# Clamav funcation call flow

bm scan with normal signature(just numbers)

***by eqmcc***

[Clamav funcation call flow 1](#_Toc345424717)

[The call flow 1](#_Toc345424718)

[Data structures 1](#_Toc345424719)

[Test case 5](#_Toc345424720)

[Engine initialiazation 7](#_Toc345424721)

[Load signatures 7](#_Toc345424722)

[Scan 10](#_Toc345424723)

## The call flow

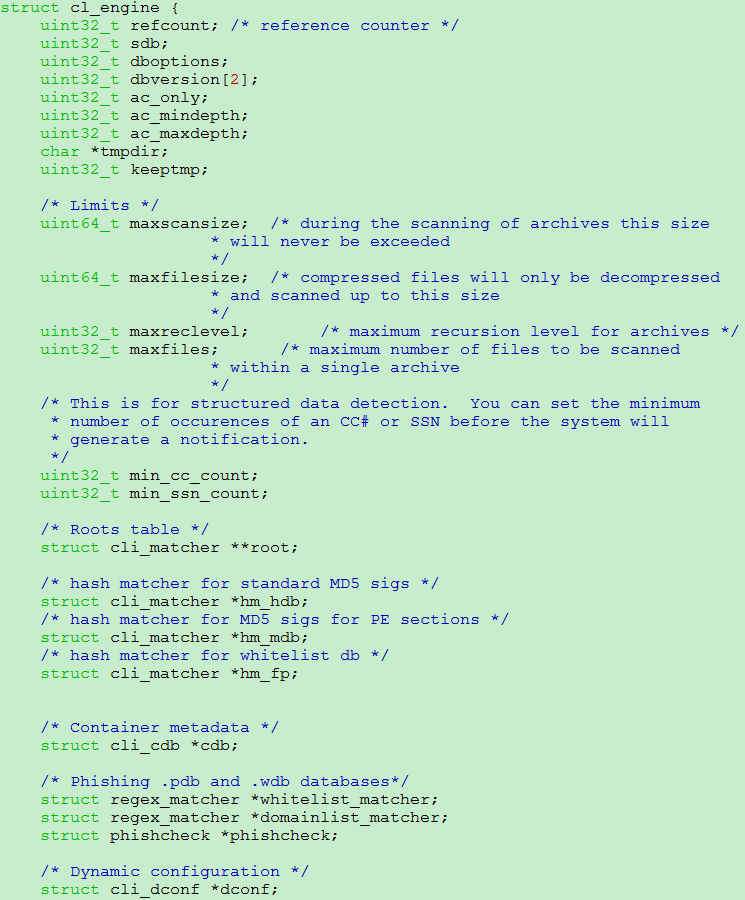
When Clamav doing specific file scan (clamscan.c), there are following procedures:

* Initialize data structures
* Set engine parameters
* Load signatures
* Scan

## Data structures

**cl\_engine**

this is a global data structure controls the behavior of a scan engine





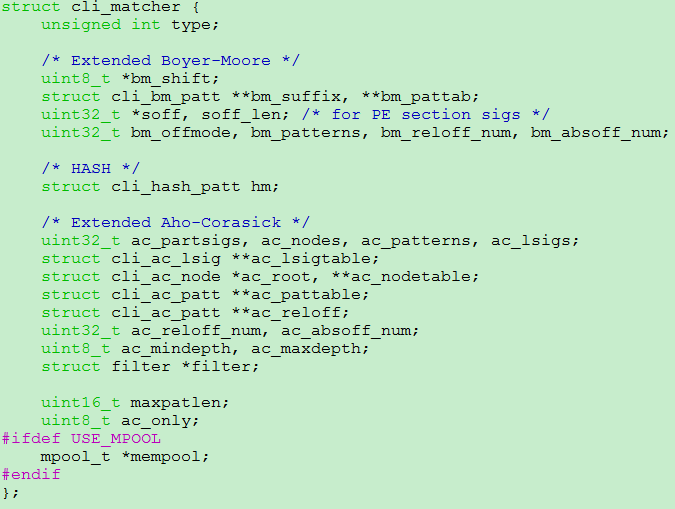
**cli\_matcher**

the data structure of BM, AC and HASH scan:

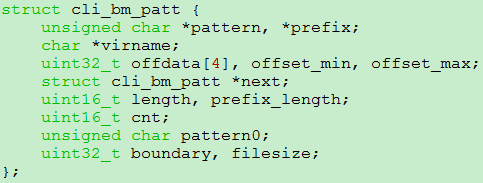
cli\_bm\_patt

cli\_ac\_patt

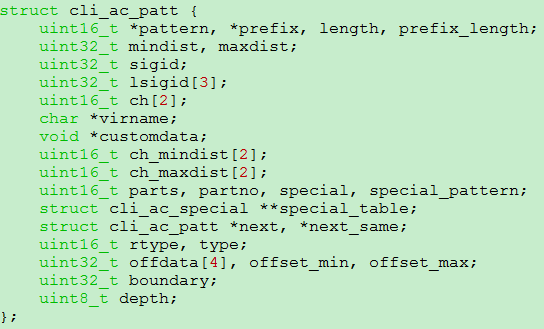
cli\_hash\_patt



**cli\_bm\_patt**



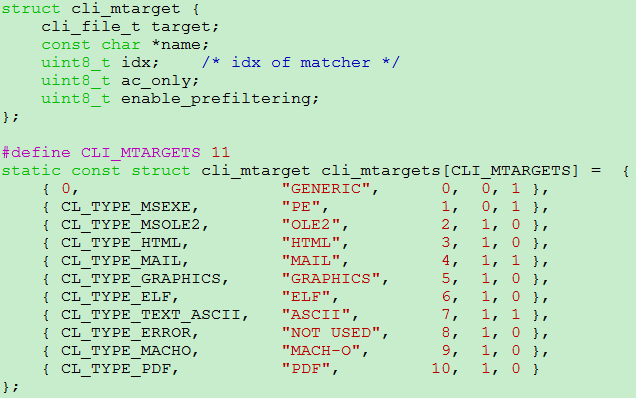
**cli\_ac\_patt**



**cli\_hash\_patt**



**cli\_mtarget**



Have BM scan: GENERIC and PE, others are AC only

Have filter: GENERIC, PE, MAIL and ASCII

Have both BM and filter: GENERIC and PE with wildcard and other detailed signature formats specified in cli\_parse\_add

## Test case

File: *test.txt*

testtesttesttestMYOtestTEST

ndb test

**create ndb signature for test.txt**

format

MalwareName:TargetType:Offset:HexSignature

where TargetType is one of the following numbers specifying the type of the

target file:

0: Any file

1: Portable Executable

2: OLE2 component (eg: VBA script)

3: HTML (normalized)

4: Mail File

5: Graphics

6: ELF

7: ASCII text file (normalized)

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.

user@ubuntu:~/clamav$ sigtool --hex-dump

testtesttesttestMYOtestTEST

746573747465737474657374746573744d594f74657374544553540a

File test.ndb

test\_ndb:0:0:746573747465737474657374746573744d594f74657374544553540a

sudo cp test.ndb /var/lib/clamav/test.ndb

## Engine initialiazation

scanmanager

cl\_engine\_new // init engine structure

cli\_mpool\_dconf\_init // dynamic configure

crtmgr\_init // certification

cl\_engine\_set\_str // config vars loaded from command line

cl\_engine\_set\_num // config vars loaded from command line

cl\_load // load virus database

phishing\_init // init phishing data structure

cli\_bytecode\_init // init bytecode data structure

cli\_cache\_init // init cache structure

cli\_load // load a virus file

cli\_loaddbdir // load a dir with virus files

cl\_engine\_compile // compile the engine

cli\_loadftm // load file types

cli\_ac\_buildtrie // build ac trie

cli\_ac\_buildtrie // build stat machine for AC match algo  
 cli\_build\_regex\_list

cli\_bytecode\_prepare2 // Compile bytecode

## Load signatures

**Functions of loading virus database**

cli\_cvdload

cli\_dconf\_load

cli\_loadcbc

cli\_loadcdb

cli\_loadcrt

cli\_loaddb

cli\_loadftm

cli\_loadhash

cli\_loadidb

cli\_loadign

cli\_loadinfo

cli\_loadldb

cli\_loadmd

cli\_loadmscat

cli\_loadndb

cli\_loadpdb

cli\_loadwdb

**the loading:**

if it’s a normal signature(hex numbers only), then load to BM db only, otherwise, if there’s wildcard involved, then should be loaded to AC db.

Meanwhile, if the signature specifics a target type, it should be loaded to a specified format’s root, otherwise, it should b be loaded to root[0](generic). Also,only generic and PE target type will be having BM db, so only signature with targe type=0 or 1 will be loaded into BM db if the signature is using number only.

During the db loading process, filter\_add\_static would be called to calculate prefiltering(using shift or FSM) data of the signatures which will speed up following bm scan a little bit.

cli\_bytecodde\_load is called in cli\_loadcdb to load the db for bytecode scan and in the process cl\_engine\_complie, and cli\_bytecode\_prepare2 is called to further initiate the bytecode engine

**load for ndb**

#define NDB\_TOKENS 6 // NDB have 6 fields

cli\_loadndb

cli\_initroots

for(i = 0; i < CLI\_MTARGETS; i++) {

if(cli\_mtargets[i].ac\_only || engine->ac\_only) root->ac\_only = 1;

cli\_ac\_init // allocate memory for

// root->ac\_root and root->ac\_root->trans

// config and init filter filter\_init, set all bits to 1:

// memset(m->B, ~0, sizeof(m->B));

// memset(m->end, ~0, sizeof(m->end));

if(!root->ac\_only) cli\_bm\_init // size = HASH(255, 255, 255) + 1;

// allocate memory for root->bm\_shift

// root->bm\_shift[i] = BM\_MIN\_LENGTH - BM\_BLOCK\_SIZE + 1;

engine->root[1]->bm\_offmode = 1; /\* BM offset mode for PE files \*/

target = (unsigned short) atoi(pt); // target is defined in each ndb record

root = engine->root[target];

cli\_parse\_add // add the pattern finally

**add pattern: select algo – AC or BM**

cli\_parse\_add

if (hexsig[0] == '$') // macro

cli\_ac\_addpatt

if((wild = strchr(hexsig, '{'))) // wildcard

if(sscanf(wild, "%c%u%c", &l, &range, &r) == 3 && l == '{' && r == '}' && range > 0 && range < 128) // recursively add

cli\_parse\_add

root->ac\_partsigs++;

cli\_ac\_addsig

if(strchr(hexsig, '\*')) // \*

cli\_ac\_addsig

if(root->ac\_only || type || lsigid || strpbrk(hexsig, "?([") || (root->bm\_offmode && (!strcmp(offset, "\*") || strchr(offset, ','))) || strstr(offset, "VI") || strchr(offset, '$')) // cases that also applies ac algo

// ac\_only

// targeting specific file type instead of generic

// PE’s bm offset mode with offset defined in signature

// have VI(version information) offset

cli\_ac\_addsig

if(the rest case) //numbers only

cli\_bm\_addpatt

**add pattern - BM**

cli\_bm\_addpatt

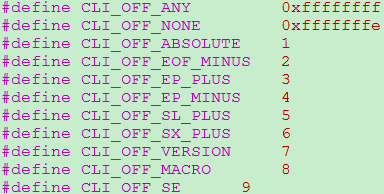
cli\_caloff // calculate offset information before adding

// offdata[0]: type

// offdata[1]: offset value

// offdata[2]: max shift

// offdata[3]: section number



for cli\_caloff, there would be absolute offset and relative offset, also there will be offset information defined in virus db record or passed in via cli\_target\_info which defines a offset info for a specific file type, and offset info via cli\_target\_info will be disabled in bm non offmode and enabled in bm offmode

offset\_min and offset\_max marks the range of the pattern if there’s string like “3,6”

if(!info) /\* decode offset string \*/

// will check offset string loaded from virus db record

// If contains “\*|,|EP+|EP-|SL+|EOF-|VI|$”, will calc the offset info

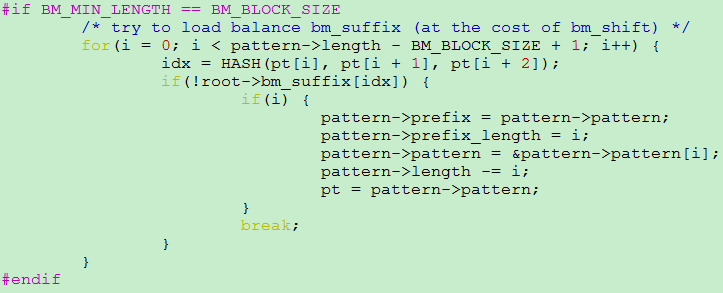
// Otherwise, just do “offdata[0] = CLI\_OFF\_ABSOLUTE”

else calc offset using specific file type’s offset info

if(root->filter && !root->bm\_offmode)

filter\_add\_static // do prefiltering calculation

bellow code will take chance to see if certain hash has not yet defined in other signatures, if it is the case and current position is not the start of a pattern, then load balance the bm\_suffix table via making the pattern shorter to the point where bm\_suffix is not defined and the part of the pattern before this point are treated as prefix



// set bm shift

root->bm\_shift[idx] = MIN(root->bm\_shift[idx], BM\_MIN\_LENGTH - BM\_BLOCK\_SIZE - i);

// then insert the bm\_suffix into hash chain where the item share the same index and the chain is sorted by first letter of the pattern

// inf in offset(PE) mode, will add to a data structure called bm\_pattab indexed by a int counting the bumber of the bm patterns

if(root->bm\_offmode)

root->bm\_pattab[root->bm\_patterns] = pattern;

root->bm\_patterns++;

## Scan

**scan logic design**

there are 4 scan methods

1. BM
2. AC
3. Hash
4. Bytecode

There are 2 entry points to begin a scan: cli\_map\_scandesc and cli\_magic\_scandesc

cli\_map\_scandesc will scan a file that is mapped to virtual memory already, this method is not yet used except in unit test case.

cli\_magic\_scandesc however is used for now as the primary entry of a scan and actually in a later stage, the file to be scanned will be mapped to memory also.

Before the actual scan, the type of the file is assumed as CL\_TYPE\_ANY, and the actual type of the incoming file would be decided with cli\_filetype2 at magic\_scandesc

After the filetype is decided, specific scan function dedicated to the file will be called directly. However, for ASCII file, - CL\_TYPE\_TEXT\_ASCII, the scan will only be called with certain config. So for ascii file, cli\_scanraw will be called to make the scan.

In raw scan, ASCII type will be assumed as CL\_TYPE\_ANY again and calling cli\_fmap\_scandesc to do further scan.

In cli\_fmap\_scandesc, according to **ftonly**(if configured as scan specific file type only) and **ftype**(the type of the file which will further decide the root to load) to decide the db to load and scan algo to use in match\_run:

* Generic db or type specific db
* BM(normal signature mode or offset mode, currently off mode is only enabled for PE type) or AC or Hash scan
* Hash scan will be performed if BM and AC scan return clean
* If hash scan is clean also, then logic code scan/bytecode scan will be performed via calling cli\_lsig\_eval and further cli\_magic\_scandesc\_type(mormal BM/AC scan), matchicon or cli\_bytecode\_runlsig(bytecode scan)
* Bytecode scan will be run finally via cli\_bytecode\_run
* Bytecode scan can also be triggered via cli\_pdf and cli\_scanpe
* The bytecode scan will be finally done at cli\_vm\_execute

In matcher\_run, a prefiltering(filter\_search\_ext) is called to reduce the length of actual scan if possible. After that, BM scan firstly and AC scan later is performed to match against the virus db loaded

Case of scanning a text file

**scanfile**

cl\_scandesc\_callback

scan\_common

// in normal case, argument ‘map’ passed into will be NULL

//except for cl\_scanmap\_callback in unit test

cli\_magic\_scandesc

// fmap function defined at libclamav/fmap.c

if(!(\*ctx->fmap = fmap(desc, 0, sb.st\_size)))

// call magic\_scandesc with type=CL\_TYPE\_ANY

// in cli\_magic\_scandesc\_type, will call magic\_scandesc with specific type

magic\_scandesc

if(type == CL\_TYPE\_ANY)

type = cli\_filetype2(\*ctx->fmap, ctx->engine);

call cli\_filetype

call cli\_texttype

filetype = cli\_ftname(type);

cache\_check // calculate hash for a file and do first hash scan???

// what is following doing???

hashed\_size = (\*ctx->fmap)->len;

old\_hook\_lsig\_matches = ctx->hook\_lsig\_matches;

ctx->hook\_lsig\_matches = NULL;

… …

ctx->hook\_lsig\_matches = cli\_bitset\_init();

in case CL\_TYPE\_TEXT\_ASCII, will not do cli\_scan\_structured

cli\_scanraw

unsigned int acmode = AC\_SCAN\_VIR

if(typercg) acmode |= AC\_SCAN\_FT; // specific value for acmode will be used in cli\_ac\_scanbuff

cli\_fmap\_scandesc(ctx, type == CL\_TYPE\_TEXT\_ASCII ? 0 : type, 0, &ftoffset, acmode, NULL, refhash)

**cli\_fmap\_scandesc**

// ftonly is the file type [assed ion, if is not CL\_TYPE\_ANY(=0), then ftonly is true

// ret=cli\_fmap\_scandesc(ctx, type == CL\_TYPE\_TEXT\_ASCII ? 0 : type, 0, &ftoffset, acmode, NULL, refhash), so ftonly is set here called from cli\_scandesc

if(!ftonly) groot = ctx->engine->root[0]; /\* generic signatures \*/

if(ftype) // in ascii text case, it is 0 which is converted to before

//now pick up a root for targets

/\* the metrix is:

If ftonly is set then use generic root, if recognized specific file type, then use corresponding root

ftony is set:

means engine will only scan structured file, hence will not use generic root

ftype is set:

means incoming file is a structured file so should pick a specific root

\*/

targetinfo(&info, i, map); // get offset and other info according to decided target type(root[index])

if(target == 1)

einfo = cli\_peheader; // PE

else if(target == 6)

einfo = cli\_elfheader; // ELF

else if(target == 9)

einfo = cli\_machoheader; // MACHO

else

return;

if(!ftonly)

cli\_ac\_initdata // init data for groot(generic root)

if(troot) // if use specific root for a file type, not applicable in this case

cli\_ac\_initdata

cli\_ac\_caloff

if(troot->bm\_offmode) // if in bm offset mode

cli\_bm\_initoff

if(!ftonly && hdb) // if it’s a specific file type and has hash db loaded, try do hash can preparation

if(troot)

matcher\_run with troot and ac mode decided via acmode

if(!ftonly)

matcher\_run with groot and ac mode decided via acmode

in matcher\_run calls

cli\_bm\_scanbuff

// byte by byte scan

shift=root->bmsift[idx]

if(shift==0)

//scan over whole pattern

else

i+=shift

cli\_ac\_scanbuff