code> .
--ascii	Displays filenames as ASCII text.
--atime-preserve --p --plad	Preserve the last-accessed time and date for files that are scanned. Some backup software archives only changed files, and determines this information from each file's last-accessed date (or 'a-time'). Normally, scanning changes that date. This option will preserve the date, enabling the backup software to work as intended. Sometimes when this option is used, the file date is not preserved; if a file contains a virus, or the scan was started by a user who does not own the file, the file date is changed.
--config <file>	Use the options specified in <file>. You cannot nest configuration files within other configuration files.
--d <directory> --dat <directory> --data-directory <directory>	Specify the location of the DAT files — <code>avvscan.dat</code> , <code>avvnames.dat</code> , and <code>avvclean.dat</code> . If you do not use this option in the command line, the program looks in the same directory from where it was executed. If it cannot find these data files, the program issues exit code 6.
--decompress	Decompress DAT files after an update.
--exclude <file>	Exclude the directories or files from the scan as specified in <file>. List the complete path to each directory or file on its own line. You may use wildcards, <code>*</code> and <code>?</code> .

Table 3-2 Scanning options (continued)

Option	Description
--e --exit-on-error	Quit and display an error message if an error occurs. The error message indicates the severity of the error.
--extensions <EXT1[,EXT2,...]>	Examine files that have the specified extensions. You can specify as many extensions as you want. Separate each with a comma, but without a space. If you choose this option, the program scans only susceptible files, files with execute permissions, and those you specify here. To see the list of susceptible files, use the <code>--extlist</code> option.
--extra <file>	Specify the full path and file name of any extra.dat file. If you do not specify this option in the command line, the program looks in the same directory from where it was executed. If it cannot find this file, the program issues exit code 6.
--fam	Find all macros, not just macros suspected of being infected. The scanner treats any macro as a possible virus and reports that the file contains macros. However, the macros are not removed. If you suspect that you have an infection in a file, you can remove all macros from the file using the <code>--fam</code> and <code>--cleandocall</code> or <code>--dam</code> options (listed under response options) together, although you should only do this with caution.
--f <file> --file <file>	Scan the directories or files as specified in <i><file></i> .
--floppya --floppyb	Scan the boot sector of the disk in drive A or B. This option is only for Intel-based Linux systems.
--hidemd5	This option allows the display of MD5 checksum of infected files to be hidden, if required.
--ignore-compressed --nocomp	Ignore compressed executables created with file-compression programs. Although this option reduces the scanning time, it increases the threat because these file types are not scanned.
--ignore-links	Do not resolve any symbolic links and do not scan the link targets. Normally, the program follows each symbolic link and scans the linked file.
--load <file>	See <code>--config</code> option.
--mailbox	Scan plain-text mailboxes. These include Eudora, PINE, and Netscape. Most mailboxes will be in MIME format, and therefore the <code>--mime</code> option is also required. This option does not clean or rename infected mail items; you must first extract them from the mailbox.
--analyze --manalyse --macro-heuristics	Use heuristic analysis to identify potential macro viruses. (In Microsoft Word, you can automate a task by using a macro - a group of Word commands that run as a single command.) This is a subset of <code>--analyze</code> .
--maxfilesize <size>	Examine only those files smaller than the specified size. Specify the file size in megabytes. For example, <code>maxfilesize 5</code> means scan only files that are smaller than 5MB.
--memsize	Set maximum size of file that will be cached in memory for scanning, in Kb.

Table 3-2 Scanning options (continued)

Option	Description
--mime	Scan MIME-encoded files. This type of file is not scanned by default.
--noboot	Do not scan the boot sector.
--nodecrypt	Do not decrypt Microsoft Office compound documents that are password-protected. By default, macros inside password-protected compound documents are scanned by employing password cracking techniques. If, for reasons of security, you do not require these techniques, use this option. Password cracking does not render the file readable.
--nocomp	See ignore-compressed.
--nodoc	Do not scan Microsoft Office document files. This includes Microsoft Office documents, OLE2, CorelDraw, PowerPoint, WordPerfect, RTF, Visio, Adobe PDF 5, Autodesk Autocad 2000, and Corel PhotoPaint 9 files.
--noexpire	Do not issue a warning if the DAT files are out of date.
--nojokes	Do not scan files that contain HTML, JavaScript, Visual Basic, or Script Component Type Libraries. This type of file is normally scanned by default. Stand-alone Javascript and Visual Basic Script files will still be scanned.
--one-file-system	Scan an entire directory tree without scanning mounted file systems, if you use this option in conjunction with the --sub option. Normally, the program treats a mount point as a subdirectory and scans that file system. This option prevents the scan from running in subdirectories that are on a different file system to the original directory.
--panalyze --panalyse	Use heuristic analysis to identify potential program viruses. By default, the program scans only for known viruses. The --panalyze option is a subset of --analyze.
--program	Scan for potentially unwanted applications. Some widely available applications, such as “password crackers” can be used maliciously or can pose a security threat.
--r --recursive --sub	Examine any subdirectories in addition to the specified target directory. By default, the scanner examines only the files within the specified directory.
--secure	Examine all files, unzip archive files and use heuristic analysis. This option activates the --analyze and --unzip options. If the --selected and --extensions options are in the command line, they are ignored.
--showcomp	Report any files that are packaged.
--showencrypted	Display encrypted documents. This switch retains the 5800 reporting behavior while scanning encrypted MS Office and PDF documents (without this parameter, the 5900 engine by default reverts to 5700 reporting behavior). The reporting of encrypted files is performed by using this parameter as these files are not reported by default.

Table 3-2 Scanning options *(continued)*

Option	Description
--s --selected	Look for viruses in any file that has execute permissions, and in all files that are susceptible to virus infection. By default, all files are scanned. By scanning only files that are susceptible to virus infection, the program can scan a directory faster. To see the list of susceptible files, use the <code>--extlist</code> option.
--sub	See <code>-r</code> .
--threads <nThreads>	Scan multithreaded with specified number of threads.
--timeout <seconds>	Set the maximum time to scan any one file.
--unzip	Scan inside archive files, such as those saved in ZIP, LHA, PKarc, ARJ, TAR, CHM, and RAR formats. If used with <code>--clean</code> , this option attempts to clean non-compressed files inside .ZIP files only. No other archive formats can be cleaned. The program cannot clean infected files found within any other archive format; you must first extract them manually from the archive file.
--xmlpath <filename>	Create XML report

See also[Using heuristic analysis on page 19](#)[Configuring scans on page 17](#)[Scheduling scans on page 18](#)[Exit codes on page 29](#)[General options on page 26](#)[Response options on page 25](#)

Response options

Response options determine how your scanner responds to an infection. You can use a combination of these options to customize the scan. None of the options in Table 3-3 occur automatically. To activate each option, specify it in the command line.

Table 3-3 Response options

Option	Definition
--c --clean	Automatically remove any viruses from infected files. By default, the program states only that infections were found but does not try to clean the infected file. If the program cannot clean the file, it displays a warning message. If you use this option, repeat the scan to ensure that there are no more infections.
--cleandocall --dam	Delete all macros in a file if an infected macro is found. If you suspect that a file is infected, you can choose to remove all macros from the file to prevent any exposure to a virus. To pre-emptively delete all macros in a file, use this option with <code>--fam</code> (listed under scanning options), although you should do this with caution. If you use these two options together, all found macros are deleted, regardless of the presence of an infection.
--delete	Automatically delete any infected files that are found.

Table 3-3 Response options *(continued)*

Option	Definition
<code>--m <directory></code>	Move any infected files to a quarantine location as specified.
<code>--move <directory></code>	When the program moves an infected file, it replicates the full directory path of the infected file inside the quarantine directory so you can determine the original location of the infected file. If you use this option with <code>--clean</code> , the program copies the infected files to a quarantine location and tries to clean the original. If the program cannot clean the original, it deletes the file.
<code>--norename</code>	Do not rename an infected file that cannot be cleaned.

See also[Scanning options on page 22](#)[Handling an infected file that cannot be cleaned on page 20](#)

General options

General options provide help or give extra information about the scan. You may use a combination of these options to customize the scan. None of the options in Table 3-4 occur automatically. To activate each option, specify it as part of the command line.

Table 3-4 General options

Option	Definition
<code>--extlist</code>	Display a list of all file extensions that are susceptible to infection. In other words, those file extensions that are scanned when <code>--selected</code> is set.
<code>--h</code> <code>--help</code>	List the most commonly used options, with a short description. For a full description, use <code>man uvscan</code> .
<code>--html <FILENAME></code>	Create a file containing the results in HTML format.
<code>--summary</code>	Produce a summary of the scan. This includes the following: <ul style="list-style-type: none"> • How many files were examined. • How many infected files were found. • How many viruses were removed from infected files.
<code>--v</code> <code>--verbose</code>	Display a progress summary during the scan.
<code>--version</code>	Display the scanner's version number.
<code>--virus-list</code>	Display the name of each virus that the scanner can detect. This option produces a long list, which is best viewed from a text file. To do this, redirect the output to a file for viewing.

See also[Producing reports on page 20](#)[Contact information on page 8](#)

Options in alphabetic order

For convenience, the options are repeated in this section in alphabetic order. For fuller descriptions, see the previous sections.

Table 3-5 Option definitions

Option	Definition
<code>--afc <size></code>	Specify the file cache size.
<code>--alole</code>	Check every file for OLE objects.
<code>--analyse</code>	Same as <code>--analyze</code> .
<code>--analyze</code>	Use heuristic analysis to find possible new viruses in clean files.
<code>--ascii</code>	Displays filenames as ASCII text.
<code>--atime-preserve</code>	Preserve the last-accessed time and date for files that are scanned.
<code>--c</code>	Same as <code>--clean</code> .
<code>--clean</code>	Automatically remove any viruses from infected files.
<code>--cleandocall</code>	Same as <code>--dam</code> .
<code>--config <file></code>	Run the options specified in <code><file></code> .
<code>--d <directory></code>	Same as <code>--dat <directory></code> .
<code>--dam</code>	Delete all macros in a file if an infected macro is found.
<code>--dat <directory></code>	Specify the location of the DAT files - avvscan.dat, avvnames.dat, and avvclean.dat.
<code>--data-directory <directory></code>	Same as <code>--dat <directory></code> .
<code>--decompress</code>	Decompress DAT files after an update.
<code>--delete</code>	Automatically delete any infected files that are found.
<code>--e</code>	Same as <code>--exit-on-error</code> .
<code>--exclude <file></code>	Exclude the directories or files from the scan as specified in <code><file></code> .
<code>--exit-on-error</code>	Quit and display an error message if an error occurs.
<code>--extensions <EXT1[,EXT2,...]></code>	Examine files that have the specified extensions.
<code>--extlist</code>	Display a list of all file extensions that are susceptible to infection.
<code>--extra <file></code>	Specify the full path and file name of any extra.dat file.
<code>--f <file></code>	Same as <code>--file <file></code> .
<code>--fam</code>	Find all macros, not just macros suspected of being infected.
<code>--file <file></code>	Scan the directories or files as specified in <code><file></code> .
<code>--floppya --floppyb</code>	Scan the boot sector of the disk in drive A or B.
<code>--h</code>	Same as <code>--help</code> .
<code>--help</code>	List the most commonly used options, with a short description.
<code>--html <FILENAME></code>	Create a file containing the results in HTML format.
<code>--ignore-compressed</code>	Ignore compressed files.
<code>--ignore-links</code>	Do not resolve any symbolic links and do not scan the link targets.
<code>--load <file></code>	Same as <code>--config <file></code> .
<code>--m <directory></code>	Same as <code>--move <directory></code> .
<code>--macro-heuristics</code>	Same as <code>--manalyze</code> .
<code>--mailbox</code>	Scan plain-text mailboxes.
<code>--manalyse</code>	Same as <code>--manalyze</code> .
<code>--manalyze</code>	Use heuristic analysis to identify potential macro viruses.
<code>--maxfilesize <size></code>	Examine only those files smaller than the specified size.

Table 3-5 Option definitions *(continued)*

Option	Definition
--memsize	Set maximum size of file that will be cached in memory for scanning, in Kb.
--mime	Scan MIME-encoded files.
--move <directory>	Move any infected files to a quarantine location as specified.
--noboot	Do not scan the boot sector.
--nocomp	Same as --ignore-compressed .
--nodecrypt	Do not decrypt Microsoft Office compound documents that are password-protected.
--nodoc	Do not scan Microsoft Office document files.
--noexpire	Do not issue a warning if the DAT files are out of date.
--nojokes	Do not report any joke programs.
--norename	Do not rename an infected file that cannot be cleaned.
--noscript	Do not scan files that contain HTML, JavaScript, Visual Basic, or Script Component Type Libraries.
--one-file-system	Scan an entire directory tree without scanning mounted file systems, if you use this option in conjunction with the --sub option.
--p	Same as --atime-preserve .
--panalyse	Same as --panalyze .
--panalyze	Use heuristic analysis to identify potential program viruses.
--plad	Same as --atime-preserve .
--program	Scan for potentially unwanted applications.
--r	Same as --sub .
--recursive	Same as --sub .
--s	Same as --selected .
--secure	Examine all files, unzip archive files and use heuristic analysis.
--selected	Look for viruses in any file that has execute permissions, and in all files that are susceptible to virus infection.
--showcomp	Report any files that are packaged.
--showencrypted	Display encrypted documents.
--sub	Examine any subdirectories in addition to the specified target directory.
--summary	Produce a summary of the scan.
--threads <nThreads>	Scan multithreaded with specified number of threads.
--timeout <seconds>	Set the maximum time to scan any one file.
--unzip	Scan inside archive files, such as those saved in ZIP, LHA, PKarc, ARJ, TAR, CHM, and RAR formats.
--v	Same as --verbose .
--verbose	Display a progress summary during the scan.
--version	Display the scanner's version number.
--virus-list	Display the name of each virus that the scanner can detect.
--xmlpath <filename>	Create XML report.

See also*Scanning options on page 22**Response options on page 25**General options on page 26*

Exit codes

When it exits, Command Line Scanner for UNIX returns a code to identify any viruses or problems that were found during a scan.

Table 3-6 Exit codes

Code	Description
0	The scanner found no viruses or other potentially unwanted software and returned no errors.
2	Integrity check on a DAT file failed.
6	A general problem occurred.
8	The scanner could not find a DAT file.
12	The scanner tried to clean a file, and that attempt failed for some reason, and the file is still infected.
13	The scanner found one or more viruses or hostile objects - such as a Trojan-horse program, joke program, or test file.
15	The scanner's self-check failed; it may be infected or damaged.
19	The scanner succeeded in cleaning all infected files.

4

Preventing Infections

Command Line Scanner for UNIX is an effective tool for preventing infections, and it is most effective when combined with regular backups, meaningful password protection, user training, and awareness of threats from viruses and other potentially unwanted software.

To create a secure system environment and minimize the chance of infection, we recommend that you do the following:

- Install Command Line Scanner for UNIX software and other McAfee anti-virus software where applicable.
- Include a `uvscan` command in a `crontab` file.
- Make frequent backups of important files. Even if you have Command Line Scanner for UNIX software to prevent infections, damage from fire, theft, or vandalism can render your data unrecoverable without a recent backup.

Detecting new and unidentified viruses

To offer the best virus protection possible, we continually update the definition (DAT) files that the Command Line Scanner for UNIX software uses to detect viruses and other potentially unwanted software. For maximum protection, you should regularly retrieve these files.

We offer free online DAT file updates for the life of your product, but cannot guarantee they will be compatible with previous versions. By updating your software to the latest version of the product and updating regularly to the latest DAT files, you ensure complete virus protection for the term of your software subscription or maintenance plan.

Why do I need new DAT files?

Hundreds of new viruses appear each month. Often, older DAT files cannot detect these new variations. For example, the DAT files with your original copy of Command Line Scanner for UNIX might not detect a virus that was discovered after you bought the product.

If you suspect you have found a new virus, use McAfee Labs WebImmune.

See also

[Contact information on page 8](#)

Updating your DAT files

DAT files are contained in a single compressed file that you can download from the internet.

Task

- 1 Navigate to the FTP location <ftp://ftp.mcafee.com/commonupdater/current/vscandata1000/dat/0000/>.
- 2 To gain access, type anonymous as your user name and your email address as your password when prompted.
- 3 Look for a filename that is of the format **avvdat-nnnn.zip**, where nnnn is the DAT version number.

Tasks

- [Using the new DAT files: on page 32](#)

Using the new DAT files:

Task

- 1 Create a download directory.
- 2 Change to the download directory, and download the new compressed file from the source you have chosen.
- 3 Type this command to move the DAT files to the directory where your software is installed. Name the file using lower case.

```
mv *.dat installation-directory
```

Here, *installation-directory* is the directory where you installed the software.

Your computer overwrites the old DAT files with the new files. Your anti-virus software will now use the new DAT files to scan for viruses.



After an update, run the following command once to decompress the newly downloaded DATs and accelerate the time for subsequent initializations: `uvscan --decompress`

This product is not suitable for on-access (single file) scanning.

Sample update script for UNIX

The following script retrieves the most recent DAT files from the McAfee website. It is provided as an example, for you to use and modify for your environment. It has not been thoroughly tested.



This script should be executed from the directory where `uvscan` has been installed. The `EMAIL_ADDRESS` variable needs to be set to a valid email address.

```
#!/bin/sh
# Copyright (c) 2021 Musarubra US LLC.
# required programs: unzip, ftp, awk, echo, cut, ls, printf

### defaults: do not modify
unset md5checker leave_files debug

#=====
### change these variables to match your information
# Set the following to your own e-mail address
EMAIL_ADDRESS=""

### change these variables to match your environment
# install_dir must be a directory and writable
install_dir=`dirname "$0`
# tmp_dir must be a directory and writable
tmp_dir="/tmp/dat-update"
# optional: this prg is responsible for calculating the md5 for a file
md5checker="md5sum"

### set your preferences
```



```
# set to non-empty to leave downloaded files after the update is done
#leave_files="true"
# show debug messages (set to non-empty to enable)
debug=yes

# these variables are normally best left unmodified
UVSCAN_EXE="uvscan"
UVSCAN_SWITCHES=""
#=====

Cleanup()
{
    if [ -z "$leave_files" ] ; then
        for f in "$avvdat_ini" "$download" ; do
            [ -n "$f" -a -e "$f" ] && rm -f "$f"
        done
    fi
}
exit_error()
{
    [ -n "$1" ] && printf "$prgname: ERROR: $1\n"
    Cleanup ; exit 1
}
print_debug()
{
    [ -n "$debug" ] && printf "$prgname: [debug] $@\n"
}

# Function to parse avvdat.ini and return, via stdout, the
# contents of a specified section. Requires the avvdat.ini
# file to be available on stdin.
# $1 - Section name
FindINISection()
{
    unset section_found

    section_name="$1"
    while read line ; do
        if [ "$line" = "$section_name" ] ; then
            section_found="true"
        elif [ -n "$section_found" ] ; then
            if [ "`echo $line | cut -c1`" != "[" ] ; then
                [ -n "$line" ] && printf "$line\n"
            else
                unset section_found
            fi
        fi
    done
}

# Function to return the DAT version currently installed for
# use with the command line scanner
# $1 - uvscan exe (including path)
# $2 - any extra switches for uvscan
GetCurrentDATVersion()
{
    dirname=`dirname "$1"`
    uvscan_bin=`basename "$1"`

    output=`(cd "$dirname"; ./"$uvscan_bin" $2 --version )`
    [ $? -eq 0 ] || return 1

    lversion=`printf "$output\n" | grep "Dat set version:" |
        cut -d' ' -f4`
    printf "${lversion}.0\n"

    return 0
}

# Function to download a specified file from ftp.mcafee.com
# $1 - Path on ftp server
# $2 - name of file to download.
# $3 - download type (either bin or ascii)
```

```

# $4 - download directory
DownloadFile()
{
    [ "$3" = "bin" -o "$3" = "ascii" ] || return 1
    dtype="$3"

    # An e-mail address must be set in this environment variable
    [ -n "$EMAIL_ADDRESS" ] || return 1

    print_debug "downloading file '$2' into '$4'"
    echo ""
    open ftp.mcafee.com
    user anonymous $EMAIL_ADDRESS
    cd $1
    lcd $4
    $dtype
    get $2
    " | ftp -i -n || return 1

    return 0
}

# Function to check the specified file against its expected size, checksum and MD5 checksum.
# $1 - File name (including path)
# $2 - expected size
# $3 - MD5 Checksum
ValidateFile()
{
    # Check the file size matches what we expect...
    size=`ls -l "$1" | awk ' { print $5 } '`
    [ -n "$size" -a "$size" = "$2" ] || return 1

    # make md5 check optional. return "success" if there's no support
    [ -z "$md5checker" -o "(" ! -x "`which $md5checker 2> /dev/null`" \
    ")" ] && return 0

    # Check the md5 checksum...
    md5_csum=`$md5checker "$1" 2>/dev/null | cut -d' ' -f1`
    [ -n "$md5_csum" -a "$md5_csum" = "$3" ] # return code
}

# Function to extract the listed files from the given zip file.
# $1 - directory to install to
# $2 - downloaded file.
# $3 - list of files to install
Update_ZIP()
{
    unset flist
    for file in $3 ; do
        fname=`printf "$file\n" | awk -F':' ' { print $1 } '`
        flist="$flist $fname"
    done

    # Backup any files about to be updated...
    [ ! -d "backup" ] && mkdir backup 2>/dev/null
    [ -d "backup" ] && cp $flist "backup" 2>/dev/null

    # Update the DAT files.
    print_debug "uncompressing '$2'..."
    unzip -o -d $1 $2 $flist >/dev/null || return 1
    for file in $3 ; do
        fname=`printf "$file\n" | awk -F':' ' { print $1 } '`
        permissions=`printf "$file\n" | awk -F':' ' { print $NF } '`
        chmod "$permissions" "$1/$fname"
    done

    return 0
}

# globals
prgname=`basename "$0"`
unset perform_update avvdat_ini download

# sanity checks

```

```
[ -d "$tmp_dir" ] || mkdir -p "$tmp_dir" 2>/dev/null
[ -d "$tmp_dir" ] || exit_error "directory '$tmp_dir' does not exist."
[ -x "$install_dir/$UVSCAN_EXE" ] \
|| exit_error "could not find uvscan executable"

DownloadFile "commonupdater" "avvdat.ini" "ascii" "$tmp_dir" \
|| exit_error "downloading avvdat.ini"
avvdat_ini="$tmp_dir/avvdat.ini"
# Did we get avvdat.ini?
[ -r "$avvdat_ini" ] || exit_error "unable to get avvdat.ini file"

ini_section=AVV-ZIP
file_list="avvscan.dat:444 avvnames.dat:444 avvclean.dat:444"

# Get the version of the installed DATs...
current_version=`GetCurrentDATVersion "$install_dir/$UVSCAN_EXE" "$UVSCAN_SWITCHES"`
[ -n "$current_version" ] \
|| exit_error "unable to get currently installed DAT version."
current_major=`echo "$current_version" | cut -d. -f1`
current_minor=`echo "$current_version" | cut -d. -f2`

INISection=`FindINISection "$ini_section" < $avvdat_ini`
[ -n "$INISection" ] \
|| exit_error "unable to get section $ini_section from avvdat.ini"

unset major_ver file_name file_path file_size md5
# Some INI sections have the MinorVersion field missing.
minor_ver=0 # To work around this, we will initialise it to 0.

# Parse the section and keep what we are interested in.
for field in $INISection ; do
    name=`echo "$field" | awk -F=' ' { print $1 } `
    value=`echo "$field" | awk -F=' ' { print $2 } `
    case $name in
        "DATVersion") major_ver=$value ;; # available: major
        "MinorVersion") minor_ver=$value ;; # available: minor
        "FileName") file_name="$value" ;; # file to download
        "FilePath") file_path=$value ;; # path on FTP server
        "FileSize") file_size=$value ;; # file size
        "MD5") md5=$value ;; # MD5 checksum
    esac
done

# sanity check
[ -n "$major_ver" -a -n "$minor_ver" -a -n "$file_name" \
-a -n "$file_path" -a -n "$file_size" -a -n "$md5" ] \
|| exit_error "avvdat.ini: '[$ini_section]' has incomplete data"

[ "(" "$current_major" -lt "$major_ver" ")" -o "(" \
"$current_major" -eq "$major_ver" -a \
"$current_minor" -lt "$minor_ver" ")" ] && perform_update="yes"

if [ -n "$perform_update" ]; then
printf "$prgname: Performing an update ($current_version -> $major_ver.$minor_ver)\n"

# Download the dat files...
DownloadFile "commonupdater$file_path" "$file_name" "bin" "$tmp_dir" \
|| exit_error "downloading '$file_name'"
download="$tmp_dir/$file_name"

# Did we get the dat update file?
[ -r "$download" ] || exit_error "unable to get $file_name file"

ValidateFile "$download" "$file_size" "$md5" \
|| exit_error "DAT update failed - File validation failed"
Update_ZIP "$install_dir" "$download" "$file_list" \
|| exit_error "updating DATs from file '$download'"

# Check the new version matches the downloaded one.
new_version=`GetCurrentDATVersion "$install_dir/$UVSCAN_EXE" \
"$UVSCAN_SWITCHES"`

new_major=`echo "$new_version" | cut -d. -f1`
new_minor=`echo "$new_version" | cut -d. -f2`
```

```
if [ "$new_major" = "$major_ver" -a "$new_minor" = "$minor_ver" ]
then printf "$prgname: DAT update succeeded $current_version -> $new_version\n"
else exit_error "DAT update failed - installed version different than expected\n"
fi
else
printf "$prgname: DAT already up to date ($current_version)\n"
fi

Cleanup ; exit 0
```

A

Schema for the XML reports

The formal schema for the XML reports is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema for the VSCL 6.0 XML Report format-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="Scan">
<xs:complexType>
<xs:sequence>
<xs:element ref="Preamble"/>
<xs:element ref="Date_Time"/>
<xs:element ref="Options"/>
<xs:group ref="FileSummary" maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref="Time"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Preamble">
<xs:complexType>
<xs:sequence>
<xs:element ref="Product_name"/>
<xs:element ref="Version"/>
<xs:element ref="License_info"/>
<xs:element ref="AV_Engine_version"/>
<xs:element ref="Dat_set_version"/>
<xs:element ref="Extra_Dat_Info" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Date_Time">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Options">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Time">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:group name="FileSummary">
<xs:sequence>
<xs:element ref="File" maxOccurs="unbounded" minOccurs="0"/>
<xs:element ref="Summary" maxOccurs="unbounded"/>
</xs:sequence>
</xs:group>
<xs:element name="File">
<xs:complexType>
<xs:attribute name="status" type="xs:string" use="required"/>
<xs:attribute name="name" type="xs:string" use="required"/>
<xs:attribute name="virus-name" type="xs:string" use="optional"/>
<xs:attribute name="detection-type" type="xs:string" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Summary">
```

```

<xs:complexType>
<xs:attribute name="Total-processes" type="xs:int" use="optional"/>
<xs:attribute name="On-Path" type="xs:string" use="optional"/>
<xs:attribute name="Total-files" type="xs:int" use="optional"/>
<xs:attribute name="Total-Objects" type="xs:int" use="optional"/>
<xs:attribute name="Possibly-Infected" type="xs:int" use="optional"/>
<xs:attribute name="Objects-Possibly-Infected" type="xs:int" use="optional"/>
<xs:attribute name="Not-Scanned" type="xs:int" use="optional"/>
<xs:attribute name="Clean" type="xs:int" use="optional"/>
<xs:attribute name="Possibly-Infected-MBR" type="xs:int" use="optional"/>
<xs:attribute name="Possibly-Infected-BootSector" type="xs:int" use="optional"/>
<xs:attribute name="Master-Boot-Records" type="xs:int" use="optional"/>
<xs:attribute name="Boot-Sectors" type="xs:int" use="optional"/>
<xs:attribute name="Cleaned" type="xs:int" use="optional"/>
<xs:attribute name="Moved" type="xs:int" use="optional"/>
<xs:attribute name="Deleted" type="xs:int" use="optional"/>
</xs:complexType>
</xs:element>
<xs:element name="Product_name">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Version">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="License_info">
<xs:complexType>
<xs:attribute name="value" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="AV_Engine_version">
<xs:complexType>
<xs:attribute name="value" type="xs:decimal" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Dat_set_version">
<xs:complexType>
<xs:attribute name="value" type="xs:short" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Extra_Dat_Info">
<xs:complexType>
<xs:attribute name="Path" type="xs:string" use="required"/>
<xs:attribute name="Additional_Viruses" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>

```

The following table lists the Status attributes and the description for each attribute.

Table A-1 Status attribute description

Status attributes	Description
ok	The object was scanned ok.
infected	Virus-name and detection-type will contain further detail.
corrupted	The object is corrupt. This message is usually issued for files within archives; for example, a corrupted .ZIP file.
error locked	The object could not be opened for reading or data could not be read from the object.
password-protected	The object is encrypted and the engine does not understand the encryption method. This value is usually issued for compressed files that are password-protected.
packaged	The object is packaged with a packer. The object is neither being reported as infected nor as being clean.

Table A-1 Status attribute description *(continued)*

Status attributes	Description
error bcs-file	The object is a Block, FIFO, or character special file. This is only used in UNIX systems.
error outofmemory	A memory allocation failed and the scan cannot continue.
error process not running	The specified process was not found. This for process objects only.
Scan Time Out	Scanning stopped due to /TIMEOUT option.
Unknown error	Unspecified internal error. This may be returned for errors that are not covered by other XML status strings.

Index

bloodhound (See heuristic analysis) 19
IDE (See DAT files) 31
pattern files (See DAT files) 31
switches (See options 17)
virus definitions (See DAT files) 31

A

about this guide 5
access date of files, preserving last 22
ascii switch 22
automatic scan 18

B

backup software 22
boot-sector viruses 15

C

cache sizes, for archives 22
cleaning infected files 25
COM2EXE 22
compressed files, ignore during scans 22
configuration file, option for loading saved 22
configuration options 17
conventions and icons used in this guide 5
conventions, command line 16
cron, UNIX command 18
crontab files, for automatic scans 18
Cryptcom 22

D

DAT file 31
 do not show expiration notice 22
 updates 31
disk scanning 22
distributions, versions of software 9
documentation
 audience for this guide 5
 product-specific, finding 6
 typographical conventions and icons 5

E

EICAR "virus" for testing installation 12

encrypted files 22
error codes 29
error messages 11
Eudora 22
examples
 configuring scans 17
 configuring scans 17
 consecutive options 16
 cron 18
 --verbose option [examples : verbose] 20
 --summary option [examples: summary] 20
 reports 20
 scanning and cleaning 19
 scanning and cleaning 19
 scheduling scans 18
 update script for UNIX 32
exit codes 29
exit-on-error, setting for scans 22
extra.dat 22

F

files, list of types scanned 26

G

general options 26
GZIP 22

H

help, online 16, 26
heuristic analysis 22, 25
HTML 22

I

infected files
 cannot be cleaned 20
 cleaning 25
 quarantine 25
 renaming 20
installation requirements 10
installation, testing effectiveness of 12
installing Command Line Scanner
software 10

J

JavaScript 22
joke programs 22

L

library paths 10
links, creating to uvscan and shared library 10
list of viruses 26

M

macros 22
 delete from files 25
mailboxes
 not cleaned 22
 plain-text 22
McAfee ServicePortal, accessing 6
Microsoft Expand 22
Microsoft Word files, do not scan 22
MIME 22

N

Netscape 22

O

on-demand scanning 15
options
 alphabetic list o 26
 examples 17, 19
 examples 17, 19
 general 26
 report 20
 response 18

P

password cracking 22
permissions 10
PINE 22
PKLITE 22
plain-text mailboxes 22
preventing virus infection 31
progress of scan
 summary of scan 20
progress summary 26

Q

quarantine, moving infected files to 25

R

recursion 22
removing the software
 by hand 13
 with the uninstallation script 13
reports 20

response options 18
return values 29
root account 10

S

scan targets, supplying by a file 22
scanning
 ARC files 22
 boot sector of disk 22
 diskette 22
 on-demand 15
 secure 22
 time taken for 20
 with maximum security 16
scheduling a scan 18
Script Component Type Libraries 22
secure scanning 22
ServicePortal, finding product documentation 6
shared library path, removing 13
standard input, to set scan targets 22
subdirectories, scanning of 22
summary of scan results, displaying; scan results, displaying 26
syntax, variables in 21
system requirements 10

T

task file 17
technical support, finding product information 6
Teledisk 22
testing your installation 12
troubleshooting installation 11

U

updates 15

V

variables, in command line 21
verbose scan reports, setting 26
version number 16, 26
viruses
 cleaning infected files 25
 list of detected 16
 obtaining a list of 26
 preventing infections 31
 signature 19
Visual Basic 22

W

warning, “-” option 19
what's in this guide 6

Z

zipped files, ignore during scans 22

