

Creating signatures for ClamAV

1 Introduction

CVD (ClamAV Virus Database) is a digitally signed container that includes signature databases in various text formats. The header of the container is a 512 bytes long string with colon separated fields:

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

`sigtool --info` displays detailed information about a given CVD file:

```
zolw@localhost:/usr/local/share/clamav$ sigtool -i main.cvd
File: main.cvd
Build time: 09 Dec 2007 15:50 +0000
Version: 45
Signatures: 169676
Functionality level: 21
Builder: sven
MD5: b35429d8d5d60368eea9630062f7c75a
Digital signature: dxsus0/HWP3/GAA7VuZpxYwVsE9b+tCk+tPN6OyjVF/U8
JVh4vYmW8mZ62ZHYMlM903TMZFG5hZIXc jQB3SX0TapdF1SFNzoWjsyH53eXvMDY
eaPVNe2ccXLfEegoda4xU2TezbGfbSEGoU1qolyQYLX674sNA2Ni6l6/CEKYYh
Verification OK.
```

The ClamAV project distributes two CVD files: *main.cvd* and *daily.cvd*.

2 Signature formats

2.1 MD5

The easiest way to create signatures for ClamAV is to use MD5 checksums, however this method can be only used against static malware. To create a signature for `test.exe` use the `--md5` option of `sigtool`:

```
zolw@localhost:/tmp/test$ sigtool --md5 test.exe > test.hdb
zolw@localhost:/tmp/test$ cat test.hdb
48c4533230e1aelc118c741c0db19dfb:17387:test.exe
```

That's it! The signature is ready to use:

```
zolw@localhost:/tmp/test$ clamscan -d test.hdb test.exe
test.exe: test.exe FOUND
```

```
----- SCAN SUMMARY -----
Known viruses: 1
Scanned directories: 0
Engine version: 0.92.1
Scanned files: 1
Infected files: 1
Data scanned: 0.02 MB
Time: 0.024 sec (0 m 0 s)
```

You can change the name (by default sigtool uses the name of the file) and place it inside a *.hdb file. A single database file can include any number of signatures. To get them automatically loaded each time clamscan/clamd starts just copy the database file(s) into the local virus database directory (eg. /usr/local/share/clamav).

2.2 MD5, PE section based

You can create a MD5 signature for a specific section in a PE file. Such signatures shall be stored inside .mdb files in the following format:

```
PESectionSize:MD5:MalwareName
```

The easiest way to generate MD5 based section signatures is to extract target PE sections into separate files and then run sigtool with the option --mdb

2.3 Hexadecimal signatures

ClamAV stores all signatures in a hexadecimal format. By a hex-signature here we mean a fragment of a malware's body converted into a hexadecimal string which can be additionally extended with various wildcards.

2.3.1 Hexadecimal format

You can use `sigtool --hex-dump` to convert any data into a hex-string:

```
zolw@localhost:/tmp/test$ sigtool --hex-dump
How do I look in hex?
486f7720646f2049206c6f666b20696e206865783f0a
```

2.3.2 Wildcards

ClamAV supports the following extensions inside hex signatures:

- `??`
Match any byte.
- `a?`
Match a high nibble (the four high bits). **IMPORTANT NOTE:** The nibble matching is only available in libclamav with the functionality level 17 and higher therefore please only use it with `.ndb` signatures followed by `":17"` (MinEngineFunctionalityLevel, see 2.3.4).
- `?a`
Match a low nibble (the four low bits).
- `*`
Match any number of bytes.
- `{n}`
Match n bytes.
- `{-n}`
Match n or less bytes.
- `{n-}`
Match n or more bytes.
- `(aa|bb|cc|..)`
Match aa or bb or cc..
- `HEXSIG[x-y]aa` or `aa[x-y]HEXSIG`
Match aa anchored to a hex-signature, see https://www.clamav.net/bugzilla/show_bug.cgi?id=776 for a discussion and examples.

The range signatures * and {} virtually separate a hex-signature into two parts, eg. aabbcc*bbaacc is treated as two sub-signatures aabbcc and bbaacc with any number of bytes between them. It's a requirement that each sub-signature includes a block of two static characters somewhere in its body.

2.3.3 Basic signature format

The simplest (and now deprecated) signature format is:

```
MalwareName=HexSignature
```

ClamAV will scan the entire file looking for HexSignature. All signatures of this type must be placed inside *.db files.

2.3.4 Extended signature format

The extended signature format allows for specification of additional information such as a target file type, virus offset or engine version, making the detection more reliable. The format is:

```
MalwareName:TargetType:Offset:HexSignature[:MinEngineFunctionalityLevel:[Max]]
```

where TargetType is one of the following numbers specifying the type of the target file:

- 0 = any file
- 1 = Portable Executable
- 2 = OLE2 component (e.g. a VBA script)
- 3 = HTML (normalised)
- 4 = Mail file
- 5 = Graphics
- 6 = ELF
- 7 = ASCII text file (normalised)

And Offset is an asterisk or a decimal number n possibly combined with a special modifier:

- * = any
- n = absolute offset
- EOF-n = end of file minus n bytes

Signatures for PE and ELF files additionally support:

- EP+n = entry point plus n bytes (EP+0 for EP)
- EP-n = entry point minus n bytes
- Sx+n = start of section x's (counted from 0) data plus n bytes
- Sx-n = start of section x's data minus n bytes
- SL+n = start of last section plus n bytes
- SL-n = start of last section minus n bytes

All the above offsets except * can be turned into **floating offsets** and represented as `Offset,MaxShift` where `MaxShift` is an unsigned integer. A floating offset will match every offset between `Offset` and `Offset+MaxShift`, eg. `10,5` will match all offsets from 10 to 15 and `EP+n,y` will match all offsets from `EP+n` to `EP+n+y`. Versions of ClamAV older than 0.91 will silently ignore the `MaxShift` extension and only use `Offset`.

All signatures in the extended format must be placed inside *.ndb files.

2.4 Signatures based on archive metadata

Signatures based on metadata inside archive files can provide an effective protection against malware that spreads via encrypted zip or rar archives. The format of a metadata signature is:

```
virname:encrypted:filename:normal size:csize:crc32:cmethod:fileno:max depth
```

where the corresponding fields are:

- Virus name
- Encryption flag (1 – encrypted, 0 – not encrypted)
- File name (* to ignore)
- Normal (uncompressed) size (* to ignore)

- Compressed size (* to ignore)
- CRC32 (* to ignore)
- Compression method (* to ignore)
- File position in archive (* to ignore)
- Maximum number of nested archives (* to ignore)

The database file should have the extension of .zmd or .rmd for zip or rar metadata respectively.

2.5 Whitelist databases

To whitelist a specific file use the MD5 signature format and place it inside a database file with the extension of .fp.

To whitelist a specific signature inside main.cvd add the following entry into daily.ign or a local file local.ign:

```
db_name:line_number:signature_name
```

2.6 Signature names

ClamAV uses the following prefixes for signature names:

- *Worm* for Internet worms
- *Trojan* for backdoor programs
- *Adware* for adware
- *Flooder* for flooders
- *HTML* for HTML files
- *Email* for email messages
- *IRC* for IRC trojans
- *JS* for Java Script malware
- *PHP* for PHP malware
- *ASP* for ASP malware

- *VBS* for VBS malware
- *BAT* for BAT malware
- *W97M*, *W2000M* for Word macro viruses
- *X97M*, *X2000M* for Excel macro viruses
- *O97M*, *O2000M* for generic Office macro viruses
- *DoS* for Denial of Service attack software
- *DOS* for old DOS malware
- *Exploit* for popular exploits
- *VirTool* for virus construction kits
- *Dialer* for dialers
- *Joke* for hoaxes

Important rules of the naming convention:

- always use a *-zippwd* suffix in the malware name for signatures of type *zmd*,
- always use a *-rarpwd* suffix in the malware name for signatures of type *rmd*,
- only use alphanumeric characters, dash (-), dot (.), underscores (_) in malware names, never use space, apostrophe or quote mark.

3 Special files

3.1 HTML

ClamAV contains a special HTML normalisation code which helps to detect HTML exploits. Running `sigtool --html-normalise` on a HTML file should generate the following files:

- *nocomment.html* - the file is normalised, lower-case, with all comments and superflous white space removed
- *notags.html* - as above but with all HTML tags removed

The code automatically decodes `JScript.encode` parts and char ref's (e.g. `f`). You need to create a signature against one of the created files. To eliminate potential false positive alerts the target type should be set to 3.

3.2 Text files

Similarly to HTML all ASCII text files get normalised (converted to lower-case, all superflous white space and control characters removed, etc.) before scanning. Use `clamscan --leave-temps` to obtain a normalised file then create a signature with the target type 7.

3.3 Compressed Portable Executable files

If the file is compressed with UPX, FSG, Petite or other PE packer supported by libclamav, run `clamscan` with `--debug --leave-temps`. Example output for a FSG compressed file:

```
LibClamAV debug: UPX/FSG/MEW: empty section found - assuming compression
LibClamAV debug: FSG: found old EP @119e0
LibClamAV debug: FSG: Unpacked and rebuilt executable saved in
/tmp/clamav-f592b20f9329ac1c91f0e12137bcce6c
```

Next create a type 1 signature for `/tmp/clamav-f592b20f9329ac1c91f0e12137bcce6c`