Phishing signatures creation HOWTO

Török Edwin

September 16, 2006

1 Database file format

The database file format is common for the whitelist (.wdb), and domainlist (.pdb), and it consists of (multiple) lines of form:

Flags RealURL DisplayedURL
or:
H RealURL

- Where FLAGS is:
  - an (optional) character :
    R regex, has to match entire url, see section
    H has to match the host part of REALURL only (a simple pattern, i.e. it is matched literally)
    no character matches the entire url, but as a simple pattern (non-regex)
  - followed by an (optional) 3-digit hexadecimal number representing flags that should be filtered.
    * flag filtering only makes sense in .pdb files, (however clamav won't complain if you put flags in .wdb files, it just won't use them)
    * for details on how to construct a flag number see section 1.4 on page 4
- REALURL is the URL the user is sent to
- DISPLAYEDURL is the URL description displayed to the user, that is where it is *claimed* they are sent, the most obvious example is that of an html anchor (<a>tag): its href attribute is the REALURL, and its contents is the DISPLAYEDURL.
- see section 1.1.3 on the next page for more details on what REALURL/DISPLAYEDURL is

Note: The spaces are mandatory, and empty lines are skipped.

If any of the lines of daily.wdb/daily.pdb don't conform to the above file format, the loading of the file shall fail, and whitelist/domainlist feature will be disabled. If the loading of the whitelist fails, the phishing checks will be disabled entirely.

Therefore it is important to test the daily.wdb/daily.pdb before packing it into daily.cvd!

1.1 How matching works

1.1.1 RealURL, displayedURL concatenation

The phishing detection module processes pairs of realURL/displayedURL, and the matching against daily.wdb/daily.pdb is done as follows: the realURL is concatenated with a space, and with the displayedURL, then that *line* is matched against the lines in daily.wdb/daily.pdb

So if you have a line like

www.google.ro www.google.com

and a href like: *www.google.com,* then it will match, but: *www.google.com* will not match.

If you use the **H** flag, then a line like:

H paypal.com

Will match paypal.com.

1.1.2 What happens when a match is found

In the case of the whitelist, a match means that the realURL/displayedURL combination is considered CLEAN, and no further checks are performed on it.

In the case of the domainlist, a match means that the realURL/displayedURL is going to be checked for phishing attempts. This is only done if you don't run clamav with the *alldomains* option (since then all urls are checked). Furthermore you can restrict what checks are to be performed by specifying the 3-digit hexnumber.

1.1.3 Extraction of REALURL, DISPLAYEDURL from HTML tags

The html parser extracts pairs of REALURL/DISPLAYEDURL based on the following rules:

a (anchor) the *href* is the REALURL, its *contents* is the DISPLAYEDURL

contents is the tag-stripped contents of the <a> tags, so for example <b> tags are stripped (but not their contents)

nesting another <a> tag withing an <a> tag (besides being invalid html) is treated as a <a..

form the *action* attribute is the REALURL, and a nested <a> tag is the DISPLAYEDURL

img/area if nested within an <*a*> tag, the REALURL is the *href* of the a tag, and the *src/dynsrc/area* is the DISPLAYEDURL of the img

if nested withing a *form* tag, then the action attribute of the *form* tag is the REALURL

iframe if nested withing an <*a*> tag the *src* attribute is the displayedURL, and the *href* of its parent *a* tag is the REALURL

if nested withing a *form* tag, then the action attribute of the *form* tag is the REALURL

1.2 Simple patterns

Simple patterns are matched literally, i.e. if you say: www.google.com, it is going to match www.google.com, and only that. The . character has no special meaning (see the section on regexes 1.3 for how the . character behaves there)

1.3 Regular expressions

POSIX regular expressions are supported, and you can consider that internally it is wrapped by ^, and *$*. In other words, this means that the regular expression has to match the entire concatenated (see section 1.1.1 on the previous page for details on concatenation) url.

It is recomended that you read section 2 on the following page to learn how to write regular expressions, and then come back and read this for hints.

Be advised that clamav contains an internal, very basic regex matcher to reduce the load on the regex matching core. Thus it is recomended that you avoid using regex syntax not supported by it at the very beginning of regexes (at least the first few characters).

Currently the clamav regex matcher supports:

- . (dot) character
- \ (escaping special characters)
- | (pipe) alternatives
- [] (character classes)
- () (paranthesis for grouping, but no group extraction is performed)
- other non-special characters

Thus the following are not supported:

- + repetition
- * repetition
- {} repetition
- backreferences
- lookaround
- other "advanced" features not listed in the supported list ;)

This however shouldn't discourage you from using the "not directly supported features ", because if the internal engine encounters unsupported syntax, it passes it on to the POSIX regex core (beginning from the first unsupported token, everything before that is still processed by the internal matcher). An example might make this more clear:

www\.google\.(com|ro|it) www\.([a-zA-Z])+\.google\.com

Everything till *([a-zA-Z])+* is processed internally, that paranthesis (and everything beyond) is processed by the posix core.

1.4 Flags

Flags are a binary OR of the following numbers:

HOST_SUFFICIENT 1

DOMAIN_SUFFICIENT 2

DO_REVERSE_LOOKUP 4

CHECK_REDIR 8

CHECK_SSL 16

CHECK_CLOAKING 32

CLEANUP_URL 64

CHECK_DOMAIN_REVERSE 128

CHECK_IMG_URL 256

DOMAINLIST_REQUIRED 512

The names of the constants are self-explanatory.

These constants are defined in libclamav/phishcheck.h, you can check there for the latest flags.

There is a default set of flags that are enabled, these are currently: (CLEANUP_URL|DOMAIN_SUFFICIENT|CHE ssl checking is performed only for a tags currently.

You must decide for each line in the domainlist if you want to filter any flags (that is you don't want certain checks to be done), and then calculate the binary OR of those constants, and then convert it into a 3-digit hexnumber. For example you devide that domain_sufficient shouldn't be used for ebay.com, and you don't want to check images either, so you come up with this flag number: $2|256 \Rightarrow 258(decimal) \Rightarrow 102(hexadecimal)$

So you add this line to daily.wdb:

R102 www.ebay.com .+

2 Introduction to regular expressions

Recomended reading:

- http://www.regular-expressions.info/quickstart.html
- http://www.regular-expressions.info/tutorial.html
- regex(7) man-page: http://www.tin.org/bin/man.cgi?section=7&topic=regex

2.1 Special characters

[the opening square bracket - it marks the beginning of a character class, see section 2.2

**** the backslash - escapes special characters, see section 2.3

^ the caret - matches the beginning of a line (not needed in clamav regexes, this is implied)

$ the dollar sign - matches the end of a line (not needed in clamav regexes, this is implied)

. the period or dot - matches *any* character

| the vertical bar or pipe symbol - matches either of the token on its left and right side, see section 2.4

? the question mark - matches optionally the left-side token, see section 2.5

***** the asterisk or star - matches 0 or more occurences of the left-side token, see section 2.5

+ the plus sign - matches 1 or more occurences of the left-side token, see section 2.5

(the opening round bracket - marks beginning of a group, see section 2.6

) the closing round bracket - marks end of a group, see section 2.6

2.2 Character classes

2.3 Escaping

Escaping has two purposes:

- it allows you to actually match the special characters themselves, for example to match the literal +, you would write \+
- it also allows you to match non-printable characters, such as the tab (\t), newline (\n), ..

However since non-printable characters are not valid inside an url, you won't have a reason to use them.

2.4 Alternation

2.5 Optional matching, and repetition

2.6 Groups

Groups are usually used together with repetition, or alternation. For example: *(com|it)+* means: match 1 or more repetitions of *com* or *it,* that is it matches: com, it, comcom, comcomcom, comit, itit, ititcom,... you get the idea.

Groups can also be used to extract substring, but this is not supported by the clam engine, and not needed either in this case.

3 Hints and recomandations

4 Examples