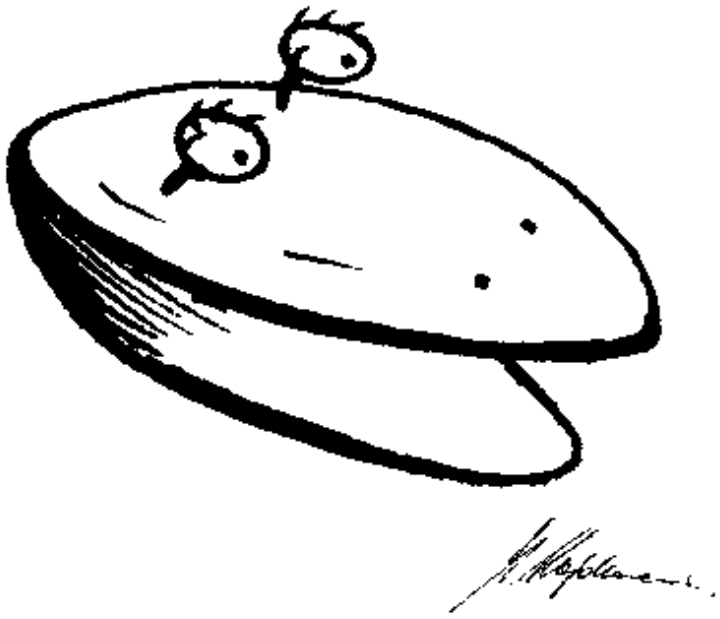

Clam Antivirus: User Manual

Tomasz Kojm

1 Introduction

Clam Antivirus was born in April '2002. It uses the database from OpenAntiVirus, which is another free anti-virus project. Clam Antivirus was written from scratch in a period of one month (first public available version). I hope, it will still be in development [1] when you read this. OpenAntiVirus is a good project, but it supports only Java and lacks a command line scanner. That's why ClamAV in the first stage is scanner-minded.

1.1 Features

- GNU GPL
- POSIX compliant, portable
- Secure
- Very fast
- Multithreaded
- Userfriendly
- Detects over 1700 viruses, even some popular Linux worms
- Supported by AMaViS - "Next Generation", Qmail-Scanner, MIMEDefang
- Supports AMaViS-perl
- Scans compressed files
- (Auto) Updatable via Internet
- A bit documented ;)

2 Installation

2.1 Supported platforms

Clam Antivirus is prepared to install and run without problems on the following operating systems / architectures (tested platforms in brackets):

- GNU/Linux 2.2/2.4 (All flavours, Intel/SPARC/Alpha)

[1] Clam Antivirus is developed on Debian GNU/Linux

- Solaris 2.6/7/8 (Intel/SPARC)
- FreeBSD 4.5 (Intel/Alpha)
- OpenBSD 3.0/3.1 (Intel)
- NetBSD[2]
- AIX 4.1/4.2/4.3/5.1 (RISC 6000)
- Mac OS X
- Cobalt MIPS boxes (RAQ1, RAQ2, QUBE2)
- Cygwin B20 (Windows 98)

These systems are detected during the installation process and required options are set. If you have run Clam Antivirus on other systems, please let me know. If your system requires some changes, please look at the *configure.in* file.

2.2 Current versions

Clam Antivirus can be obtained from:

`http://clamav.elektrapro.com`

The site is sponsored by ElektraPro.com

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 may find it on debian mirrors, look at `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`). The binary packages for AIX are available in AIX PDSLIB, UCLA `http://aixpdslib.seas.ucla.edu/packages/clamav.html`.

[2]Threads are not supported, because NetBSD lacks POSIX semaphore support.

2.4 Installation

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

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

The above method works on Linux and Solaris, if you don't have *groupadd*, *useradd* please consult your system manual, the section about creating new users and groups. If you are not a system administrator, and won't be using Clam Antivirus in superuser mode, you may omit this step with the option *–disable-clamav* passed to configure:

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

This disables testing for *clamav* user and group. Clam Antivirus still requires *clamav* for superuser mode. Please don't set a password on this account, just assure it's locked with "**!**" in */etc/passwd* or */etc/shadow*. It must be a normal, unprivileged user. Don't add it to any special groups.

Hint: If your system uses quotas it's a good idea to set bigger quota on this account. *clamscan* will extract compressed files to a temporary directory (default */tmp* or user selected) with *clamav* priviledges. I strongly advise you to use filesystem quota for *clamav* if you want to avoid security problems. Some protections have already been implemented, check **clamscan(1)** manual for details.

After this, extract the archive, configure and compile:

```
$ tar zxpvf clamav-x.yz.tar.gz
$ cd clamav-x.yz
$ ./configure; make
$ su -c "make install"
```

That's all !

WARNING: Never set SUID/SGID bit on Clam Antivirus programs.

[3] Cygwin note: If you don't have /etc/passwd, you don't need the *clamav* user/group.

2.5 Testing

Ok. Let's do some tests. Try to scan the source directory recursively:

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

It should find the viruses in the clamav-x.yz/test directory. You may check it in the created log - scan.txt. **You will find more about clamscan options in the clamscan(1) manual.** [4]

2.6 Setting up auto-updating

freshclam is a tool, which automates the virus database update process for Clam Antivirus. You will find more information about it in the section 3.4. It may be used in two ways:

- interactive - from command line
- as a daemon - works alone, silently

Run *freshclam* (as root) without any parameters to check if it is working correctly. If everything is ok, create a log file in /var/log which should be owned by *clamav*:

```
# touch /var/log/clam-update.log
# chmod 644 /var/log/clam-update.log
# chown clamav /var/log/clam-update.log
```

Now, you may run *freshclam* as a daemon:

```
# freshclam -d -c 2 -l /var/log/clam-update.log
```

It will check for the new database 2 times a day. Please add this line to your startup scripts. The other way is to use the *cron* daemon. You have to add a similar line to the crontab of **root** or **clamav**:

```
0 8 * * * /usr/local/bin/freshclam --quiet -l /var/log/clam-update.log
```

It will check for the new database daily at 8 am.

[4] Please run *man clamscan*

2.7 AMaViS - ”Next Generation”

This is prefered. AMaViS-ng is a rewritten, more modular version of amavis-perl/amavisd, developed by Hilko Bengen. You may find it on:

```
http://sourceforge.net/projects/amavis
```

Please download the newest version (at least 0.1.3). 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
```

2.8 Support for AMaViS-perl

The first thing you need is amavis-perl-11.tar.gz from `http://amavis.org`, then

```
$ tar zxpvf amavis-perl-11.tar.gz
$ cp clamav-x.yz/support/amavis/clamavis.patch amavis-perl-11
$ cd amavis-perl-11
$ patch -p1 < clamavis.patch
$ find . -exec touch 01010000 {} \;
```

Now please do a standard AMaViS installation.

Hint: AMaViS will use *clamscan* with standard options, which should work on all systems. If you want to add other options (eg. decompression, limits) please edit the file */usr/sbin/amavis* after installation.

2.9 Installation problems

You shouldn’t have any problems with installation of Clam Antivirus. The only problem you may find is POSIX threads support, because some systems lack pthreads support. The Clam Antivirus installation process detects the presence of the pthread library and enables or disables POSIX threads support. But your system may be broken, so please try the following configure option:

```
$ ./configure --disable-pthreads
```

If it doesn't help, please contact me.

3 Usage

3.1 Archives and compressed files

Clam Antivirus supports many popular compressors. It uses external programs for each format. It doesn't have built-in code due to licences and the number of formats. **If the scanner runs with superuser priviledges unpackers are executed with *clamav* priviledges, which makes Clam Antivirus far more secure.** It also makes sure, that *clamav* has read access to all scanned compressed files. **You should enable recursive scanning with the *-r* option (*–recursive*), if you want to scan all contents of archives (with subdirectories)**, also all archives in archives will be recursively scanned - just everything. If files in archives are virus free the archive itself is scanned - just for prevention (it may not be an archive). Please look at the options below, each option has an optional argument - the absolute path to unpacker. If it can't be found in *$PATH* please supply it. *Because Clam Antivirus uses the standard GNU options format, the long options with optional arguments, you **must** remember about the = between option and argument. So the proper way to supply optional arguments is for example –unzip=/path/to/unzip.*

–unzip: zip is the most popular compression format. With this option all zipped files are scanned. If you turn on this option without the argument, the *unzip* program will be searched in *$PATH*. Clam Antivirus was tested with *UnZip 5.41 of 16 April 2000, by Info-ZIP*.

–unrar: Tested with *UNRAR 2.50 freeware*.

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

–unarj: Tested with *UNARJ (Demo version) 2.41a*.

–zoo: Tested with *zoo 2.1*.

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

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

–tar: This option supports non-compressed archives. Tested with *GNU tar*

1.13.17.

–deb: This option supports 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 as *gtar* and if this is in *$PATH* just use *–tgz=gtar* or supply the full path to this command as an argument.

3.2 Output format

clamscan writes all messages (only help is written to **stdout** by default) to **stderr**. In some situations you may want to redirect it to **stdout** with *–stdout*. *stdout* in contrast to *stderr* is buffered, that's why *clamscan* flushes this buffer after each message, to prevent the creation of trashes on the output. During scanning it writes something like this:

```
/TEST/test: OK
/TEST/Makefile: OK
/TEST/getopt.c: OK
/TEST/virfile: Phantom #1 FOUND
```

When a virus is found, its name is printed between *filename:* and *FOUND*. The example below shows scanning zip files in the current directory (*/TEST*), with the command *clamscan -r –unzip*:

```
Archive:  /TEST/virii.zip
 extracting: ph
  inflating: others.c
  inflating: others.h
 extracting: vir.zip
/tmp/a85793d0a566631f/ph: Phantom #1 FOUND
/tmp/a85793d0a566631f/others.c: OK
/tmp/a85793d0a566631f/others.h: OK
Archive:  /tmp/a85793d0a566631f/vir.zip
 extracting: nan
  inflating: options.c
  inflating: options.h
/tmp/0521ea0370ad3c49/anke: OK
/tmp/0521ea0370ad3c49/options.c: OK
/tmp/0521ea0370ad3c49/options.h: OK
```

```
/tmp/a85793d0a566631f/vir.zip: OK
/TEST/virii.zip: Infected Archive FOUND
```

As you can see, zip files inside the zip file were scanned. If a virus is found in the (compressed) archive, it's noticed with *Infected Archive*. Infected archives are not counted as infected files - just files in them are. After scanning you should see *Scan summary* (it may be disabled with *–disable-summary*. It looks like:

```
----------- SCAN SUMMARY -----------
Known viruses: 1773
Scanned directories: 18
Scanned files: 172
Data scanned: 31.33 Mb
Infected files: 9
I/O buffer size: 131072 bytes
Time: 5.068 sec (0 m 5 s)
```

3.3 OpenAntiVirus Update

oav-update is currently the best autoupdater for ClamAV. Its main features are:

- proxy support
- downloading from multiple sources
- verification of each .credo file using the OAV public keys, including handling trust levels.
- addition of local signatures
- email notification of errors and problems

The script is written in Perl by Matthew Grant and may be obtained from `http://www.anathoth.gen.nz/tarballs`. It downloads the database directly from `http://www.openantivirus.org`.

3.4 FreshClam

The *freshclam* utility is the default database updater for Clam Antivirus. It works in two modes - from the command line and as a daemon. When started by the superuser it drops the priviledges, by default it works as *clamav*. *freshclam* downloads the database from the Clam Antivirus homepage and checks its consistency

using MD5 sum. More information about the usage of this program you will find in **freshclam(1)** manual. *FreshClam has a few disadvantages: it lacks proxy support and downloads the database from a hardcoded site only. If you need these features, please use oav-update.*

3.5 Signature Tool

sigtool automates signature creation. If you have a infected file, which isn't detected by ClamAV, but is by another anti-virus scanner working in the console you can create the signature easily. *Example of usage:* Create a file and put the **eicar.com** content into it. We will use *clamscan* to generate the signature, it's just an example. Scan it with *clamscan –stdout testfile*, the output is

```
testfile: Eicar-Test-Signature FOUND

----------- SCAN SUMMARY -----------
Known viruses: 1773
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 "Eicar-Test-Signature". Run *sigtool* with the following parameters:

```
$ sigtool -c "clamscan --stdout" -f testfile -s "Eicar-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 80 bytes in our example. It contains the proper signature.

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 directly after the scanner exits:

```
$ echo $?
```

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

0: No virus was found.

1: Virus(es) detected.

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

50: Problem with initialization of virus database. Probably it doesn't exist in the default place or wrong file was passed to *–database*.

51: Wrong number of threads was passed to *–threads*. It must be a natural number ≥ 0.

52: Not supported file type. Scanner supports regular files, directories and symlinks.

53: Can't open directory.

54: Can't open file.[5]

55: Error reading file. Probably the medium you are reading is broken. [5]

56: Can't stat input file or directory. File / 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. When clamscan is started without a input file / directory it scans the current directory. For some reasons it needs absolute pathnames, the buffer is hardcoded to 200 characters and that should be sufficient.

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 limits.

62: Can't execute selected unpacker. You selected a program that doesn't exist or you don't have permissions to execute it.

63: Can't create temporary file or directory. Please check permissions.

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

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

system.

71: Can't allocate memory. Look above.

5 Technicals

5.1 Security

Clam Antivirus cares about security. Dangerous operations (such as extracting, temporary file creation, unlink() operations) are executed with *clamav* priviledges. **But there are no programs without bugs.** This is young project and everything is possible. In some places it uses the *snprintf()* function, some older systems (C libraries) however the buffer length in this function isn't checked. This example shows, that you should check your system first. Never set SUID/SGID bits on Clam Antivirus executables. If the SUID bit is set and *clamscan* is owned by root, every file on the system may be modified with the *–log* option. Normal users may use *clamscan* to scan their files, other files shouldn't interest them.

5.2 Scan engine

New versions of the Clam Antivirus are using a mutation of Aho-Corasic pattern matching algorithm. This algorithm uses a finite state pattern matching automaton [1]. The algorithm itself is a generalization of the Knuth-Morris-Pratt algorithm. Please look at *matcher.h* for data type definitions. The automaton is represented by the trie. Trie is a rooted tree with some specific properties [2]. Each node of the trie represents some state of the automaton. In the implementation, the node is defined as following:

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

[To be continued...]

5.3 Threads

Clam Antivirus uses POSIX threads. There is a *threadscan()* function, you can find in *treewalk.c*. Threads are created in the joinable state (PTHREAD_CREATE_JOINABLE, default). If the program is started in superuser mode, the realtime scheduling policy SCHED_FIFO is turned on. Each thread reads files list, as long as value of *stopper* is false. Threads are synchronized by *semaphore*, modification of list is protected by *mutex*.

5.4 Signals

There are no proper signal handling routines now. Only in compression mode are all signals blocked because of some filesystem operations. This will be fixed in the future, but it isn't very important now, so may wait some time.

5.5 Best editor ;)

The entire Clam Antivirus was written in the VIM (Vi IMproved) editor. You may get it from `http://www.vim.org`

6 Credits

In alphabetical order:

- AIX PDSLIB, University of California at Los Angeles `http://aixpdslib.seas.ucla.edu` - binary packages for AIX
- Kamil Andrusz `<wizz@mniam.net>` - OpenBSD support patch
- Jean-Edouard BABIN `<Jeb@jeb.com.fr>` - NetBSD support; made his NetBSD box available to me.
- Marc Baudoin `<babafou@babafou.eu.org>` - NetBSD testing
- Hilko Bengen `<bengen@vdst-ka.inka.de>` - support for Clam Antivirus in his AMaViS - "Next Generation"
- Eric I. Lopez Carreon `<elopezc@technitrade.com>` - Spanish "Sendmail + AMaViS + ClamAV Installation" how-to
- Magnus Ekdahl `<magnus@debian.org>` - Debian (`http://www.debian.org`) package maintainer; fixes and improvements.

- David Ford <david+cert@blue-labs.org> - gcc 3.x support fix
- Wieslaw Glod <wkg@x2.pl> - bug description: FreeBSD compile problem in 0.22.
- Matthew A. Grant <grantma@anathoth.gen.nz> - OpenAntiVirus Update script (*oav-update*)
- Michal Hajduczenia <michalis@mat.uni.torun.pl> - Clam title logo.
- Thomas W. Holt Jr. <twh@cohesive.net> - informations about ClamAV compiling on Solaris 2.6 and Cobalt MIPS boxes.
- Kurt Huwig <kurt@iku-netz.de> - smart suggestions.
- Dave Jones <dave@kalkbay.co.za> - bug description: problem in option parser.
- Kazuhiko <kazuhiko@fdiary.net> - Qmail-Scanner 0.12 support patch.
- Dr Andrzej Kurpiel <akurpiel@mat.uni.torun.pl> - choice of this project from my list.
- Dennis Leeuw <dleeuw@made-it.com> - *"Debian GNU/Linux Mail Server"* how-to, **corrections of this document**.
- Peter N Lewis <peter@stairways.com.au> - Mac OS X data type problem bugfix.
- Stefan Martig <sm@officeco.ch> - bug description: /proc/cpuinfo analysis problem on Linux/Alpha.
- Ken McKittrick <klmac@usadatanet.com> - intensive FreeBSD testing.
- Arkadiusz Miskiewicz <misiek@pld.org.pl> - Polish(ed) Linux Distribution (http://www.pld.org.pl) rpm package maintainer; fixes and ideas.
- NERvOus <nervous@nervous.it> - ElektraPro.com representative, he offered new (fast) site for ClamAV.
- Masaki Ogawa <proc@mac.com> - Mac OS X support.
- Martijn van Oosterhout <kleptog@svana.org> - code analysis and suggestions.
- OpenAntivirus.org Team - virus database.

- Sergei Pronin <sp@finndesign.fi> - bug description: access problems in superuser mode.
- Thomas Quinot <thomas@cuivre.fr.eu.org> - patch for non-default prefix and incoherent database location specification in defaults.h of clamscan and freshclam.
- Marcin T. Rzewucki <marcinr@mat.uni.torun.pl> - Fancy Headers.
- Dr Zbigniew Szewczak <zssz@mat.uni.torun.pl> - ideas, suggestions and spent time on discussing some aspects of ClamAV.
- Trashware trashware@gmx.net - TrashScan
- Troy Wollenslegel <troy@intranet.org> - bug description: handling inaccessible directories in archives.
- Andoni Zubimendi <andoni@lpsat.net> - fix for segmentation fault in 0.12 (NULL pointer dereference).

7 Author

Please be patient. This is free software, it will be much better with each release. If you have some questions, feel free to mail me.

Tomasz Kojm <zolw@konarski.edu.pl>

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-Corasic algorithm description