



Clam AntiVirus: User Manual

version 0.51

Tomasz Kojm

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 in your own software. Clam AV uses a virus database from OpenAntiVirus.org, we also help with signature generating.

1.1 Features

- GNU GPL license
- POSIX compliant, portable
- Secure
- Very fast
- Multi-threaded
- User friendly
- On-access scanning (Linux only)
- Detects over 7000 viruses, worms and trojans
- Supports compressed files and archives
- Built-in support for RAR (2.0), Zip, Gzip

1.2 Mailing lists

There are three mailing lists available:

announce@clamav.elektrapro.com - info about new versions (including debian package releases), moderated¹.

users@clamav.elektrapro.com - user questions

devel@clamav.elektrapro.com - developement

You can subscribe by sending an empty email to `listname-subscribe@clamav.elektrapro.com`, or via www at

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

<http://clamav.elektrapro.com/ml>

After subscribing you must reply to a special message sent at your address.

Mailing lists are archived at:

<http://archive.elektrapro.com/clamav.elektrapro.com/users/>

<http://archive.elektrapro.com/clamav.elektrapro.com/devel/>

2 Installation

2.1 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)
- Solaris 2.6/7/8 (Intel/SPARC)
- FreeBSD 4.5 (Intel/Alpha)
- OpenBSD 3.0/3.1 (Intel)
- NetBSD
- AIX 4.1/4.2/4.3/5.1 (RISC 6000)
- Mac OS X
- Cobalt MIPS boxes (RAQ1, RAQ2, QUBE2)

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

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 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>). The binary packages for AIX are available in AIX PDSLIB, UCLA <http://aixpdslib.seas.ucla.edu/packages/clamav.html>.

2.4 Installation

Please read the README file in the current version, because it probably contains some important release notes. If you are installing Clam AV for the first time, you have to add a new user and group to your system - *clamav*:²

```
# 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 or won't be using **clamscan** in superuser mode, you may omit this step with the option *--disable-clamav* passed to the *configure* script:

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

This disables testing for *clamav* user and group. **clamscan** 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.

After this, extract the archive, configure and compile:

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

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

²Cygwin note: If you don't have */etc/passwd*, you don't need the *clamav* user/group.

2.5 Configuration

If you are going to use the daemon, you need to configure it.

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

Now you know, where the configuration file is located. You will find more information about it in 3.1 section.

Clamuko requires the Dazuko software and is supported with Linux 2.2 and 2.4 only. You can obtain Dazuko from <http://dazuko.org> or clamav-x.yz/support/dazuko (version tested with clamd). Installation:

```
$ tar xzpfv dazuko-a.b.c.tar.gz
$ cd dazuko-a.b.c
$ make dazuko
or
$ make dazuko-smp (for smp kernels)
$ su -c "insmod dazuko.o"
```

You must create also the */dev/dazuko* device:

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

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

2.6 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.**³

³Please run *man clamscan*

2.7 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 3.5. 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 is it working correctly. If everything is OK, create a log file in /var/log 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 a 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 a new database daily at 8 am. You may need to setup the proxy support on your system. You should set the environment variable *\$http_proxy*, eg.

```
export http_proxy="my.proxy.server:8080"
```

There is also *-http-proxy* option available.

2.8 AMaViS - "Next Generation"

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

```
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
```

2.9 Qmail-Scanner

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

2.10 Support for AMaViS-perl

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

```
$ tar xzpf 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.

3 Usage

3.1 Clam daemon

clamd is a fully multi-threaded daemon, based on *libclamav*. It's able to work in one of the two modes, using:

- Unix (local) sockets
- TCP sockets

The daemon is configured by the *clamav.conf* file. You will find a description of all the options in the **clamav.conf(5)** manual. *clamd* recognizes the following commands:

- **PING**
Check server's state. It should reply with "PONG".
- **RELOAD**
Reload the databases.
- **QUIT**
Perform a clean exit.
- **SCAN file/directory** Scan a file or directory (recursively) with archive support. A full path is required.
- **RAWSCAN file/directory** Scan a file or directory (recursively) with archive support disabled. A full path is required.

Internal threads (except clamuko) are ignoring all external signals. The main thread handles *SIGTERM* and *SIGINT* signals and performs a proper exit when one of them is caught.

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* because of Dazuko implementation. Client (clamuko) - server (clamd) model is currently not supported by Dazuko. There are some benefits from current implementation - clamuko is sharing the database with *clamd*, and it's updated with the RELOAD command. **You must obey the following principles when using clamuko:**

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

You need to enable clamuko in *clamav.conf*. To protect directory /home, please use the option:

```
ClamukoIncludePath /home
```


To protect the whole system:

```
ClamukoIncludePath /  
ClamukoExcludePath /proc  
ClamukoExcludePath /tmpdir/of/mail/scanner
```

You can use clamuko to protect file access on Samba/Netatalk. NFS is not supported (Dazuko doesn't intercept NFS access calls). Another idea - you can build a database containing a signatures of the popular exploits, it will protect you against script-kiddies.

3.3 Archives and compressed files

Clam AntiVirus depends on LibClamAV. It has built-in support for the following formats:

- Zip
- Gzip
- RAR (2.0 only)

Archive files are detected by checking a magic strings.⁴ 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) won't be supported.

The daemon scans archives supported by libclamav only. Clamscan tries to scan an archive with built-in code, but when it fails it's able to switch to the external unpacker:

```
$ 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
```

⁴Just like the file(1) command.

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

clamscan supports many popular compressors - it uses external programs for each format. **If the scanner runs with superuser privileges unpackers are executed with *clamav* privileges, which makes the process far more secure.** It also makes sure, that *clamav* user has read access to all scanned compressed files. **You should have enabled recursive scanning with the *-r* option (*-recursive*), if you want to scan the whole content of the archive (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 the optional arguments is for example *-unzip=/path/to/unzip*.*

-unzip: You probably don't need this option, because Zip is supported by libclamav. But if libclamav will fail to unzip some file, it may be useful. clamscan was tested with *UnZip 5.41 of 16 April 2000, by Info-ZIP*.

-unrar: Tested with *UNRAR 3.00 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.

3.4 Output format

clamd uses uniformed 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.
```

It always closes the connection when first virus is found. In the case of archives, the output is exactly the same like with normal files:

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

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:

3.5 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 privileges, 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.

3.6 Signature Tool

sigtool automates signature creation. If you have an 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 random file and put the **test1** file 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: ClamAV-Test-Signature FOUND
```

```
----- SCAN SUMMARY -----
Known viruses: 2033
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". Run *sigtool* with the following parameters:

```
$ 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 at 12103, moving backward.
Detected at 11983, moving backward.
Detected at 11923, moving backward.
Not detected, increasing pos 11893 -> 11923
Detected at 11923, moving backward.
Not detected, increasing pos 11908 -> 11923
Detected at 11923, moving backward.
Not detected, increasing pos 11915 -> 11923
Detected at 11923, moving backward.
Detected at 11919, moving backward.
Detected at 11917, moving backward.
Detected at 11916, moving backward.
Starting precise loop
*** Found signature end at 11916
```

The scanner was executed 46 times.
Signature length is 50, so length of hex string should be 100
Saving signature in testfile.sig file.

4 LibClamAV

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

4.1 API

Each program using libclamav must include *clamav.h* header file:

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

The first step is an engine initialization. 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 one database per time, *cl_loaddbdir()* loads all *.db* and *.db2* files from the directory *dirname*. *cl_retdbdir()* returns hardcoded database directory path. The database will be saved under *root* and the number of the loaded signatures will be **added** to *virnum*. Pointer to the tree structure (trie, see 6.2) must initially point to the NULL. If you don't want to save the number of signatures loaded pass the NULL as the 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 elegant way to print libclamav's error codes:

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

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

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

When database is loaded, you must create the proper trie with:

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

In our example:

```
cl_buildtrie(root);
```

OK, now you can scan a buffer, descriptor or 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 the name in the trie structure, thus it can't be released directly. *cl_scandesc()* and *cl_scanfile()* can increase *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) and **CL_RAW** (disables archive scanning). The 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;

/* maximal number of files in archive */
limits.maxfiles = 100
/* maximal archived file size == 10 Mb */
limits.maxfilesize = 10 * 1048576;
/* maximal recursion level */
limits.maxreclevel = 8;
```

```
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_perror(ret));
}
```

When you don't need to scan more files, the trie should be released with:

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

You will find some examples in clamav sources. Each program using libclamav must be linked against it:

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

Enjoy !

5 Problem solving

5.1 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 than 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.
- 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 system.
- 71:** Can't allocate memory. Look above.

6 Technicals

6.1 Security

Clam AntiVirus cares about security. Dangerous operations (such as extracting, temporary file creation, *unlink()* operations) are executed with *clamav* privileges. **But there are no programs without bugs.** This is a young project and everything

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

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.

6.2 Scan engine

New versions of 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...]

6.3 Threads

Clam AntiVirus uses POSIX threads.

7 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
- Nicholas Chua <nicholas@ncmbx.net> - big database updates
- Alejandro Dubrovsky <s328940@student.uq.edu.au> - patch for including and excluding multiple patterns.
- 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
- Piotr Gackiewicz <gacek@intertele.pl> - bug report: clamd THREXIT bug
- Wieslaw Glod <wkg@x2.pl> - bug report: 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. <tw@cohesive.net> - informations about ClamAV compiling on Solaris 2.6 and Cobalt MIPS boxes.
- Nigel Horne <njh@smsltd.demon.co.uk> - mbox support in clamscan, support/mboxscan, many fixes/improvements.
- Kurt Huwig <kurt@iku-netz.de> - smart suggestions, ScannerDaemon (OpenAntiVirus) author.
- Dave Jones <dave@kalkbay.co.za> - bug report: problem in option parser.
- Kazuhiko <kazuhiko@fdiary.net> - Qmail-Scanner 0.12 support patch.
- Henk Kuipers <henk@opensourcesolutions.nl> - bug report: 0.50 compile problem
- 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**.
- Free Oscar <freeoscar@wp.pl> - hex2str() enhancement
- Peter N Lewis <peter@stairways.com.au> - Mac OS X data type problem bugfix.
- Mike Loewen <mloewen@sturgeon.cac.psu.edu> - bug report: clamscan 0.24 compile error on Solaris 8.
- Stefan Martig <sm@officeco.ch> - bug report: /proc/cpuinfo problem analysis 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.
- Doug Monroe <doug@planetconnect.com> - Qmail-Scanner problem analysis
- NERvOus <nervous@nervous.it> - ElektraPro.com representative, he offered new (fast) site for ClamAV.
- Masaki Ogawa <proc@mac.com> - Mac OS X support, Japanese documentation.
- Martijn van Oosterhout <kleptog@svana.org> - code analysis and suggestions.
- OpenAntivirus.org Team - virus database.
- Sergei Pronin <sp@finndesign.fi> - bug report: 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.
- David Sanchez <dsanchez@veloxia.com> - bug report: thread deadlocking in a critical error situation.

- 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 report: handling inaccessible directories in archives.
- Andoni Zubimendi <andoni@lpsat.net> - fix for segmentation fault in 0.12 (NULL pointer dereference).

8 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