Creating signatures for ClamAV (for beginners)

v. 20040416: update
v. 20040310: general update
v. 20040310: general update
v. 20031101: updated for CVD
v. 20030506: first version

1 Introduction

ClamAV 0.65 introduces a new container file format called CVD (ClamAV Virus Database). This is a digitally signed tarball file that contains one or more databases. You can find some useful information in the ASCII header of each CVD 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 daily updates. In older ClamAV versions there are two old-style databases: *viruses.db* and *viruses.db2*. The first one contains static signatures while the further contains signatures with simple regular expressions (used to match mutating viruses). The goal of a signature creation process is to get a small part of an infected file which identifies a virus. It must be *unique* to avoid false positive alarms.

2 Getting a hexadecimal string

The best way is to create the signature manually however sometimes you can automate the process:

2.1 sigtool

Sigtool is only partially useful because it only detects the last part of a real signature. It will fail for multipart signatures (mutating viruses). Sigtool can use third party software to create a signature from an infected file. The following example uses ClamAV: [1]

```
zolw@Wierszokleta:/tmp/bug$ sigtool -c "clamscan --stdout" -s FOUND -f bug.exe
Detected, decreasing end 50688 -> 40550
Detected, decreasing end 40550 -> 30412
Detected, decreasing end 30412 -> 20274
Detected, decreasing end 20274 -> 10136
Not detected at 0, moving forward.
Not detected at 5069, moving forward.
...
Detected, decreasing end 6353 -> 6352
Not detected at 6352, moving forward.
Increasing end 6352 -> 6353
 *** Signature end found at 6353
Not detected, moving backward 6303 -> 6253
Detected at 6253, moving forward.
Not detected, moving backward 6278 -> 6253
Detected at 6253, moving forward.
Detected at 6266, moving forward.
Detected at 6273, moving forward.
Detected at 6277, moving forward.
Not detected, moving backward 6279 -> 6277
Detected at 6277, moving forward.
Not detected, moving backward 6278 -> 6277
Detected at 6277, moving forward.
Not detected, moving backward 6278 -> 6277
Detected at 6277, moving forward.
Moving forward 6277 -> 6278
 *** Signature start found at 6278

The scanner was executed 39 times.
The signature length is 75 (150 hex)
Saving signature in bugbear.exe.sig file.
Saving binary signature in bugbear.exe.bsig file.
```

See below how to finish the signature.

[1] Make sure your commercial scanner's license does not disallow sigtool usage !

2.2 by hand...

This is a recommended (but arduous) method. In most cases it's possible to create a signature for a virus/worms without complex analysis. You can try to examine the file with your favorite editor (with support for a hex mode) and eventually use *strings* to locate some "fingerprints" of the virus. Let's look at the BugBear example: the worm is compressed so you shouldn't expect to find a simple plain text in it. strings will return many lines of printable strings from the file:

```
zolw@Wierszokleta:/tmp/bug$ strings bugbear.exe|more
!This program cannot be run in DOS mode.
Rich5
.rsrc
LHVW3
S6_u
@=$r
h~j9
Wr*w-

.
.
.

P/1.1$H
: Apache
3.26 (U
e:''
XTzPOST
<author
IRr+l
.
.
.
```

Yep, the second block is what we were searching for (you can read about BugBear in Internet and it should make clear). My favorite editor is ViM - you can view the file in hex mode by filtering it with *:%!xxd*

```
0009b00: 17fd 2fd5 4db1 7369 6e67 2064 6174 61e3  ../.M.sing data.
0009b10: b07c b2ff 6d61 6765 2f67 6966 0b6a 7065  .|..mage/gif.jpe
0009b20: 6761 f16f a82f 6e6c 6963 61c5 2f6f 6374  ga.o./nlica./oct
0009b30: 6574 2d73 dbdb a36e 3365 612f 0d78 742f  et-s...n3ea/.xt/
```

```
0009b40: 1e61 6ba8 076b 470b 6874 6d30 7238 705b  .ak..kG.htm0r8p[
0009b50: c09b 1369 0062 7f68 0f6b edd6 1673 7a1f  ...i.b.h.k...sz.
0009b60: 1e00 634b 030f b9b3 2f98 2607 0065 0007  ..cK..../.&..e..
0009b70: 3754 6baf 067d 231a 7676 7864 b8a1 daf6  7Tk..}#.vvxd....
0009b80: 0073 7973 0f6f 2372 626d 708d 3d7f 0bb3  .sys.o#rbmp.=...
0009b90: 2c20 2530 3264 640a 3aba 35d5 9304 2047  , %02dd.:.5... G
0009ba0: 4d87 3f00 0048 2853 bddf 1b50 2f31 2e31  M.?..H(S...P/1.1
0009bb0: 2448 fbbb ffe6 6302 3a20 4170 6163 6865  $H....c.: Apache
0009bc0: 1933 2e32 3620 2855 a251 b1db 7678 291d  .3.26 (U.Q..vx).
0009bd0: 44a5 653a 2760 a56e adb0 0a02 2d74 e711  D.e:'‘.n....-t..
0009be0: be6d 35f7 5075 62fd 0b58 547a 504f 5354  .m5.Pub..XTzPOST
0009bf0: f6b7 49d5 12ab 3c61 7574 686f 72da 866e  ..I...<author..n
0009c00: 5fa1 1fbf 460a 6269 2ca5 5a08 0c0a a374  _...F.bi,.Z....t
0009c10: 7a3d bb75 df75 6e18 4952 722b 6c20 8420  z=.u.un.IRr+l .
0009c20: 45f7 3658 6bd2 2923 49d3 6c65 6d71 85a0  E.6Xk.)#I.lemq..
0009c30: 733b 4242 61c1 8977 c7a1 d09d 5413 2063  s;BBa..w....T. c
0009c40: 765c 5fb4 ea63 5b83 651f 5265 7175 5aa1  v\_..c[.e.RequZ.
0009c50: 6dad 10c3 490d 5025 a7db cedc 6e6e 3d6c  m...I.P%....nn=l
0009c60: 794f 1354 4e70 5e2e fc62 6d61 9513 636d  yO.TNpˆ..bma..cm
```

You can now read the hex code on the left side but before that you must remember a few important rules for ClamAV signatures:

- it should contain some ”binary” data to avoid false positive alarms with plain (ASCII) text files
- **it shouldn’t start with 00** because there’s a problem in ClamAV version ≤ 0.54 which will cause a dramatic performance loss. There’s a one such a signature in viruses.db2 (W32.Magistr.B) but it has to be.
- it should be long enough to avoid false positives but must only contain a viral code
- the recommended size of a hex signature is 40 up to 300 characters

OK, you can now read the signature directly from the left side, eg: (this one contains the ”Apache” string)

```
6302 3a20 4170 6163 6865 1933 2e32 3620 2855 a251 b1db 7678 291d
44a5 653a 2760 a56e adb0 0a02 2d74
```

what gives:

```
63023a204170616368651933 2e3236202855a251b1db7678291d44a5653a2760a56eadb00a022d74
```

If you don't want to read the signature from a hex editor, you can "cut out" (in a binary mode !) the viral part of the file and convert it into a hex string with:

```
cat viruspart | sigtool --hex-dump > virus.sig
```

3 Building the final signature

If you have the hex string the last thing is to add the virus name. Because ClamAV's database was build on OAV basis, we use *(Clam)* marker in every signature created by our team. To finish your signature just add the **VirusName (Clam)=** string:

```
Worm.BugBear.A (Clam)=63023a20417061636865193 32e3236202855a251b1db7678291d
44a5653a2760a56eadb00a022d74
```

Some important rules:

- remember about the (Clam) marker (it's automatically removed by the parser)
- use the most popular name of the virus/worm
- don't use white characters or slashes in virus names
- prefixes for particular malware
  - *Worm* for worms
  - *Trojan* for backdoor programs
  - *JS* for Java Script malware
  - *VBS* for VBS malware
  - *W97M*, *W2000M* for Word macros
  - *X97M*, *X2000M* for Excel macros
  - *DoS* for Denial of Service attack software
  - *VirTool* for virus construction kits
  - *Dialer* for dialers
  - *Joke* for hoaxes

4 Creating a local virus database

You can easily create your own database. Just put all signatures to some file with the *db* extension (eg. local.db) and install it in the clamav database directory. That's it !

5 CVD building - ClamAV maintainers only

Run freshclam and eventually check `www.clamav.net->Database` you have the latest databases installed. Go to some **empty** temporary directory and execute the following command:

```
sigtool --unpack-current daily.cvd
```

This will unpack the current *daily.cvd* database. Now you only need to update the internal database, eg:

```
cat virus.sig >> viruses.db[2]
```

And build the final CVD:

```
sigtool --build daily.cvd --server SIGNING_SERVER
```

where SIGNING_SERVER is one of the ClamAV Signing Servers you have access to. This command will automatically generate the final CVD: it will increment the version number (by one), count signatures, etc.:

```
LibClamAV debug: Loading databases from .
LibClamAV debug: Loading ./viruses.db2
LibClamAV debug: Initializing trie.
Database properly parsed.
Signatures: 90
COPYING
tar: viruses.db: Cannot stat: No such file or directory
viruses.db2
tar: Notes: Cannot stat: No such file or directory
tar: Error exit delayed from previous errors
Builder id: tkojm
Password:
Signature received (length = 171).
Database daily.cvd created.
```

Don't worry about potential *tar* errors. Now you can verify the new database with:

```
zolw@Wierszokleta:/tmp/db$ sigtool -i daily.cvd
Build time: Nov-01 02-39 CET 2003
Version: 9
# of signatures: 90
```

```
Functionality level: 1
Builder: tkojm
MD5: 4c6713fb002c6eb2ecbb8b04276a66fa
Digital signature: 30rYGWKFPpu5YZgiczIUrNvn5wioITl...
Verification OK.
```

Now you must update the main rsync server:

```
rsync -tcz --stats --progress -e ssh daily.cvd clamupload@rsync1.clamav.net:public_html/
ssh rsync1.clamav.net -i ~/.ssh/id_rsa -l clamavdb sleep 1
```

Please consult [1] for more information. After an update please send a summary to `clamav-virusdb@lists.sf.net`. Thanks !

References

[1] Luca Gibelli: *Mirroring the Virus Database*
`http://www.clamav.net/doc/mirrors`