Creating signatures for ClamAV

1 Introduction

CVD (ClamAV Virus Database) is a digitally signed tarball file that contains one or more databases. The header 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 CVD file:

```
zolw@localhost:/usr/local/share/clamav$ sigtool -i main.cvd
Build time: 09 Jun 2006 22-19 +0200
Version: 39
# of signatures: 58116
Functionality level: 8
Builder: tkojm
MD5: a9a400e70dcbfe2c9e11d78416e1c0cc
Digital signature: 0s12V8OxLWO95fNNv+kTxj7CEWBW/1TKOGC7G4RelhogruBYw8dJeIX2+yhxex/Xs
Verification OK.
```

There are two CVD databases in ClamAV: *main.cvd* and *daily.cvd* for daily updates.

2 Signature format

2.1 MD5

There's an easy way to create signatures for static malware using MD5 checksums. 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
48c4533230e1ae1c118c741c0db19dfb: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: 0.88.2
Scanned files: 1
Infected files: 1
Data scanned: 0.02 MB
Time: 0.024 sec (0 m 0 s)
```

You can edit it to change the name (by default sigtool uses the file name). Remember that all MD5 signatures must be placed inside `*.hdb` files and you can include any number of signatures inside a single file. To get them automatically loaded every time clamscan/clamd starts just copy them to the local virus database directory.

2.2 MD5, PE section based

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

```
PESectionSize:MD5:MalwareName
```

2.3 Hexadecimal signatures

ClamAV keeps viral fragments in hexadecimal format. If you don't know how to get a proper signature please try the MD5 method or submit your sample at `http://www.clamav.net/sendvirus`

2.3.1 Hexadecimal format

You can use `sigtool --hex-dump` to convert arbitrary data into hexadecimal format:

```
zolw@localhost:/tmp/test$ sigtool --hex-dump
How do I look in hex?
486f7720646f2049206c6f6f6b20696e206865783f0a
```

2.3.2 Wildcards

ClamAV supports the following extensions inside hex signatures:

- `??`
  Match any byte.
- `a?`
  Match high nibble (high four bits). **IMPORTANT NOTE:** Nibble matching is only available in libclamav with the functionality level 15 therefore please only use it with .ndb signatures, each followed by ":15" (MinEngineFunctionalityLevel, see 2.3.4).
- `?a`
  Match low nibble (low four bits).
- `*`
  Match any number of bytes.
- `{n}`
  Match n bytes.
- `{-n}`
  Match n or less bytes.
- `{n-}`
  Match n or more bytes.
- `(a|b)`
  Match a or b (you can use more alternate characters).

2.3.3 Basic signature format

The simplest signatures are of the format:

```
MalwareName=HexSignature
```

ClamAV will analyse a whole content of a file trying to match it. All signatures of this type must be placed in `*.db` files.

2.3.4 Extended signature format

Extended signature format allows on including additional information about target file type, virus offset and required engine version. The format is:

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

where `TargetType` is one of the following decimal numbers describing the target file type:

- 0 = any file
- 1 = Portable Executable
- 2 = OLE2 component (e.g. VBA script)
- 3 = HTML (normalised)
- 4 = Mail file
- 5 = Graphics (to help catching exploits in JPEG files)
- 6 = ELF

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

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

Signatures for Portable Executables files (target = 1) also support:

- `EP+n` = entry point plus n bytes (`EP+0` if you want to anchor to `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

In order to detect some malware which spreads inside of Zip or RAR archives (especially encrypted ones) you can try to create a signature describing a malicious archived file. The general format is:

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

- 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 should have the extension `.zmd` or `.rmd` for Zip or RAR archive respectively.

2.5 Whitelist database

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

2.6 Signature names

ClamAV uses the following prefixes for particular malware:

- *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 general 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 required to detect HTML exploits. Running `sigtool --html-normalise` on a HTML file should create the following files:

- comment.html - the whole file normalised
- nocomment.html - the file normalised, with all comments removed
- script.html - the parts of the file in `<script>` tags (lowercased)

The code automatically decodes JScript.encode parts and char ref's (e.g. `&#102;`). You need to create a signature against one of the created files. To eliminate potential false positive alerts you should use extended signature format with target type of 3.

3.2 Compressed Portable Executable files

If the file is compressed with UPX, FSG, Petite or other executable packer (supported by libclamav) run `clamscan` with `--debug --leave-temps`. Example output on FSG compressed file:

```
LibClamAV debug: UPX/FSG: empty section found - assuming compression
LibClamAV debug: FSG: found old EP @1554
LibClamAV debug: FSG: Successfully decompressed
LibClamAV debug: UPX/FSG: Decompressed data saved in /tmp/clamav-4eba73ff4050a26
```

and then create a signature for `/tmp/clamav-4eba73ff4050a26`