Clam AntiVirus 0.67
User Manual

by Tomasz Kojm

Contents

1 Introduction

Clam AntiVirus is an anti-virus toolkit for UNIX. The main purpose of this software is the integration with mail servers (attachment scanning). The package provides a flexible and scalable multi-threaded daemon, a command line scanner, and a tool for automatic updating via Internet. The programs are based on a shared library distributed with the Clam AntiVirus package, which you can use with your own software.

1.1 Features

- Licensed under the GNU General Public License, Version 2
- POSIX compliant, portable
- Very fast scanning
- On-access scanning (Linux only)
- Detects over 20000 viruses, worms and trojans
- Supports archives and compressed files
- Built-in support for RAR (2.0), Zip, Gzip, Bzip2
- Built-in protection against archive bombs
- Built-in support for Mbox, Maildir and raw mail files
- Includes a database updater with support for digital signatures

1.2 Mailing lists

There are four mailing lists available:

- **clamav-announce*lists.sf.net** - info about new versions
(including debian package releases), moderated[1].
- **clamav-users*lists.sf.net** - user questions
- **clamav-devel*lists.sf.net** - developement
- **clamav-virusdb*lists.sf.net** - database update announcements

You can subscribe and check the mailing list archives at: `http://www.clamav.net/ml.html`

[1] That means the subscribers are not allowed to write into the mailing list

1.3 Virus submitting

If you have got a virus that is not detected by your ClamAV with latest databases, please check it with the *ClamAV Online Specimen Scanner*:

`http://www.gietl.com/test-clamav`

and then submit on our website:

`http://www.clamav.net/cgi-bin/sendvirus.cgi`

Alternatively you can send it to this address:

`virus*clamav.net`

If your system doesn't allow to send infected files, please create a zip archive protected with the password: *virus*

2 Installation

2.1 Requirements

You will need the *zlib* and *zlib-devel* packages and the *gcc* compiler (both 2.9x and 3.x are supported). You can install the *bzip2* library (and its development files) to get bzip2 support, but it's not required. **It's highly recommended to install the GNU MP 3 library in order to enable support for a database digital signatures.**

SOLARIS TIP: *You should set the* ABI *system variable to 32 (e.g.* ***setenv ABI 32****) before running GMP's configure.* [2]

2.2 Supported platforms

Clam AntiVirus is prepared for the installation on the following operating systems / architectures (tested platforms in brackets):

- GNU/Linux 2.2/2.4 (All flavours, Intel/SPARC/Alpha/zSeries/S/390)
- Solaris 2.6/7/8/9 (Intel/SPARC)
- FreeBSD 4.5/6/7 5.0 (Intel/Alpha)
- OpenBSD 3.0/1/2 (Intel)
- AIX 4.1/4.2/4.3/5.1 (RISC 6000)

[2] Thanks to Ed Phillips

- HPUX 11.0
- SCO UNIX
- IRIX 6.5.20f
- Mac OS X
- BeOS
- Cobalt MIPS boxes (RAQ1, RAQ2, QUBE2)
- Windows/Cygwin

Some features may not be available on your operating system. If you have run Clam AntiVirus on the system not listed above, please let us know.

2.3 Binary packages

There are high quality *deb* and *rpm* packages available for Linux. The Debian package is maintained by Magnus Ekdahl and you will find it on debian mirrors, `http://www.debian.org`. The RPM package is maintained by Arkadiusz Miskiewicz and is distributed with Polish(ed) Linux Distribution (`ftp://ftp.pld.org.pl`). There is also available an official RPM package for Mandrake (maintained by Oden Eriksson) and binary package for AIX in AIX PDSLIB, UCLA `http://aixpdslib.seas.ucla.edu/packages/clamav.html`. BSD ports are available for Free, Net and OpenBSD. The official FreeBSD port is maintained by Masahiro Teramoto. The official port for NetBSD is maintained by UNKNOWN. The unofficial port for OpenBSD (maintained by Flinn Mueller) is available at:
`http://www.activeintra.net/openbsd/article.php?id=5`.

2.4 Installation

Please read the README file in the current version, because it may contain some important release notes. If you are installing ClamAV for the first time, you have to add a new *clamav* user and group to your system: [3]

```
# groupadd clamav
# useradd -g clamav -s /bin/false -c "Clam AntiVirus" clamav
```

[3] Cygwin note: If you don't have /etc/passwd, you don't need that.

The above method works on Linux and Solaris, in case you don’t have *groupadd, useradd* please consult a system manual. If you are installing ClamAV on user account you may omit this step with the option *–disable-clamav* passed to the *configure* script:

```
$ ./configure --disable-clamav
```

This disables test for the *clamav* user and group. **clamscan** still requires *clamav* to work in the superuser mode. The password for the *clamav* account should be locked in */etc/passwd* or */etc/shadow*.
Once you have created the clamav user and group, please extract the archive:

```
$ zcat clamav-x.yz.tar.gz | tar xvf -
$ cd clamav-x.yz
```

Assuming you want to have the configuration file installed in /etc, configure the package as follows:

```
$ ./configure --sysconfdir=/etc
```

Currently *gcc* is required for the compilation. Support for other compilers will be added in a near future.

```
$ make
$ su -c "make install"
```

In the last step software is installed in the /usr/local directory and the config file in /etc.
WARNING: Never set SUID or SGID bits on Clam AntiVirus programs.

2.5 Configuration

If you are going to use the daemon you have to configure it because it won’t run with default settings:

```
$ clamd
ERROR: Please edit the example config file
       /etc/clamav.conf.
```

This shows a location of the configuration file. The format and options of this file are fully described in the *clamav.conf(5)* manual. clamd configuration is very easy because the config file is well commented. Remember - you must remove the ”Example” directive.

Another feature of clamd is on-access scanning based on the Dazuko module, available from `http://dazuko.org`. **This is not required to run clamd - furthermore, you shouldn’t run Dazuko on production systems**. A special thread in clamd responsible for a communication with Dazuko is called ”Clamuko” (it’s due to the funny name of Dazuko - I don’t know what Clamuko means). Clamuko is supported on Linux 2.2 and 2.4 only. To compile dazuko execute:

```
$ tar zxpvf dazuko-a.b.c.tar.gz
$ cd dazuko-a.b.c
$ make dazuko
or
$ make dazuko-smp (for smp kernels)
$ su
# insmod dazuko.o
# cp dazuko.o /lib/modules/‘uname -r‘/misc
# depmod -a
```

Depending on your Linux distribution you have to add a ”dazuko” entry to
/etc/modules or something like:

```
modprobe dazuko
```

to some startup file in order to load dazuko at a boot time. You must also create the */dev/dazuko* device:

```
$ cat /proc/devices | grep dazuko
254 dazuko
$ su -c "mknod -m 600 /dev/dazuko c 254 0"
```

Now just configure Clamuko in *clamav.conf*. Please check 3.2 section.

2.6 Testing

OK. Let’s do some tests. Try to scan recursively the source directory:

```
$ clamscan -r -l scan.txt clamav-x.yz
```

It should find some (test) viruses in the clamav-x.yz/test directory. The scan result is saved in the scan.txt log file. [4]. To test clamd: start it and use *clamdscan* (you can also connect directly to clamd and run the SCAN command):

```
$ clamdscan -l scan.txt clamav-x.yz
```

The output and the logfile should be similar to the ones of *clamscan*.

2.7 freshclam: Setting up auto-updating

freshclam is a default database updater for Clam AntiVirus. It may work in two modes:

- interactively - from command line, verbosly
- daemon - alone, silently

When started by a superuser it (by default) drops privileges and switches to the *clamav* user. *freshclam* uses the `database.clamav.net` round-robin DNS which automatically selects a database mirror2.8. freshclam is an advanced tool: supports proxy servers (with authentication), digital signature verification and various error scenarious. **Quick test: run *freshclam* (as superuser) with no parameters and check the output.** If everything is OK you may create the log file in /var/log (owned by *clamav* or another user freshclam is running as (*–user*):

```
# touch /var/log/clam-update.log
# chmod 600 /var/log/clam-update.log
# chown clamav /var/log/clam-update.log
```

Now you *should* edit the configuration file (usually *freshclam.conf*) and configure the *UpdateLogFile* directive to point the created log file (it's highly recommended). Optionally you may force the log file path with *-l*. Finally, to run freshclam in a daemon mode execute:

```
# freshclam -d
```

The other method is to use the *cron* daemon. You have to add the following line to the crontab of the **root** or **clamav** users:

```
0 * * * * /usr/local/bin/freshclam --quiet
```

[4] More info on clamscan options: **man clamscan**

to check for a new database every hour. Proxy settings are only configurable via the configuration file (so you can setup proper permissions to protect your proxy password):

```
HTTPProxyServer myproxyserver.com
HTTPProxyPort 1234
HTTPProxyUsername myusername
HTTPProxyPassword mypass
```

2.8 Database mirrors

freshclam downloads the database from `http://database.clamav.net`. This is a round robin record that tries to equally balance the traffic between all the database mirrors:

Mirror	IP	Location	Administrator
clamav.man.olsztyn.pl	213.184.16.3	Olsztyn, Poland	Robert d'Aystetten dart*man.olsztyn.pl
avmirror1.prod.rxgsys.com	64.74.124.90	USA	graham*rxgsys.com
avmirror2.prod.rxgsys.com	207.201.202.73	USA	graham*rxgsys.com
clamav.e-admin.de	212.162.12.159	Dusseldorf, Germany	Andreas Gietl a.gietl*e-admin.de
clamav.essentkabel.com	195.85.130.84	Netherlands	Chris van Meerendonk mirror*essentkabel.com
clamav.inet6.fr	62.210.153.201 62.210.153.202	France	Lionel Bouton clamavdb*inet6.fr
clamav.netopia.pt	193.126.14.29	Portugal	Miguel Bettencourt Dias mbd*netopia.pt
clamav.sonic.net	209.204.175.217	USA	Kelsey Cummings kgc*sonic.net
clamav.nettron.co.za	160.124.112.17	South Africa	Ryan Zwankhuizen info*nettron.co.za
clamav.nchost.net	203.208.228.153	Singapore	Nicholas Chua nicholas*ncmbox.net
clamav.gossamer-threads.com	64.69.64.158	Canada	Alex Krohn mirrors*gossamer-threads.com
clamav.catt.com	64.18.100.4	USA	Mike Cathey mirrors*catt.com
clamav.datahost.com.ar	200.32.4.47	Argentina	Federico Omoto federico.omoto*datahost.com.ar
clamav.antispam.or.id	202.134.0.71	Indonesia	Fajar Nugraha fajar*telkom.co.id
clamav-du.viaverio.com	199.239.233.95	USA	Scott Wiersdorf scott*perlcode.org
clamav-sj.viaverio.com	128.121.60.235	USA	Scott Wiersdorf scott*perlcode.org
clamavdb.heanet.ie	193.1.219.100	Ireland	Colm MacCarthaigh mirrors*heanet.ie
clamav.crysys.hu	152.66.249.132	Hungary	Bencsath Boldizsar boldi*mail2004.crysys.hit.bme.hu
clamav.rockriver.net	209.94.36.5	Illinois, USA	Thomas D. Harker tom*rockriver.net
clamav.xmundo.net	200.68.106.40	Argentina	Cristian Daniel Merz mirrors*xmundo.net
clamav.infotex.com	66.139.73.146	Texas, USA	Matthew Jonkman matt*infotex.com
clamav.santafesolutions.com	196.40.71.226	Costa Rica	Gregory Cascante Avils gregory*emailcr.com

There is a *DatabaseMirror* directive in the config file, which specifies the main database server freshclam will attempt to connect to (up to *MaxAttempts* times) in order to download the database. If there are more *DatabaseMirror* lines specified it will switch automatically to the next one if the previous mirror failed for some reason.

3 Usage

3.1 Clam daemon

clamd is a multi-threaded daemon based on *libclamav*. It may work in one of the two following modes, listening on:

- Unix (local) socket
- TCP socket

The daemon is fully configurable via a *clamav.conf* file. You will find a description for every directive in the **clamav.conf(5)** manual. *clamd* recognizes the following commands:

- **PING**
  Check a daemon state (should reply with ”PONG”).
- **VERSION**
  Print version information.
- **RELOAD**
  Reload databases.
- **QUIT**
  Perform a clean exit.
- **SCAN file/directory** Scan a file or directory (recursively) with archive support enabled. A full path is required.
- **RAWSCAN file/directory** Scan a file or directory (recursively) with archive support disabled. A full path is required.
- **CONTSCAN file/directory** Scan a file or directory (recursively) with archive support enabled and don’t stop even if virus is found.
- **STREAM** Scan stream - clamd will return a new port number you should connect to and send a data to scan. *The protocol is obsolete and there will be a new version soon (however this one will still be supported).*
- **SESSION, END** Start/end a clamd session - you can do multiple commands per TCP session, however due to the clamd implementation the **RELOAD** command will break the session.

Internal threads (except clamuko) ignore all external signals. The main thread handles *SIGTERM* and *SIGINT* signals and performs a clean exit.

3.2 Clamuko

Clamuko is a special thread in *clamd* that performs on-access scanning under Linux. It was implemented as a thread in clamd due to the Dazuko implementation. Client (clamuko) - server (clamd) model is currently not supported by Dazuko. However there are some benefits of the current implementation - clamuko is sharing the internal virus database with clamd and it's updated with the RELOAD command. **You must obey the following important principles when using clamuko:**

- Always stop the daemon cleanly - using the QUIT command or SIGTERM signal. In other case you can lose your access to protected files until the system is restarted.
- Never protect a directory your mail-scanner software uses for attachment unpacking. Access to all infected files will be automagically blocked and the scanner (even clamd) won't be able to detect a virus. **The infected mail will be delivered.**

You need to enable clamuko in *clamav.conf*. To protect the /home directory enable the following directive:

```
ClamukoIncludePath /home
```

To protect the whole system:

```
ClamukoIncludePath /
ClamukoExcludePath /proc
ClamukoExcludePath /temporary/dir/of/your/mail/scanning/software
```

You can use clamuko to protect files on Samba/Netatalk (but far more better and safe idea is to use the **samba-vscan** module 5.17. NFS is not supported (Dazuko doesn't intercept NFS access calls). Yet another idea - you may build a database that contains signatures for popular exploits and setup clamd to protect your server from script-kiddies.

3.3 Archives and compressed files

All ClamAV scanners depend on LibClamAV. It has a built-in support for the following formats:

- Zip
- Gzip

- Bzip2
- RAR (2.0 only)

Archive file types are determined by magic number tests.[5] You need the zlib library for the Zip/Gzip support. Zip archives are accessed with the zziplib library by Guido Draheim and Tomi Ollila. RAR support is based on the UniquE RAR File Library by Christian Scheurer and Johannes Winkelmann. Both of them are included and slightly modified in the clamav sources. Unrarlib supports RAR 2.0 archives only and according to Christian the new format (introduced in WinRAR 3.0) will never be supported (however clamscan can scan WinRAR 3.0 archives, see below). Due to security reasons clamd only scans archives supported by libclamav. Clamscan is more clever and it can also use external unpackers - this is especially useful when the built-in decompresor fails:

```
$ clamscan --unrar rarfail.rar
/home/zolw/Clam/test/rarfail.rar: RAR module failure.

UNRAR 3.00 freeware      Copyright (c) 1993-2002 Eugene Roshal


Extracting from /home/zolw/Clam/test/rarfail.rar

Extracting  test1                                           OK
All OK
/tmp/44694f5b2665d2f4/test1: ClamAV-Test-Signature FOUND
/home/zolw/Clam/test/rarfail.rar: Infected Archive FOUND
```

TIP: *You can force clamscan to list all infected files in archive using –disable-archive (it disables the built-in decompressors) and –unzip –unrar....*

clamscan supports many popular compressors - it uses external programs for each format. **If the scanner runs with superuser privileges unpackers are executed with a *clamav* privileges what makes the process far more secure.** It also cares *clamav* user has read access to all files. **You must enable recursive scanning with the *-r* option (*–recursive*), if you want to scan a whole content of an archive (including subdirectories)**, this option is also (usually) required for scanning nested archive. External unpackers supported:

–unzip: Usually you don't need this option because Zip format is supported by libclamav. However it may be useful if libclamav fails to unzip some file. clamscan was

[5] It works similarly to the well known file(1) command.

tested with *UnZip 5.41 of 16 April 2000, by Info-ZIP*.

–unrar: Tested with *UNRAR 3.00 freeware*.

–unace: It uses an options supported by *UNACE v1.2 public version*, not tested, but should work.

–arj: Tested with *arj 3.10b*.

–zoo: Tested with *zoo 2.1*.

–lha: Tested with *LHa for Unix V 1.14e*.

–jar: clamscan uses *unzip* for .jar files. Tested with *UnZip 5.41 of 16 April 2000, by Info-ZIP*.

–tar: This option enables support for non-compressed archives. Tested with *GNU tar 1.13.17*.

–deb: This option enables support for debian binary packages. Tested with *GNU ar 2.12.90.0.14*. Implies –tgz , but doesn't conflict with –tgz=FULLPATH.

–tgz: This option supports .tar.gz and .tgz files. You need *GNU tar*, on non-Linux system you probably have it installed as *gtar* and if it can be found in *$PATH* please use *–tgz=gtar* to tell clamscan to use *gtar* instead of *tar*. Otherwise please supply a full path with *–tgz*.

3.4 Output format

clamd uses a clamscan compatible output format:

```
zolw@Wierszokleta:~$ telnet localhost 3310
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
SCAN /home/zolw/infected
/home/zolw/infected/sobre.com: W32/Magistr.B FOUND
Connection closed by foreign host.
```

In **SCAN** mode it closes the connection when first virus is found. In the case of archives the output is exactly the same as with normal files:

```
SCAN /home/zolw/Clam/test/test2.zip
/home/zolw/Clam/test/test2.zip: ClamAV-Test-Signature FOUND
```

CONTSCAN displays all infected files found.
Error messages are printed in the following format:

```
SCAN /no/such/file
/no/such/file: Can't stat() the file ERROR
```

and they can be easily parsed.

clamscan writes all messages to **stderr** (only help is written to **stdout** by default) to **stderr**. You may want to redirect it to **stdout** - this is handled by *–stdout*. An example clamscan output is:

```
/tmp/test/removal-tool.exe: Worm.Sober FOUND
/tmp/test/md5.o: OK
/tmp/test/blob.c: OK
/tmp/test/message.c: OK
/tmp/test/error.hta: VBS.Inor.D FOUND
```

When a virus is found, its name is printed between the *filename:* and *FOUND* strings. If a virus is found in an archive that has been extracted with an external unpacker it's noticed with *Infected Archive*. "Infected Archives" are not counted as infected files - only files within them are. Notice the difference with built-in unarchiver - extraction process is realized transparently by libclamav and clamscan doesn't know which concrete file is infected - just marks a whole archives as infected.

3.5 Signature Tool

sigtool automates signature creation. If you have got an infected file not recognized by ClamAV and there is another anti-virus scanner working in a console that detects the virus, then you can try to create the signature automatically. *Sigtool is only partially useful because it only detects a last part of a real signature. It will fail for multipart signatures (often used to match polymorphic viruses). Example of usage:* Create a random file (with any content) and insert the **test1** file content into it. We will use *clamscan* to generate the signature. This is only an example - in real life you don't need such tricks - just an infected file. Scan it with *clamscan –stdout testfile* - the output should be:

```
testfile: ClamAV-Test-Signature FOUND

----------- SCAN SUMMARY -----------
Known viruses: 20673
Scanned directories: 0
Scanned files: 1
Data scanned: 0.95 MB
Infected files: 1
I/O buffer size: 131072 bytes
Time: 0.245 sec (0 m 0 s)
```

The unique string in this output is ”ClamAV-Test-Signature” so run *sigtool* with the following arguments:

```
$ sigtool -c "clamscan --stdout" -f testfile -s "ClamAV-Test"
```

The program will concatenate arguments for *-c (–command)* and *-f (–file)*, that’s why the scanner’s options must be given in the proper order. At the end it will generate a file *testfile.sig*, which should contain 100 bytes in our example. It contains the proper signature.

```
Detected, decreasing end 20051 -> 16040
Detected, decreasing end 16040 -> 12029
Detected, decreasing end 12029 -> 8018
Not detected at 8018, moving forward.
Detected, decreasing end 10024 -> 8018
Not detected at 8018, moving forward.
Detected, decreasing end 9021 -> 8018
Not detected at 8018, moving forward.
Not detected at 8520, moving forward.
Detected, decreasing end 8771 -> 8520
Not detected at 8520, moving forward.
Not detected at 8646, moving forward.
Not detected at 8709, moving forward.
Detected, decreasing end 8741 -> 8709
Not detected at 8709, moving forward.
Not detected at 8725, moving forward.
Detected, decreasing end 8733 -> 8725
Not detected at 8725, moving forward.
Not detected at 8729, moving forward.
Detected, decreasing end 8731 -> 8729
Not detected at 8729, moving forward.
Detected, decreasing end 8730 -> 8729
Not detected at 8729, moving forward.
Increasing end 8729 -> 8730
```

```
 *** Signature end found at 8730
Detected at 8680, moving forward.
Detected at 8680, moving forward.
Not detected, moving backward 8693 -> 8680
Detected at 8680, moving forward.
Not detected, moving backward 8687 -> 8680
Detected at 8680, moving forward.
Not detected, moving backward 8684 -> 8680
Detected at 8680, moving forward.
Not detected, moving backward 8682 -> 8680
Detected at 8680, moving forward.
Not detected, moving backward 8681 -> 8680
Detected at 8680, moving forward.
Not detected, moving backward 8681 -> 8680
Detected at 8680, moving forward.
Moving forward 8680 -> 8681
 *** Signature start found at 8681

The scanner was executed 33 times.
The signature length is 49 (98 hex)
Saving signature in testfile.sig file.
Saving binary signature in testfile.bsig file.
```

To make the generated signature complete you only to add the *VirusName=* string at the beginning of the hexadecimal signature in testfile.sig.

TIP: *ClamAV scanners read all .db files in the database directory. You can create your own database files (e.g. local.db) and they won't be modified by freshclam !*

4 Problem solving

4.1 Return codes

Return codes are very useful, especially in system scripts. You may check the return code from *clamscan*, by running the following command:

```
$ clamscan; echo Return code: $?
```

Here is a list of return codes for *clamscan*:

0: No virus was found.

1: Virus(es) detected.

40: Unknown option passed to *clamscan*. Please check *clamscan –help* or manual page for available options.

50: Virus database initialization error. Probably it doesn't exist at the default location or it's malformed (e.g. broken digital signature)

52: Not supported file type - clamscan only supports regular files, directories and symlinks.

53: Can't open directory.

54: Can't open file.[6]

55: I/O error during read. [6]

56: Can't stat input file or directory. File (or directory) you want to scan doesn't exist.

57: Can't get absolute pathname of current working directory. Your current pathname is longer then 200 characters. This is bad and you may need to recompile ClamAV to fix it.

58: I/O error. Please check the filesystem.

59: Can't get information about current user (running clamscan).

60: Can't get information about user *clamav*. User *clamav* (default unprivileged user) doesn't exist in /etc/passwd.

61: Can't fork. Can't create new process, please check your system limits.

63: Can't create temporary file or directory. Please check /tmp permissions or use –tempdir

64: Can't write to temporary directory. Please specify another one.

70: Can't allocate and clear memory. This is a critical error, please check your system.

71: Can't allocate memory. Look above.

5 Certified software

There are many projects that support ClamAV. Here is a list of software that was tested and is known to work well.

[6] Only in one-file mode (in recursive mode those errors are ignored).

5.1 clamav-milter

Homepage: included in clamav package
Supports: clamd

Nigel Horne's clamav-milter is a very fast email scanner designed for sendmail. It's written entirely in C and uses ClamAV's internal mail scanner (also written by Nigel).

Installation:
You need libmilter development files. Configure ClamAV with

```
    $ ./configure --enable-milter
```

and recompile. The program will be installed in /usr/local/sbin/clamav-milter. The following instructions were adopted from Nigel's INSTALL file: add to /etc/mail/sendmail.mc:

```
INPUT_MAIL_FILTER('clmilter','S=local:/var/run/clmilter.sock,
F=, T=S:4m;R:4m')dnl
define('confINPUT_MAIL_FILTERS', 'clmilter')
```

Check entries in clamav.conf of the form:

```
LocalSocket /var/run/clamd.sock
ScanMail
StreamSaveToDisk
```

Start clamav-milter:

```
/usr/local/sbin/clamav-milter -lo /var/run/clmilter.sock
```

and restart sendmail.

5.2 IVS Milter

Homepage: `http://ivs-milter.lbsd.net`
Supports: clamd

IVS Milter is a virus and spam scanning milter. The name stands for Industrial Virus + Spam milter. It's designed to be used by anything from home users to large ISP's.

5.3 smtp-vilter

Homepage: `http://www.etc.msys.ch/software/smtp-vilter`
Supports: clamd

smtp-vilter is a high performance content filter for sendmail using the milter API. The software scans e-mail messages for viruses and drops or marks infected messages. ClamAV is the default scanner backend.

5.4 mod_clamav

Homepage: `http://software.othello.ch/mod_clamav`
Supports: libclamav, clamd

mod_clamav is an Apache virus scanning filter. It was written and is currently maintained by Andreas Muller. The project is very well documented and the installation is quite easy.

5.5 TrashScan

Homepage: clamav-sources/support/trashscan
Supports: clamscan

This is a procmail based scanner from Trashware and it's extremely easy to setup, however this is for single users only and not as efficient as MTA based scanners.

5.6 AMaViS - ”Next Generation”

Homepage: `http://sourceforge.net/projects/amavis`
Supports: clamscan

AMaViS-ng is a rewritten, more modular version of amavis-perl/amavisd, developed by Hilko Bengen. Home site:

Installation:

Please download the newest version (at least 0.1.4). After installation (which is quite easy), please uncomment the following line in amavis.conf:

```
virus-scanner = CLAM
```

and eventually change the path to clamscan in the *[CLAM]* section:

```
[CLAM]
```

```
clamscan = /usr/local/bin/clamscan
```

5.7 amavisd-new

Homepage: `http://www.ijs.si/software/amavisd`
Supports: clamd, clamscan

amavisd-new is a rewritten version of amavis maintained by Mark Martinec.

Installation:
clamscan is enabled automatically if clamscan binary is found at amavisd-new starup time. clamd is activated by uncommenting its entry in the @av_scanners list, file /etc/amavisd.conf.

5.8 Qmail-Scanner

Homepage: `http://qmail-scanner.sf.net`
Supports: clamscan

You must increase softlimit value or wait for a daemon support.

5.9 Sagator

Homepage: `http://www.salstar.sk/sagator`
Supports: clamscan, clamd, libclamav

This program is an email antivirus/antispam gateway. It is an interface to the postfix (or any other smtpd), which runs antivirus and/or spamchecker. Its modular architecture can use any combination of antivirus/spamchecker according to configuration.

5.10 ClamdMail

Homepage: `http://clamdmail.sf.net`
Supports: clamd

A mail processing client for ClamAV. Small, fast and easy to install.

5.11 BlackHole

Homepage: `http://iland.net/˜ckennedy/blackhole.shtml`
Supports: clamscan, clamd

BlackHole is an advanced spam / virus filter for Qmail, Postfix, Sendmail, Exim and Courier written by Chris Kennedy. This tool is for advanced administrators (installation is hard).

5.12 MailScanner

Homepage: `http://www.mailscanner.info`
Supports: clamscan

MailScanner scans all e-mail for viruses, spam and attacks against security vulnerabilities. It is not tied to any particular virus scanner, but can be used with any combination of 14 different virus scanners, allowing sites to choose the ”best of breed” virus scanner.

5.13 MIMEDefang

Homepage: `http://www.roaringpenguin.com/mimedefang`
Supports: clamscan, clamd

This is an efficient mail scanner for Sendmail/milter.

5.14 exiscan

Homepage: `http://duncanthrax.net/exiscan`
Supports: clamscan, clamd

exiscan is a patch against exim version 4, providing support for content scanning in email messages received by exim. Four different scanning facilities are supported: antivirus, antispam, regular expressions, and file extensions.

5.15 scanexi

Homepage: `http://w1.231.telia.com/˜u23107873/scanexi.html`
Supports: clamscan, clamd

scanexi is a plugin for exim version 4.14 with dlopen patch, providing support for content scanning in email messages received by exim.

5.16 Mail::ClamAV

Homepage: `http://cpan.gossamer-threads.com/modules/by-authors/id/S/SA/SABECK/`
Supports: libclamav

Perl extension for the libclamav library.

5.17 OpenAntiVirus samba-vscan

Homepage: `http://www.openantivirus.org/projects.php#samba-vscan`
Supports: clamd

samba-vscan provides on-access scanning of Samba shares. It supports Samba 2.2.x/3.0 with working virtual file system (VFS) support.

5.18 Sylpheed Claws

Homepage: `http://claws.sylpheed.org`
Supports: libclamav

Sylpheed Claws is a bleeding edge branch of Sylpheed, a light weight mail user agent for UNIX. It can scan attachments in mail received from POP, IMAP or a local account and optionally delete the mail or save it to a designated folder.

5.19 nclamd

Homepage: `http://www.kyzo.com/nclamd/`
Supports: libclamav

nclamd, nclamav-milter and nclamdscan are rewritten versions of the original tools and use processes instead of threads and ripMIME instead of the clamav built-in MIME decoder.

5.20 cgpav

Homepage: `http://program.farit.ru`
Supports: clamd

This is a fast (written in C) CommuniGate Pro anti-virus plugin with support for clamd.

6 LibClamAV

libclamav may be used to add a virus protection to your software. The library is thread-safe, automatically recognizes and scans archives. Scanning is very fast - in most cases it won't be even noticeable.

6.1 General API

Every program based on libclamav must include the *clamav.h* header file:

```
#include <clamav.h>
```

A first step is to initialize the scanning engine. There are three functions available:

```
int cl_loaddb(const char *filename, struct cl_node **root,
int *virnum);

int cl_loaddbdir(const char *dirname, struct cl_node **root,
int *virnum);

char *cl_retdbdir(void);
```

cl_loaddb() loads a particular database, *cl_loaddbdir()* loads all *.cvd* (and older *.db*, .db2) databases from a directory *dirname*. *cl_retdbdir()* returns a hardcoded database directory path. Initial internal database (Aho-Corasick tree, trie; see 6.3) will be saved under *root* and a number of signatures loaded will be **added** [7] to *virnum*. Pointer to the trie must initially point to NULL. If you don't care about number of signatures pass NULL as a third argument. *cl_loaddb* functions return 0 on success and other value on failure.

```
    struct cl_node *root = NULL;
    int ret;

ret = cl_loaddbdir(cl_retdbdir(), &root, NULL);
```

There's an elegant way to print libclamav's error codes:

```
char *cl_strerror(int clerror);
```

cl_strerror() returns a (statically allocated) string describing a *clerror* code:

```
if(ret) {
    printf("cl_loaddbdir() error: %s\n", cl_strerror(ret));
    exit(1);
}
```

When database is loaded you must build the final trie with:

```
void cl_buildtrie(struct cl_node *root);
```

In our example:

[7] Remember to initialize the virus counter variable with 0.

```
cl_buildtrie(root);
```

OK, now you can scan a buffer, a descriptor or a file with:

```
int cl_scanbuff(const char *buffer, unsigned int length,
char **virname, const struct cl_node *root);

int cl_scandesc(int desc, char **virname, unsigned long int
*scanned, const struct cl_node *root, const struct cl_limits
*limits, int options);

int cl_scanfile(const char *filename, char **virname,
unsigned long int *scanned, const struct cl_node *root,
const struct cl_limits *limits, int options);
```

All the functions save a virus name address under *virname* pointer. *virname* points to a name in the trie structure thus it can't be released directly. *cl_scandesc()* and *cl_scanfile()* can increase the *scanned* value in CL_COUNT_PRECISION units, they also support archive limits:

```
struct cl_limits {
    int maxreclevel;
    int maxfiles;
    long int maxfilesize;
};
```

The last argument configures scan engine. Currently it supports **CL_ARCHIVE** (enables archive scanning), **CL_RAW** (disables archive scanning) and **CL_MAIL** (enables mbox and Maildir scanning) and **CL_DISABLERAR** (disables the built-in RAR unpacker which leaks like hell). These functions return 0 (**CL_CLEAN**) when no virus is found, **CL_VIRUS** when virus is found and other value on failure.

```
    struct cl_limits limits;
    char *virname;

memset(&limits, 0, sizeof(struct cl_limits));
/* maximal number of files in archive */;
limits.maxfiles = 1000
/* maximal archived file size == 10 MB */
limits.maxfilesize = 10 * 1048576;
```

```
/* maximal recursion level */
limits.maxreclevel = 5;

if((ret = cl_scanfile("/home/zolw/test", &virname, NULL, root,
&limits, CL_ARCHIVE)) == CL_VIRUS) {
    printf("Detected %s virus.\n", virname);
} else {
    printf("No virus detected.\n");
    if(ret != CL_CLEAN)
        printf("Error: %s\n", cl_strerror(ret));
}
```

Release the trie if you no longer need it:

```
void cl_freetrie(struct cl_node *root);
```

You will find an example scanner in clamav sources (/example). Program based on libclamav must be linked against it:

```
gcc -Wall ex1.c -o ex1 -lclamav
```

Enjoy !

6.2 Database reloading

The most important thing is to keep the memory database representation up to date. You can watch database changes with the *cl_stat* functions family:

```
int cl_statinidir(const char *dirname, struct cl_stat *dbstat);
int cl_statchkdir(const struct cl_stat *dbstat);
int cl_statfree(struct cl_stat *dbstat);
```

Initialization:

```
    struct cl_stat dbstat;

memset(&dbstat, 0, sizeof(struct cl_stat));
cl_statinidir(dbdir, &dbstat);
```

To check for a change you only need to call the following function:

```
if(cl_statchkdir(&dbstat) == 1) {
    reload_database...;
    cl_statfree(&dbstat);
    cl_statinidir(cl_retdbdir(), &dbstat);
}
```

Remember to reinitialize the structure after a reload.

6.3 Scan engine

New versions of Clam AntiVirus use a mutation of the Aho-Corasick pattern matching algorithm. The algorithm is based a finite state pattern matching automaton [1] and it's a generalization of the famous Knuth-Morris-Pratt algorithm. Please take a look at the *matcher.h* for data type definitions. The automaton is represented by a trie. It is a rooted tree with some specific properties [2]. Every node of the trie represents some state of the automaton. In our implementation, the node is defined as follows:

```
struct cl_node {
    short int islast;
    struct cli_patt *list;
    int maxpatlen;
    struct node *next[NUM_CHILDS], *trans[NUM_CHILDS], *fail;
};
```

[To be continued...]

6.4 CVD format

CVD (ClamAV Virus Database) is a digitally signed tarball file that contains one or more databases. You can find some useful information in the ASCII header of the file. It's a 512 bytes long string with the following colon separated fields:

```
ClamAV-VDB:build time:version:number of signatures:functionality
level required:MD5 checksum:digital signature:builder name
```

and can be easily parsed by scripts or with *sigtool –info*. There are two CVD databases in ClamAV: *main.cvd* and *daily.cvd* for a daily updates. You can use *sigtool* to unpack a CVD file (*–unpack*) and to list virus names (*–list-sigs*).

7 Credits

7.1 Contributors

The following people contributed to our project in some way (providing patches, bug reports, technical support, documentation, good ideas...):

- Kamil Andrusz <wizz*mniam.net>
- Jean-Edouard BABIN <Jeb*jeb.com.fr>
- Marc Baudoin <babafou*babafou.eu.org>
- Rene Bellora <rbellora*tecnoaccion.com.ar>
- Hilko Bengen <bengen*vdst-ka.inka.de>
- Patrick Bihan-Faou <patrick*mindstep.com>
- Oliver Brandmueller <ob*e-Gitt.NET>
- Igor Brezac <igor*ipass.net>
- Brian Bruns <bruns*2mbit.com>
- Len Budney <lbudney*pobox.com>
- Matt Butt <mattb*cre8tiv.com>
- Eric I. Lopez Carreon <elopezc*technitrade.com>
- Andrey Cherezov <andrey*cherezov.koenig.su>
- Nicholas Chua <nicholas*ncmbox.net>
- Christoph Cordes <ib*precompiled.de>
- Eugene Crosser <crosser*rol.ru>
- Damien Curtain <damien*pagefault.org>
- Krisztian Czako <slapic*linux.co.hu>
- Diego d’Ambra <da*softcom.dk>
- Michael Dankov <misha*btrc.ru>
- Alejandro Dubrovsky <s328940*student.uq.edu.au>

- Magnus Ekdahl <magnus*debian.org>
- Jason Englander <jason*englanders.cc>
- Oden Eriksson <oden.eriksson*kvikkjokk.net>
- Andy Fiddaman <af*jeamland.org>
- Edison Figueira Junior <edison*brc.com.br>
- David Ford <david+cert*blue-labs.org>
- Free Oscar <freeoscar*wp.pl>
- Piotr Gackiewicz <gacek*intertele.pl>
- Jeremy Garcia <jeremy*linuxquestions.org>
- Nick Gazaloff <nick*sbin.org>
- Luca ’NERvOus’ Gibelli <nervous*nervous.it>
- Wieslaw Glod <wkg*x2.pl>
- Matthew A. Grant <grantma*anathoth.gen.nz>
- Hrvoje Habjanic <hrvoje.habjanic*zg.hinet.hr>
- Michal Hajduczenia <michalis*mat.uni.torun.pl>
- Jean-Christophe Heger <jcheger*acytec.com>
- Anders Herbjornsen <andersh*gar.no>
- Paul Hoadley <paulh*logixsquad.net>
- Robert Hogan <robert*roberthogan.net>
- Przemyslaw Holowczyc <doozer*skc.com.pl>
- Thomas W. Holt Jr. <twh*cohesive.net>
- Douglas J Hunley <doug*hunley.homeip.net>
- Kurt Huwig <kurt*iku-netz.de>
- Andy Igoshin <ai*vsu.ru>
- Jay <sysop-clamav*coronastreet.net>

- Stephane Jeannenot <stephane.jeannenot*wanadoo.fr>
- Dave Jones <dave*kalkbay.co.za>
- Alex Kah <alex*narfonix.com>
- Stefan Kaltenbrunner <mm-mailinglist*madness.at>
- Kazuhiko <kazuhiko*fdiary.net>
- Tomasz Klim <tomek*euroneto.pl>
- Robbert Kouprie <robbert*exx.nl>
- Martin Kraft <martin.kraft*fal.de>
- Petr Kristof <Kristof.P*fce.vutbr.cz>
- Henk Kuipers <henk*opensourcesolutions.nl>
- Nigel Kukard <nkukard*lbsd.net>
- Dr Andrzej Kurpiel <akurpiel*mat.uni.torun.pl>
- Thomas Lamy <Thomas.Lamy*in-online.net>
- Marty Lee <marty*maui.co.uk>
- Dennis Leeuw <dleeuw*made-it.com>
- Martin Lesser <admin-debian*bettercom.de>
- Peter N Lewis <peter*stairways.com.au>
- Mike Loewen <mloewen*sturgeon.cac.psu.edu>
- David S. Madole <david*madole.net>
- Thomas Madsen <tm*softcom.dk>
- Bill Maidment <bill*maidment.com.au>
- Andrey V. Malyshev <amal*krasn.ru>
- Stefan Martig <sm*officeco.ch>
- Serhiy V. Matveyev <matveyev*uatele.com>
- Reinhard Max <max*suse.de>

- Brian May <bam*debian.org>
- Ken McKittrick <klmac*usadatanet.com>
- Chris van Meerendonk <cvm*castel.nl>
- Arkadiusz Miskiewicz <misiek*pld-linux.org>
- Mark Mielke <mark*mark.mielke.cc>
- Jo Mills <Jonathan.Mills*frequentis.com>
- Dustin Mollo <dustin.mollo*sonoma.edu>
- Doug Monroe <doug*planetconnect.com>
- Alex S Moore <asmoore*edge.net>
- Flinn Mueller<flinn*activeintra.net>
- Hendrik Muhs <Hendrik.Muhs*student.uni-magdeburg.de>
- Farit Nabiullin http://program.farit.ru
- Wojciech Noworyta <wnow*konarski.edu.pl>
- Joe Oaks <joe.oaks*hp.com>
- Washington Odhiambo <wash*wananchi.com>
- Masaki Ogawa <proc*mac.com>
- Martijn van Oosterhout <kleptog*svana.org>
- OpenAntiVirus Team (http://www.OpenAntiVirus.org)
- Tomasz Papszun <tomek*lodz.tpsa.pl>
- Eric Parsonage <eric*eparsonage.com>
- Oliver Paukstadt <pstadt*stud.fh-heilbronn.de>
- Rudolph Pereira <r.pereira*isu.usyd.edu.au>
- Ed Phillips <ed*UDel.Edu>
- Andreas Piesk <Andreas.Piesk*heise.de>
- Alex Pleiner <pleiner*zeitform.de>

- Ant La Porte <ant*dvere.net>
- Sergei Pronin <sp*finndesign.fi>
- Thomas Quinot <thomas*cuivre.fr.eu.org>
- Ed Ravin <eravin*panix.com>
- David Sanchez <dsanchez*veloxia.com>
- David Santinoli <david*santinoli.com>
- Vijay Sarvepalli <vssarvep*office.uncg.edu>
- Martin Schitter
- Enrico Scholz <enrico.scholz*informatik.tu-chemnitz.de>
- Scsi <scsi*softland.ru>
- Ole Stanstrup <ole*stanstrup.dk>
- Richard Stevenson <richard*endace.com>
- Matt Sullivan <matt*sullivan.gen.nz>
- Dr Zbigniew Szewczak <zssz*mat.uni.torun.pl>
- Joe Talbott <josepht*cstone.net>
- Gernot Tenchio <g.tenchio*telco-tech.de>
- Masahiro Teramoto <markun*onohara.to>
- Trashware <trashware*gmx.net>
- Daniel Mario Vega <dv5a*dc.uba.ar>
- Laurent Wacrenier <lwa*teaser.fr>
- Nicklaus Wicker <n.wicker*cnk-networks.de>
- David Woakes <david*mitredata.co.uk>
- Troy Wollenslegel <troy*intranet.org>
- Leonid Zeitlin <lz*europe.com>
- Andoni Zubimendi <andoni*lpsat.net>

7.2 Donors

We received financial support from:

- ActiveIntra.net Inc. (http://www.activeintra.net)
- Anonymous donor from Colorado, US
- Dynamic Network Services, Inc (http://www.dyndns.org)
- Electric Embers
- epublica
- David Eriksson (http://www.2good.nu)
- Explido Software USA Inc. (http://www.explido.us)
- David Farrick
- Andries Filmer (http://www.netexpo.nl)
- Jeremy Garcia (http://www.linuxquestions.org)
- Invisik Corporation (http://www.invisik.com)
- Keith (http://www.textpad.com)
- Brad Koehn
- Stephane Rault
- cheahch from Singapore
- Nicklaus Wicker <n.wicker*cnk-networks.de>

8 Authors

8.1 Virus Database Developers

Virus database is a heart of every anti-virus software. The following people care ClamAV's heart is in a good condition:

- aCaB <acab*clamav.net>
- Christoph Cordes <ccordes*clamav.net>

- Diego D'Ambra <diego*clamav.net>
- Jason Englander <jason*clamav.net>
- Tomasz Kojm <tkojm*clamav.net>
- Denis De Messemacker <ddm*clamav.net>
- Tomasz Papszun <tomek*clamav.net>

Our database includes the virus database (about 5000 signatures) from OpenAntiVirus.org.

8.2 Network management

Thanks to Luca 'NERvOus' Gibelli <nervous*clamav.net> you can download our database from all the mirrors listed in 2.8. Luca is also responsible for our main site www.clamav.net, mailing lists, and the virus submission mechanism.

8.3 Graphics

The authors of the nice ClamAV logo (look at the title page) are Mia Kalenius and Sergei Pronin <sp*finndesign.fi>.

8.4 Core developers

Nigel Horne <njh*clamav.net> is a very active ClamAV developer responsible for the mbox code and clamav-milter. Trog trog*clamav.net> developes the OLE2 code and introduces new features into ClamAV. Thomas Lamy is a great memory leak killer and code stablizer. Tomasz Kojm (me) navigates the project and keeps an eye on everything 8-)

Tomasz Kojm <tkojm*clamav.net>

References

[1] Cormen, Leiserson, Rivest: *Introduction to Algorithms*, Chapter 34, MIT Press.

[2] http://www-sr.informatik.uni-tuebingen.de/˜buehler/AC/AC.html: Aho-Corasick algorithm description