Clamav funcation call flow(on-access scan)

on-access scan with clamfs

by eqmcc

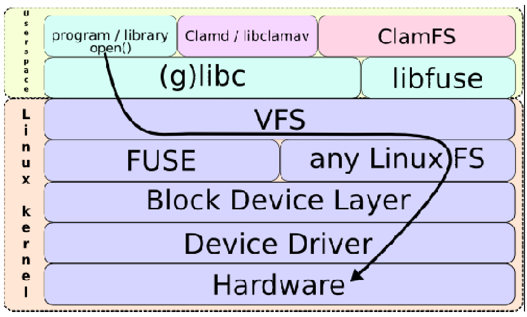
Description

this document will talk about on-access scan in clamav with the support from clamfs.

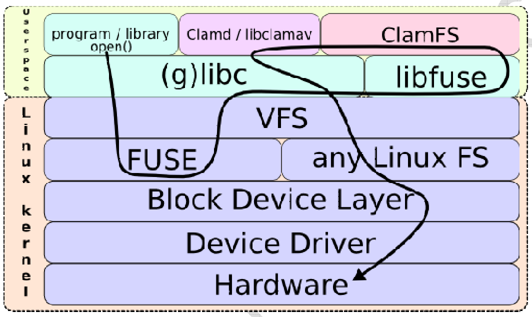
1. about clamfs

ClamFS(http://clamfs.sf.net/) is a user space file system based on FUSE.

compared to normal vfs file operation from glibc, clamfs go through extra layers in both user space and kernel space to hook to the clamav daemon asking for virus scan service.

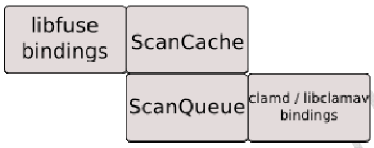


vfs file open

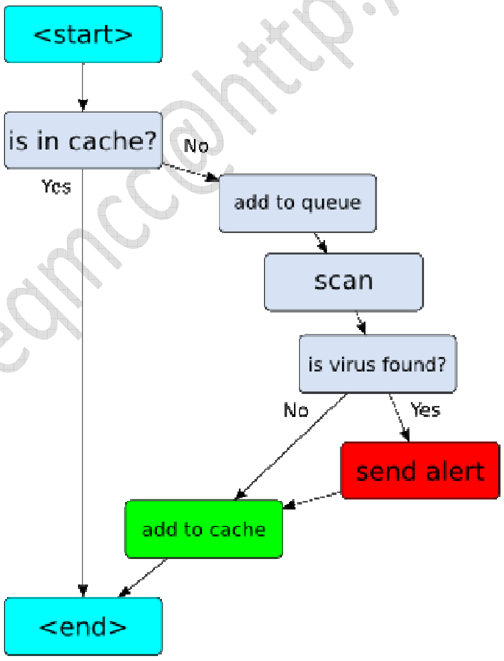


clamfs file open

in addition to above internals, clamfs also implements a internal cache (LRU with time-based and out-of-memory expiration) to speed up the file scan:



the whole scan flow is as bellow:

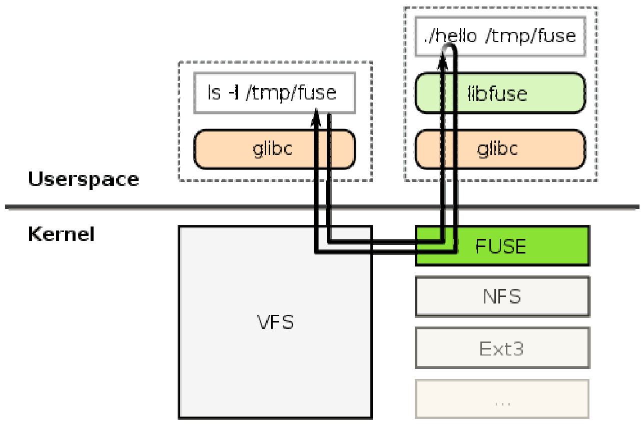


Note: all the diagrams are referred from the official web page of ClamFS.

2. about FUSE

FUSE(**F**ilesystem in **Use**rspace, http://fuse.sourceforge.net/) can be employed to make it possible to implement a fully functional file system in a user space. FUSE was originally developed to support ***AVFS*** but it has since became a separate project. Now quite a few other projects are using it.

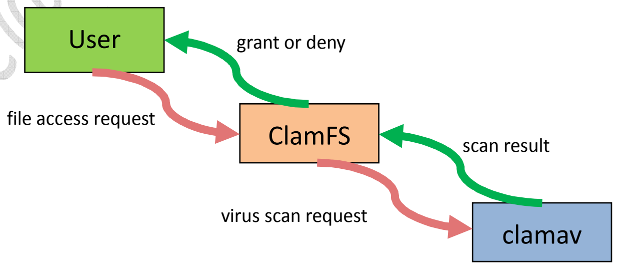
the path of a filesystem call (e.g. stat) in FUSE is as following:
(referred from http://zh.wikipedia.org/wiki/File:FUSE_structure.svg)



FUSE is used in ZFS， glusterfs, luster and clamfs, etc..

3. about clamav on-access scan

when a file open request is generated, it will be routed to clamfs, where, a request to the running clamav daemon to scan the file in question will be made. If the file is identified as infected by virus, the access to the file content will be rejected by clamfs.



Data structures

TBD

Test case

an access to file in ClamFS will match the bytecode signature in clamav virus database.

compile and install ClamFS

Patch - make following patch to make sure the compile will success

```
--- a/src/scancache.cxx
+++ b/src/scancache.cxx
@@ -37,7 +37,8 @@ CachedResult::~CachedResult() {
 }

 ScanCache::ScanCache(long int elements, long int expire):
-       ExpireLRUCache<ino_t, CachedResult>::ExpireLRUCache<ino_t, CachedResult>(elements, expire) {
+       ExpireLRUCache<ino_t, CachedResult>::ExpireLRUCache(elements, expire) {
 }
```

Preparation on dependencies

```
sudo apt-get install fuse // should be installed already
sudo apt-get install fuse-utils // should be installed already
sudo apt-get install libfuse-dev
sudo apt-get install librlog-dev
sudo apt-get install libpoco-dev
sudo apt-get install libccgnu2-1.7-0
sudo apt-get install libcommoncpp2-dev
```

Compile and Install

```
./configure
```

```
make
sudo make install
```

refer to http://www.drtak.org/teaches/modules/0169/module.pdf

Setting up “mount points"

```
mkdir ~/.customs
mkdir ~/customs
```

Setting up the configuration file - clamfs.xml

```
<clamd socket="/tmp/clamd.socket" check="yes" /> // location of the socket can be found in /etc/clamd.conf
<filesystem root="/home/user/.customs" mountpoint="/home/user/customs" public="no" />
```

bytecode preparation

source code

test_bytecode_on_access.c

```
VIRUSNAME_PREFIX("test_bytecode_on_access.c")
VIRUSNAMES("A","B")
TARGET(7)
SIGNATURES_DECL_BEGIN
DECLARE_SIGNATURE(magic)
SIGNATURES_DECL_END

SIGNATURES_DEF_BEGIN
DEFINE_SIGNATURE(magic,"61616262")        // the pattern as "aabb" in hex
SIGNATURES_END
bool logical_trigger (void)
{
        // @ clamav-bytecode-compiler/obj/Release/lib/clang/1.1/include/bytecode_local.h
        return count_match(Signatures.magic) != 1; // if "aabb" match count is ‘1’, it’s not a virus
}

int entrypoint (void)
{
        int count = count_match(Signatures.magic);
        if ( count == 3) foundVirus("B"); // 3 matches of "aabb", find virus B
        else if ( count != 0) foundVirus ("A"); // have match but no 3 times, find virus A
        return 0;
}
```

Note:
in production mode, clamav will not accept unsigned bytecode, so in order to make this test case work, following patch should be made to clamav code(also, an alternative solution is use normal signature rather than bytecode):

```
--- a/clamd/clamd.c
+++ b/clamd/clamd.c
@@ -462,6 +462,8 @@ int main(int argc, char **argv)
            dboptions |= CL_DB_BYTECODE_UNSIGNED;
            logg("#Bytecode: Enabled support for unsigned bytecode.\n");
        }
+       dboptions |= CL_DB_BYTECODE_UNSIGNED; //CHR always enable loading unsigned bytecode
+
        if((opt = optget(opts,"BytecodeMode"))->enabled) {
            enum bytecode_mode mode;
            if (!strcmp(opt->strarg, "ForceJIT"))
diff --git a/clamscan/manager.c b/clamscan/manager.c
```

compile and put in clamav virus db

```
sudo cp test_bytecode_on_access.cbc /var/lib/clamav
```

test file

test.txt <= virus match

```
aabbxxxxxxxxxxxxxxxxxxxxaabbxxxxxxxxxxxxaabb
```

test1.txt <= no match

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

test

clamfs start and stop script

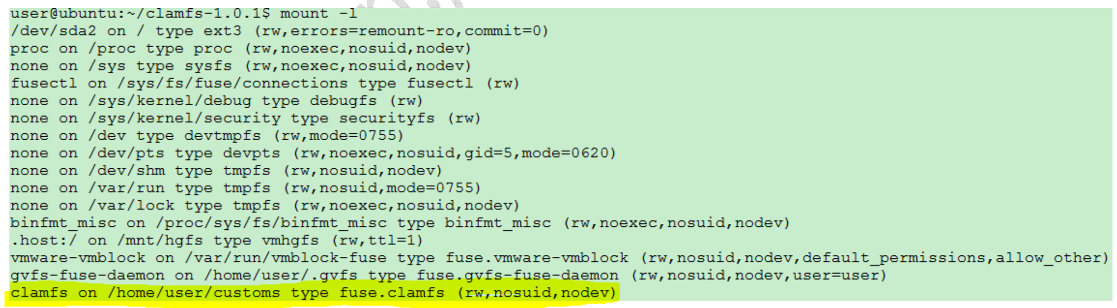
clamfs.start

```
# start clamd
sudo clamd
sleep 1
# start clamfs
sudo clamfs /home/user/clamfs.xml
```

clamfs.stop

```
# stop clamd
CLAMD_PID=`ps -ef | grep clamd | grep -v auto | awk '{print $2}'`
sudo kill    $CLAMD_PID
sleep 1
# stop clamfs
CLAMFS_PID=`ps -ef | grep clamfs.xml | grep -v auto | awk '{print $2}'`
sudo kill    $CLAMFS_PID
sleep 1
# umount
sudo umount /home/user/customs
```

check mounted file system



```
user@ubuntu:~/clamfs-1.0.1$ mount -l
/dev/sda2 on / type ext3 (rw,errors=remount-ro,commit=0)
proc on /proc type proc (rw,noexec,nosuid,nodev)
none on /sys type sysfs (rw,noexec,nosuid,nodev)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
none on /dev type devtmpfs (rw,mode=0755)
none on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
none on /dev/shm type tmpfs (rw,nosuid,nodev)
none on /var/run type tmpfs (rw,nosuid,mode=0755)
none on /var/lock type tmpfs (rw,noexec,nosuid,nodev)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noexec,nosuid,nodev)
.host:/ on /mnt/hgfs type vmhgfs (rw,ttl=1)
vmware-vmblock on /var/run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,default_permissions,allow_other)
gvfs-fuse-daemon on /home/user/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev,user=user)
clamfs on /home/user/customs type fuse.clamfs (rw,nosuid,nodev)
```

the test

copy file to clamfs

```
sudo cp test.txt /home/user/customs
sudo cp test1.txt /home/user/customs
```

try to access the file

```
sudo file /home/user/customs/test.txt
```

```
/home/user/customs/test.txt: writable, regular file, no read permission <== access denied
```

```
sudo cat /home/user/customs/test1.txt
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx <== access granted
```

check logs

clamd

```
user@ubuntu:~/clamfs-1.0.1$ sudo tail /tmp/clamd.log
Sun Apr  7 06:31:46 2013 -> Self checking every 600 seconds.
Sun Apr  7 06:31:46 2013 -> ERROR: Can't save PID in file /var/run/clamd.pid
Sun Apr  7 06:31:46 2013 -> Listening daemon: PID: 4598
Sun Apr  7 06:31:46 2013 -> MaxQueue set to: 100
Sun Apr  7 06:34:02 2013 -> /home/user/.customs/test.txt: test_bytecode_on_access.c.B(cb5132ddcb50f30d0185a19841249bad:45) FOUND
Sun Apr  7 06:44:02 2013 -> No stats for Database check - forcing reload
Sun Apr  7 06:44:02 2013 -> Reading databases from /var/lib/clamav
Sun Apr  7 06:44:09 2013 -> Database correctly reloaded (1727016 signatures)
Sun Apr  7 06:54:09 2013 -> SelfCheck: Database status OK.
Sun Apr  7 07:00:10 2013 -> /home/user/.customs/test1.txt: OK
```

<== both test.txt and test1.txt are scanned

clamfs

```
user@ubuntu:~/clamfs-1.0.1$ sudo tail /var/log/clamfs.log
05:57:43 (stats.cxx:63) Files bigger than maximal-size: 0
05:57:43 (stats.cxx:65) open() function called 5 times (allowed: 0, denied: 5)
05:57:43 (stats.cxx:66) Scan failed 0 times
05:57:43 (stats.cxx:67) --- end of statistics ---
05:57:43 (clamfs.cxx:1159) deleting stats
05:57:43 (clamfs.cxx:1165) deleting extensions ACL
05:57:43 (clamfs.cxx:1170) closing logging targets
06:31:49 (rlog.cxx:105) log goes to file /var/log/clamfs.log
06:31:49 (clamfs.cxx:1134) extension ACL size is 47 entries
06:34:02 (clamav.cxx:138) (file:4662) (root:0) /home/user/.customs/test.txt: test bytecode on access.c.B FOUND
```

<== access to test.txt is denied

==

The call flow

initialize the file system based on FUSE in main

includes:

1. initialize fuse file operation functions
2. read in config file
3. connect and test against clamd unix domain socket
4. initialize LRU cache based on Poco::ExpireLRUCache
5. register to FUSE with
   a) argv[0]
   b) config["mountpoint"]
   c) config["public"]/ config["nonempty"]/ config["readonly"]/ config["threads"]/ config["fork"]

char **fuse_argv; // argument array for fuse

```
fuse_operations clamfs_oper; // argument for fuse, used for defining fs operations

ConfigParserXML cp(argv[1]); // read in the config file, clamfs.xml

// minimal set of configuration options needed:
// config["socket"]
// config["root"]
// config["mountpoint"]

// open and test clamd socket - it is a unix domain socket
OpenClamav(config["socket"])
PingClamav()
CloseClamav();// close the socket

// Initialize cache use:
// config["entries"]
// config["expire"]
/*
        <!-- How many entries to keep in cache and for how long -->
        <cache entries="16384" expire="10800000" /> <!-- time in ms, 3h -->
*/
/*
       the ScanCache is brief LRU cache for anti-virus scan results storage
       LRU cache with time-based expiration. Based on Poco::ExpireLRUCache.
       This cache stores anti-virus scan results for later use.
*/
cache = new ScanCache(atol(config["entries"]), atol(config["expire"]));

// register to FUSE, after this point, operation in clamfs will be handled by fuse with
defined operation functions
fuse_main(fuse_argc, fuse_argv, &clamfs_oper)
```

on-access scan

when a file operation request is generated against a file in ClamFS mount point(/home/user/customs dir in this case), it will be handled by FUSE and finally calls clamfs_open function where all tricks are hidden

in clamfs_open, the file will be checked firstly against the LRU cache to see if there's a match in the cache already which will save the effort of request clamd scan service. if there's a match, will call open_backend to actually open the file and if there's no match, then the file will be handle to clamav daemon process by issuing “SCAN” command in the opened socket of clamd.

```
// the clamfs_open function
static int clamfs_open(const char *path, struct fuse_file_info *fi)

// 1st step
// Check extension ACL
// excluded from anti-virus scan because extension whitelisted
// forced anti-virus scan because extension blacklisted

// 2nd step
// Check file size
// excluded from anti-virus scan because file is too big

// 3rd step
// Check if file is in cache
if (cache != NULL) // if cache initalized
ret = lstat(real_path.get(), &file_stat); // get file stat

if (!ret) /* got file stat without error */
    if (cache->has(file_stat.st_ino)) // good news, there is a hit - early cache hit
        ptr_val = cache->get(file_stat.st_ino); // check last hit info
        if (ptr_val->scanTimestamp == file_stat.st_mtime) // nothing changed since last scan - late cache hit
            if (ptr_val->isClean)
                return open_backend(path, fi); // clean and open the file
            else
                return -EPERM; // infected
        else // file got changed since last scan - late cache miss
            // scan it again
            scan_result = ClamavScanFile(real_path.get());
            // Check for scan results and update cache
            ptr_val = cache->get(file_stat.st_ino);
            ptr_val->scanTimestamp = file_stat.st_mtime; // log scan timestamp
            if (scan_result == 1) /* virus found */
                ptr_val->isClean = false;
                return -EPERM;
            else if(scan_result == 0) /* clean */
                ptr_val->isClean = true;
                return open_backend(path, fi); // clean and open the file
    else // not hit in current cache - early cache miss
        scan_result = ClamavScanFile(real_path.get());// scan it
        // Check for scan results and add to cache
            if (scan_result == 1) /* virus found */
                cache->add(file_stat.st_ino, result); // add to cache
                return -EPERM;
            else if(scan_result == 0) /* clean */
                cache->add(file_stat.st_ino, result); // add to cache
                return open_backend(path, fi); // clean and open the file

// end of the clamfs_open function
```