# Clamav funcation call flow

ac scan with regex signature

***by eqmcc***

[Clamav funcation call flow 1](#_Toc345444841)

[The call flow 1](#_Toc345444842)

[Data structures 1](#_Toc345444843)

[Test case 1](#_Toc345444844)

[Engine initialiazation and load signatures 4](#_Toc345444845)

[Scan 9](#_Toc345444846)

## 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

See *flow\_normal\_sgin\_bm\_scan.pdf*

## Test case

*test.txt*

STARToooTESTkkkMYOtestTEST

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.

HexSignature

**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://wwws.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.

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

ooo\*kkk

6f6f6f2a6b6b6b

File test.ndb

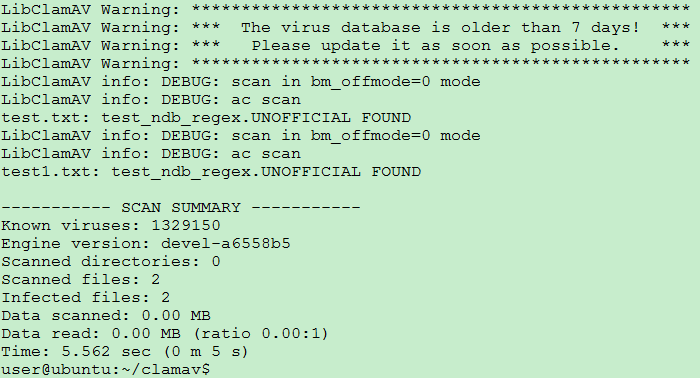
test\_ndb\_regex:0:3,5:6f6f6f{4}6b6b6b

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

Above is a regex based signature with offset info, so it will be loaded into ac pattern structure and will be used in ac scan mode.

The virus record will match start offset between 3 and 5 with pattern as “ooo{4}kkk”(i.e.: any file with sub string “ooo’any 4 bytes’kkk” with the sub string’s start at any position between absolute offset 3 and 5 will be identified as virus file)

**run**



## Engine initialiazation and load signatures

scanmanager

cl\_load

cli\_load

cli\_loadndb

cli\_initroots

cli\_ac\_init

filter\_init

cli\_bm\_init

cli\_parse\_add

// change from "6f6f6f{4}6b6b6b" to "6f6f6f????????6b6b6b"

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

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

hexcpy = cli\_calloc(hexlen + 2 \* range, sizeof(char));

strncpy(hexcpy, hexsig, wild - hexsig);

strcat(hexcpy, "??");

wild = strchr(wild, '}')

strcat(hexcpy, ++wild);

//call again

cli\_parse\_add(root, virname, hexcpy, rtype, type, offset, target, lsigid, options);

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

cli\_ac\_addsig

cli\_ac\_addpatt

cli\_ac\_addpatt **//\***

filter\_add\_acpatt

cli\_caloff **//\***

**the loading:**

this signature “test\_ndb\_regex:0:3,5:6f6f6f{4}6b6b6b” has regular expression involved, so should be loaded into AC scan sturcture.

Meanwhile, if the signature doesn’t specific a target type, it should be loaded to root[0](generic).

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

**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, '{'))) // regular expression

if(sscanf(wild, "%c%u%c", &l, &range, &r) == 3 && l == '{' && r == '}' && range > 0 && range < 128) // dealing case as “{a,b}”

// change from "6f6f6f{4}6b6b6b" to "6f6f6f????????6b6b6b"

hexcpy = cli\_calloc(hexlen + 2 \* range, sizeof(char));

strncpy(hexcpy, hexsig, wild - hexsig);

strcat(hexcpy, "??");

wild = strchr(wild, '}')

strcat(hexcpy, ++wild);

//call again

cli\_parse\_add(root, virname, hexcpy, rtype, type, offset, target, lsigid, options);

else // dealing case as “string{a,b}string{c,d}” - partial sigs

root->ac\_partsigs++;

// find all the partial sigs

for(i = 0; i < hexlen; i++)

// each hex string besides “{}” or {\*} will be split into two partial sigs

if(hexsig[i] == '{' || hexsig[i] == '\*') parts++;

// adding each sig into ac tire

start = pt = hexcpy;

for(i = 1; i <= parts; i++)

for(j = 0; j < strlen(start); j++)

if(start[j] == '{') asterisk = 0; // has not asterisk

// dealing case as “string{a,b}string\*string{c,d}” - partial sigs

if(start[j] == '\*') asterisk = 1; // has asterisk

ret = cli\_ac\_addsig(root, virname, start, root->ac\_partsigs, parts, i, rtype, type, mindist, maxdist, offset, lsigid, options)

// each hex string besides “{}” or {\*} will be split into two partial sigs

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

root->ac\_partsigs++;

for(i = 0; i < hexlen; i++) if(hexsig[i] == '\*') parts++;

for(i = 1; i <= parts; i++)

pt = cli\_strtok(hexsig, i - 1, "\*")

ret = cli\_ac\_addsig(root, virname, pt, root->ac\_partsigs, parts, i, rtype, type, 0, 0, offset, lsigid, options)

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

// enters here with ‘?’

cli\_ac\_addsig

if(the rest case) //numbers only

cli\_bm\_addpatt

**add signature(pre processing for regular expression) - AC**

cli\_ac\_addsig

new->ch[0] |= CLI\_MATCH\_IGNORE;

new->ch[1] |= CLI\_MATCH\_IGNORE;

// dealing case as “[]” – “HEXSIG[x-y]aa or aa[x-y]HEXSIG”

if(strchr(hexsig, '[')) // with “[“ – [] means a range, special case

for(i = 0; i < 2; i++)

pt = strchr(hex, '[')

pt2 = strchr(pt, ']')

sscanf(pt, "%u-%u", &n1, &n2) // AC\_CH\_MAXDIST=3

if(strlen(hex) == 2)

dec = cli\_hex2ui(hex); // case “aa[x-y]HEXSIG”

new->ch[i] = \*dec;

new->ch\_mindist[i] = n1;

new->ch\_maxdist[i] = n2;

if(strlen(pt2) == 2)

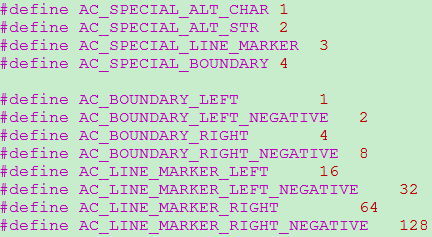
dec = cli\_hex2ui(pt2); // case “HEXSIG[x-y]aa”

new->ch[i] = \*dec;

new->ch\_mindist[i] = n1;

new->ch\_maxdist[i] = n2;

// special types



// dealing case as “()” – “(aa|bb|cc|..) or ! (aa|bb|cc|..) or (B) or (L)”

if(strchr(hexsig, '(')) // with “(“ – () means or, special case

start = pt = hexcpy;

while((pt = strchr(start, '('))) // for each “()”

/\* struct cli\_ac\_special {

unsigned char \*str;

struct cli\_ac\_special \*next;

uint16\_t len, num;

uint8\_t type, negative;

}; \*/

newspecial = (struct cli\_ac\_special \*) mpool\_calloc(root->mempool, 1, sizeof(struct cli\_ac\_special));

if(pt >= hexcpy + 2) if(pt[-2] == '!') // case “! (aa|bb|cc|..)”

newspecial->negative=1; // case “(aa|bb|cc|..)”

// newspecial->negative = 0

start = strchr(pt, ')')

if(!strcmp(pt, "B")) // case “(B)”

if(!\*start)

new->boundary |= AC\_BOUNDARY\_RIGHT;

if(newspecial->negative)

new->boundary |= AC\_BOUNDARY\_RIGHT\_NEGATIVE;

if(pt - 1 == hexcpy)

new->boundary |= AC\_BOUNDARY\_LEFT;

if(newspecial->negative)

new->boundary |= AC\_BOUNDARY\_LEFT\_NEGATIVE;

if(!strcmp(pt, "L")) // case “(L)”

if(!\*start)

new->boundary |= AC\_LINE\_MARKER\_RIGHT; if(newspecial->negative)

new->boundary |= AC\_LINE\_MARKER\_RIGHT\_NEGATIVE;

if(pt - 1 == hexcpy)

new->boundary |= AC\_LINE\_MARKER\_LEFT;

if(newspecial->negative)

new->boundary |= AC\_LINE\_MARKER\_LEFT\_NEGATIVE;

// create new special table with old one copied over

new->special++;

newtable = (struct cli\_ac\_special \*\*) mpool\_realloc(root->mempool, new->special\_table, new->special \* sizeof(struct cli\_ac\_special \*));

newtable[new->special - 1] = newspecial;

new->special\_table = newtable;

if(!strcmp(pt, "B")) newspecial->type = AC\_SPECIAL\_BOUNDARY;

if(!strcmp(pt, "L")) newspecial->type = AC\_SPECIAL\_LINE\_MARKER;

else // case “(xx|yy|zz) or (a|b|c)”

newspecial->num = 1;

for(i = 0; i < strlen(pt); i++)

if(pt[i] == '|') newspecial->num++;

// case “(a|b|c)”

if(3 \* newspecial->num - 1 == (uint16\_t) strlen(pt))

newspecial->type = AC\_SPECIAL\_ALT\_CHAR;

newspecial->str = (unsigned char \*) mpool\_malloc(root->mempool, newspecial->num);

// case “(xx|yy|zz)

else newspecial->type = AC\_SPECIAL\_ALT\_STR;

for(i = 0; i < newspecial->num; i++)

if(newspecial->num == 1) // case of only 1 “|”

c = (char \*) cli\_mpool\_hex2str(root->mempool, pt);

else // case multiple “|”

(h = cli\_strtok(pt, i, "|")

c = (char \*) cli\_mpool\_hex2str(root->mempool, h);\

// alternative chars stored in array and alternative strings stored in chain

if(newspecial->type == AC\_SPECIAL\_ALT\_CHAR)

newspecial->str[i] = \*c; // set the char

else // string case

if(i)

specialpt = newspecial;

// insert the string into chain of alterative

while(specialpt->next)

specialpt = specialpt->next;

specialpt->next = (struct cli\_ac\_special \*) mpool\_calloc(root->mempool, 1, sizeof(struct cli\_ac\_special));

specialpt->next->str = (unsigned char \*) c;

else newspecial->str = (unsigned char \*) c;

// sort the char array

if(newspecial->num>1 && newspecial->type == AC\_SPECIAL\_ALT\_CHAR)

cli\_qsort(newspecial->str, newspecial->num, sizeof(unsigned char), qcompare);

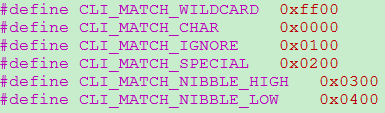
// dealing other case

new->pattern = cli\_mpool\_hex2ui(root->mempool, hex ? hex : hexsig);

// new->pattern is uint16\_t

cli\_mpool\_hex2ui

cli\_realhex2ui // in this function, each byte of the pattern would be extended to uint16\_t(low byte for the pattern byte and high byte for the matching type corresponding to the regular expression type)



if(hex[i] == '?' && hex[i + 1] == '?') val |= CLI\_MATCH\_IGNORE;

if(hex[i + 1] == '?') val |= CLI\_MATCH\_NIBBLE\_HIGH;

if(hex[i] == '?') val |= CLI\_MATCH\_NIBBLE\_LOW;

if(hex[i] == '(') val |= CLI\_MATCH\_SPECIAL;

filter\_add\_acpatt **//\*** prefiltering

// check if there’s regex in first letters

if(new->pattern[i] & CLI\_MATCH\_WILDCARD)

cli\_caloff //“test\_ndb\_regex:0:3,5:6f6f6f{4}6b6b6b”

if((pt = strchr(offcpy, ','))) offdata[2] = atoi(pt + 1); // which is 5

offdata[0] = CLI\_OFF\_ABSOLUTE;

\*offset\_min = offdata[1] = atoi(offcpy); // which is 3

\*offset\_max = \*offset\_min + offdata[2]; // which is 8

cli\_ac\_addpatt

**add pattern to AC tire**

cli\_ac\_addpatt

uint16\_t len = MIN(root->ac\_maxdepth, pattern->length);

// root->ac\_maxdepth is set via CLI\_DEFAULT\_AC\_MAXDEPTH

for(i = 0; i < len; i++)

next = pt->trans[(unsigned char) (pattern->pattern[i] & 0xff)];

if(!next) // this tran does not yet exist

next = (struct cli\_ac\_node \*) mpool\_calloc(root->mempool, 1, sizeof(struct cli\_ac\_node)); // allocate

newtable = mpool\_realloc(root->mempool, root->ac\_nodetable, root->ac\_nodes \* sizeof(struct cli\_ac\_node \*)); // allocate a new node table to copy over the old ones and store the new one, copy over is done automatically via mpool\_realloc

root->ac\_nodetable = (struct cli\_ac\_node \*\*) newtable;

root->ac\_nodetable[root->ac\_nodes - 1] = next;

// put into the tire-

pt->trans[(unsigned char) (pattern->pattern[i] & 0xff)] = next;

else

pt = next // next char

// create new pattern table and copy over

newtable = mpool\_realloc(root->mempool, root->ac\_pattable, root->ac\_patterns \* sizeof(struct cli\_ac\_patt \*));

root->ac\_pattable = (struct cli\_ac\_patt \*\*) newtable;

root->ac\_pattable[root->ac\_patterns - 1] = pattern;

/\*

ac node would have a list of ac patterns that share the same prefix

if there is pattern list, need to insert current one into it, sort according to the first 2 latters of the pattern

also the ac tree only accept a max depth of 3

\*/

// pt is ac node and ph is ac pattern and now pt is pointing at leaf of this pattern in the ac tire

ph = pt->list; // the list only exists when the last node in the ac tire is shared by other patterns

ph\_add\_after = ph\_prev = NULL;

while(ph) // if leaf is shared by other patterns which is highly possible as only first 3 bytes of the signature is used to build the ac tire, then try to insert it to the shared pattern list, also of the pattern or subpattern are same, should also add into a structure called pattern->next\_same

// compare partno???

if(!ph\_add\_after && ph->partno <= pattern->partno && (!ph->next || ph->next->partno > pattern->partno))

ph\_add\_after = ph;

// same pattern length, same prefix length and same first two letters

// ending in same leaf, need to further confirm if the two pattern are same or similar

if((ph->length == pattern->length) && (ph->prefix\_length == pattern->prefix\_length) && (ph->ch[0] == pattern->ch[0]) && (ph->ch[1] == pattern->ch[1]))

// if the characters part of the two pattern are exact the same, compare other info in the signature

if(!memcmp(ph->pattern, pattern->pattern, ph->length \* sizeof(uint16\_t)) && !memcmp(ph->prefix, pattern->prefix, ph->prefix\_length \* sizeof(uint16\_t)))

// if no other regex special case, the two sig are exact match

if(!ph->special && !pattern->special) match = 1

if(ph->special == pattern->special)

//compare the special info

a1 = ph->special\_table[i];

a2 = pattern->special\_table[i];

else match = 0;

if(match) // sig info is the same

// insert into next\_same(same signature list) and sorting according to partno

if(pattern->partno < ph->partno)

pattern->next\_same = ph; // insert into same pattern list

if(ph\_prev) ph\_prev->next = ph->next; // remove ph from the leaf node’s pattern list since it is added into same pattern list of current pattern

else pt->list = ph->next; // removing from current pattern’s list

else

while(ph->next\_same && ph->next\_same->partno < pattern->partno)

ph = ph->next\_same;

pattern->next\_same = ph->next\_same;

ph->next\_same = pattern;

else

// try next pattern in the list

ph\_prev = ph;

ph = ph->next;

if(ph\_add\_after) // insert

pattern->next = ph\_add\_after->next;

ph\_add\_after->next = pattern;

else // append in head

pattern->next = pt->list;

pt->list = pattern;

**compile the tire to build the data structure for ac scan(build goto/fail/jump table)**

cl\_engine\_compile

cli\_loadftm

in cli\_loadftm loading: MPEG video stream,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: MPEG sys stream,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: GZip,CL\_TYPE\_ANY,CL\_TYPE\_GZ

in cli\_loadftm loading: SCRENC,CL\_TYPE\_ANY,CL\_TYPE\_SCRENC

in cli\_loadftm loading: PostScript,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: BinHex,CL\_TYPE\_ANY,CL\_TYPE\_BINHEX

in cli\_loadftm loading: Real Media File,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: BMP,CL\_TYPE\_ANY,CL\_TYPE\_GRAPHICS

in cli\_loadftm loading: BZip,CL\_TYPE\_ANY,CL\_TYPE\_BZ

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Eserv mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: MBox,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Exim mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: GIF,CL\_TYPE\_ANY,CL\_TYPE\_GRAPHICS

in cli\_loadftm loading: Qmail bounce,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: MP3,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: MS CHM,CL\_TYPE\_ANY,CL\_TYPE\_MSCHM

in cli\_loadftm loading: MS-EXE/DLL,CL\_TYPE\_ANY,CL\_TYPE\_MSEXE

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Ogg Stream,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: ZIP,CL\_TYPE\_ANY,CL\_TYPE\_ZIP

in cli\_loadftm loading: ZIP,CL\_TYPE\_ANY,CL\_TYPE\_ZIP

in cli\_loadftm loading: RIFF,CL\_TYPE\_ANY,CL\_TYPE\_RIFF

in cli\_loadftm loading: RIFX,CL\_TYPE\_ANY,CL\_TYPE\_RIFF

in cli\_loadftm loading: RAR,CL\_TYPE\_ANY,CL\_TYPE\_RAR

in cli\_loadftm loading: Raw mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Maildir,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Maildir,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: compress.exed,CL\_TYPE\_ANY,CL\_TYPE\_MSSZDD

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: EVS mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Symantec,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: ARJ,CL\_TYPE\_ANY,CL\_TYPE\_ARJ

in cli\_loadftm loading: UUencoded,CL\_TYPE\_ANY,CL\_TYPE\_UUENCODED

in cli\_loadftm loading: VPOP3 Mail (UNIX),CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: VPOP3 Mail (DOS),CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: TNEF,CL\_TYPE\_ANY,CL\_TYPE\_TNEF

in cli\_loadftm loading: ELF,CL\_TYPE\_ANY,CL\_TYPE\_ELF

in cli\_loadftm loading: PNG,CL\_TYPE\_ANY,CL\_TYPE\_GRAPHICS

in cli\_loadftm loading: CryptFF,CL\_TYPE\_ANY,CL\_TYPE\_CRYPTFF

in cli\_loadftm loading: OLE2 container,CL\_TYPE\_ANY,CL\_TYPE\_MSOLE2

in cli\_loadftm loading: JPEG,CL\_TYPE\_ANY,CL\_TYPE\_GRAPHICS

in cli\_loadftm loading: MP3,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: PE,CL\_TYPE\_ANY,CL\_TYPE\_MSEXE

in cli\_loadftm loading: ZIP-SFX,CL\_TYPE\_ANY,CL\_TYPE\_ZIPSFX

in cli\_loadftm loading: RAR-SFX,CL\_TYPE\_ANY,CL\_TYPE\_RARSFX

in cli\_loadftm loading: ARJ-SFX,CL\_TYPE\_ANY,CL\_TYPE\_ARJSFX

in cli\_loadftm loading: ARJ-SFX,CL\_TYPE\_ANY,CL\_TYPE\_ARJSFX

in cli\_loadftm loading: ARJ-SFX,CL\_TYPE\_ANY,CL\_TYPE\_ARJSFX

in cli\_loadftm loading: AUTOIT,CL\_TYPE\_ANY,CL\_TYPE\_AUTOIT

in cli\_loadftm loading: NSIS,CL\_TYPE\_ANY,CL\_TYPE\_NULSFT

in cli\_loadftm loading: SIP log,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: HTML data,CL\_TYPE\_ANY,CL\_TYPE\_HTML

in cli\_loadftm loading: RTF,CL\_TYPE\_ANY,CL\_TYPE\_RTF

in cli\_loadftm loading: TAR-POSIX,CL\_TYPE\_ANY,CL\_TYPE\_POSIX\_TAR

in cli\_loadftm loading: mirc ini,CL\_TYPE\_ANY,CL\_TYPE\_SCRIPT

in cli\_loadftm loading: Mail file,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail file,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail file,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail file,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mach-O LE,CL\_TYPE\_ANY,CL\_TYPE\_MACHO

in cli\_loadftm loading: Mach-O LE 64-bit,CL\_TYPE\_ANY,CL\_TYPE\_MACHO

in cli\_loadftm loading: Mach-O BE,CL\_TYPE\_ANY,CL\_TYPE\_MACHO

in cli\_loadftm loading: Mach-O BE 64-bit,CL\_TYPE\_ANY,CL\_TYPE\_MACHO

in cli\_loadftm loading: Universal Binary/Java Bytecode,CL\_TYPE\_ANY,CL\_TYPE\_MACHO\_UNIBIN

in cli\_loadftm loading: 7zip,CL\_TYPE\_ANY,CL\_TYPE\_7Z

in cli\_loadftm loading: Mail file,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: Mail file,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: CPIO NEWC,CL\_TYPE\_ANY,CL\_TYPE\_CPIO\_NEWC

in cli\_loadftm loading: CPIO CRC,CL\_TYPE\_ANY,CL\_TYPE\_CPIO\_CRC

in cli\_loadftm loading: CPIO ODC,CL\_TYPE\_ANY,CL\_TYPE\_CPIO\_ODC

in cli\_loadftm loading: CPIO OLD BINARY BE,CL\_TYPE\_ANY,CL\_TYPE\_CPIO\_OLD

in cli\_loadftm loading: CPIO OLD BINARY LE,CL\_TYPE\_ANY,CL\_TYPE\_CPIO\_OLD

in cli\_loadftm loading: ISHIELD-MSI,CL\_TYPE\_ANY,CL\_TYPE\_ISHIELD\_MSI

in cli\_loadftm loading: PDF document,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF document,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF document,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: PDF,CL\_TYPE\_ANY,CL\_TYPE\_PDF

in cli\_loadftm loading: SYM DATFILE,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: PDF image,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: SQLite WAL,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: SQLite WAL,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: SQLite database,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: SQLite journal,CL\_TYPE\_ANY,CL\_TYPE\_IGNORED

in cli\_loadftm loading: SWF (compressed),CL\_TYPE\_ANY,CL\_TYPE\_SWF

in cli\_loadftm loading: SWF (uncompressed),CL\_TYPE\_ANY,CL\_TYPE\_SWF

in cli\_loadftm loading: MS CAB,CL\_TYPE\_ANY,CL\_TYPE\_MSCAB

in cli\_loadftm loading: CAB-SFX,CL\_TYPE\_ANY,CL\_TYPE\_CABSFX

in cli\_loadftm loading: ISO9660,CL\_TYPE\_ANY,CL\_TYPE\_ISO9660

in cli\_loadftm loading: TAR-POSIX-CVE-2012-1419,CL\_TYPE\_ANY,CL\_TYPE\_POSIX\_TAR

in cli\_loadftm loading: SIS,CL\_TYPE\_ANY,CL\_TYPE\_SIS

in cli\_loadftm loading: Mail file,CL\_TYPE\_ANY,CL\_TYPE\_MAIL

in cli\_loadftm loading: JPEG2000,CL\_TYPE\_ANY,CL\_TYPE\_GRAPHICS

cli\_ac\_buildtrie

ac\_maketrans

ac\_maketrans

## 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