

Hard Chatting

Write-up by team in/s/ane for the problem *Hard Chatting* from CSAW HSF 2016

Category: Network Forensics

Point Value: 400

Description:

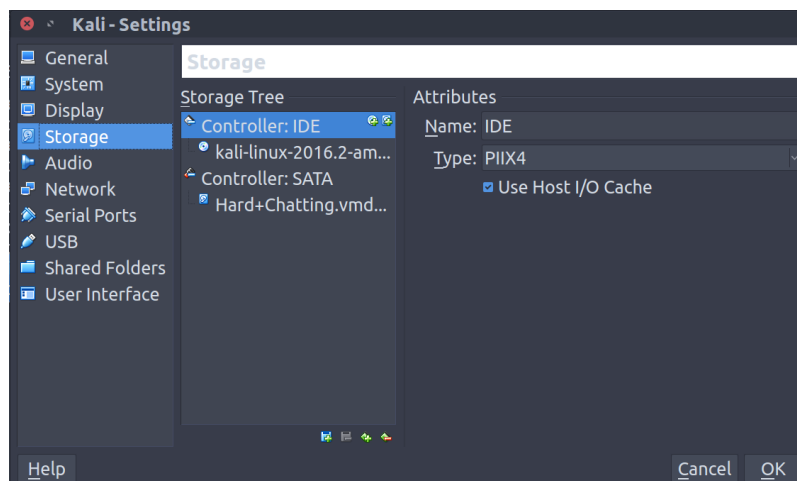
Hey, you're one of those computer people right? My boss Mike down at the garage said he got a letter from the internet company saying something about an attack? Could you take a look? I can't remember the password though...

Link: <https://s3.amazonaws.com/hsf2016/Hard+Chatting.vmdk>
[Video Walkthrough](#)

Setting up:

The problem gives us a virtual machine, but we shouldn't boot into it right away. By booting directly into the vm, we run the risk of running unwanted programs, losing temporary files, and overwriting memory addresses. Since this is a forensics challenge, and we don't know what we'll be doing on the vm, it is important that we maintain forensic integrity as we search for the flag. This is true in real-life investigations, and since this is a competition centered around forensics, we should do the same.

To proceed with our investigation, we'll use [Kali Linux](#), a Linux distribution designed for penetration testing. Conveniently, Kali comes pre-packaged with open source forensic software for us to work with. Using [VirtualBox](#), we can run a Kali vm, and mount the given .vmdk file onto it. We can do this by adding the image as a hard disk, as shown below:



Upon booting into Kali, we are prompted with a menu that looks like the following:



We will use “Forensic mode”, which differs from the normal mode in a couple of ways. The Kali people say this about forensic mode:

1. First, the internal hard disk is never touched. If there is a swap partition it will not be used and no internal disk will be auto mounted. We verified this by first taking a standard system and removing the hard drive. A hash was taken of the drive using a commercial forensic package. We then reattached the drive to the computer and booted Kali Linux “Live” in forensic mode. After using Kali for a period of time, we then shut the system down, removed the hard drive, and took the hash again. These hashes matched, indicating that at no point was anything changed on the drive in any way.
2. The other, equally important, change is that auto-mounting of removable media is **disabled**. USB thumb drives, CDs, and the like will **not** be auto-mounted when inserted. The idea behind this is simple: in forensic mode, **nothing** should happen to **any** media without **direct user action**. Anything that you do as a user is on you.

This behavior is desirable for digital forensics, because it ensures that the filesystem will not be modified without without our explicit permission.

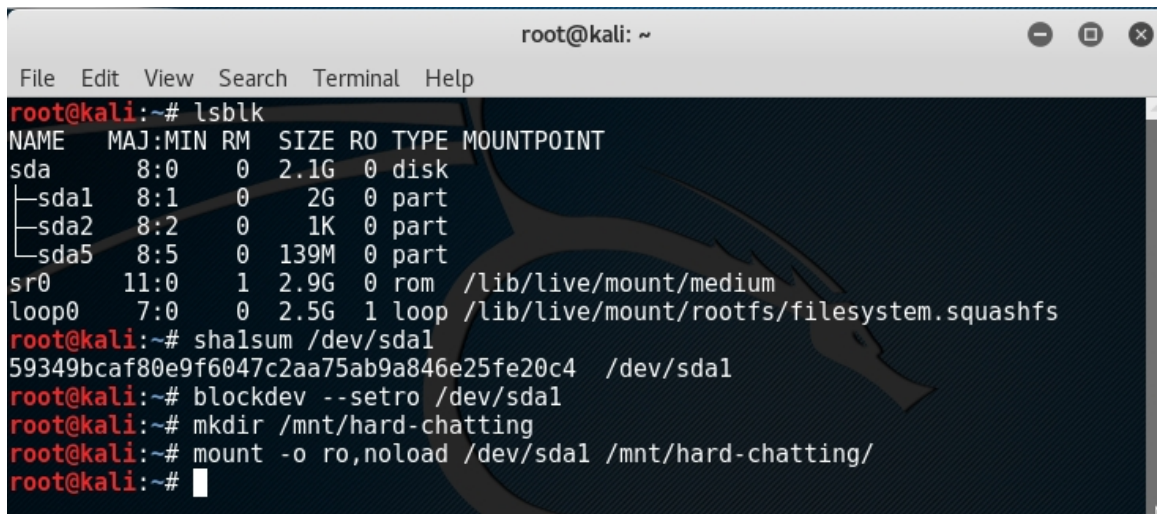
Once we are inside Kali, we can prepare the disk for investigation. Running ``lsblk``, we find that our virtual hard drive is at `/dev/sda1`. To make absolutely sure that we maintain forensic integrity throughout our investigation, we'll compute the SHA1-checksum of the partition, which we'll check again later.

Running ``sha1sum /dev/sda1``, we get our checksum:

“59349bcaf80e9f6047c2aa75ab9a846e25fe20c4”.

Next, we run `blockdev --setro /dev/sda1` to set the disk to read-only mode. We can now mount the virtual disk by running `mkdir /mnt/hard-chatting` and `mount -o ro,noload /dev/sda1 /mnt/hard-chatting`. This mounts the virtual disk to `/mnt/hard-chatting`, and also makes it read-only.

Here's a picture depicting the steps discussed above:

A terminal window titled 'root@kali: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
root@kali:~# lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda   8:0      0  2.1G  0 disk
├─sda1 8:1      0    2G  0 part
├─sda2 8:2      0    1K  0 part
└─sda5 8:5      0  139M  0 part
sr0   11:0     1  2.9G  0 rom  /lib/live/mount/medium
loop0 7:0      0  2.5G  1 loop /lib/live/mount/rootfs/filesystem.squashfs

root@kali:~# shasum /dev/sda1
59349bcdf80e9f6047c2aa75ab9a846e25fe20c4 /dev/sda1

root@kali:~# blockdev --setro /dev/sda1
root@kali:~# mkdir /mnt/hard-chatting
root@kali:~# mount -o ro,noload /dev/sda1 /mnt/hard-chatting/
root@kali:~#
```

Solution:

Looking inside `/home`, we find the user “mike”. Poking around his home directory, we find several files, including `.bash_history`, `.nano_history`, and `.gdb_history`.

The `.bash_history` file stores each command inputted by the user, and could give us a few hints about what exactly the user was doing. Unfortunately, the file is empty.

Further investigation reveals that nano saves search/replace strings in `.nano_history`. Opening it up, we find that it contains the strings “6667”, “avo”, and “avolution”.

```
root@kali: /mnt/hard-chatting/home/mike
File Edit View Search Terminal Help
root@kali:~# mount -o ro,noload /dev/sda1 /mnt/hard-chatting/
root@kali:~# shasum /dev/sda1
59349bc4f80e9f6047c2aa75ab9a846e25fe20c4 /dev/sda1
root@kali:~# cd /mnt/hard-chatting/
root@kali:/mnt/hard-chatting# cd home/mike/
root@kali:/mnt/hard-chatting/home/mike# ls -al
total 68
drwxr-xr-x 3 1000 1000 4096 Sep 25 10:16 .
drwxr-xr-x 3 root root 4096 Sep 25 10:13 ..
-rw-r--r-- 1 1000 1000 0 Sep 25 10:17 .bash_history
-rw-r--r-- 1 1000 1000 220 Sep 25 00:54 .bash_logout
-rw-r--r-- 1 1000 1000 3515 Sep 25 00:54 .bashrc
drwx----- 3 1000 1000 4096 Sep 25 09:39 .config
-rwxr-xr-x 1 1000 1000 34328 Sep 25 10:05 .gdb_history
-rw-r--r-- 1 1000 1000 21 Sep 25 10:17 .nano_history
-rw-r--r-- 1 1000 1000 675 Sep 25 00:54 .profile
-rw-r--r-- 1 1000 1000 66 Sep 25 10:16 .selected_editor
root@kali:/mnt/hard-chatting/home/mike# cat .nano_history
6667
avo
avolution
root@kali:/mnt/hard-chatting/home/mike#
```

Moving on, closer inspection of `.gdb_history` reveals that it has executable permissions (+x), and running `file` confirms that it is indeed an ELF 64 bit executable. This is suspicious, since it's supposed to contain gdb command history.

```
root@kali: /mnt/hard-chatting/home/mike
File Edit View Search Terminal Help
-rwxr-xr-x 1 1000 1000 34328 Sep 25 10:05 .gdb_history
-rw-r--r-- 1 1000 1000 21 Sep 25 10:17 .nano_history
-rw-r--r-- 1 1000 1000 675 Sep 25 00:54 .profile
-rw-r--r-- 1 1000 1000 66 Sep 25 10:16 .selected_editor
root@kali:/mnt/hard-chatting/home/mike# ls -l .gdb_history
-rwxr-xr-x 1 1000 1000 34328 Sep 25 10:05 .gdb_history
root@kali:/mnt/hard-chatting/home/mike# file .gdb_history
.gdb_history: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically l
inked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sh
al]=f8100495a42181f6bf739f4644c46a50f0bddd4, not stripped
root@kali:/mnt/hard-chatting/home/mike# ./gdb_history
root@kali:/mnt/hard-chatting/home/mike# ls -al
total 68
drwxr-xr-x 3 1000 1000 4096 Sep 25 10:16 .
drwxr-xr-x 3 root root 4096 Sep 25 10:13 ..
-rw-r--r-- 1 1000 1000 0 Sep 25 10:17 .bash_history
-rw-r--r-- 1 1000 1000 220 Sep 25 00:54 .bash_logout
-rw-r--r-- 1 1000 1000 3515 Sep 25 00:54 .bashrc
drwx----- 3 1000 1000 4096 Sep 25 09:39 .config
-rwxr-xr-x 1 1000 1000 34328 Sep 25 10:05 .gdb_history
-rw-r--r-- 1 1000 1000 21 Sep 25 10:17 .nano_history
-rw-r--r-- 1 1000 1000 675 Sep 25 00:54 .profile
-rw-r--r-- 1 1000 1000 66 Sep 25 10:16 .selected_editor
root@kali:/mnt/hard-chatting/home/mike#
```

Running `./gdb_history` doesn't reveal anything, so we'll instead run `strings` on it to see if we can glean any new information from the strings within the binary. Looking through the output, we find the string "irc.rogueterminal.com". We've seen the "rogueterminal.com" url before in the *IPMI Server Got Owned* challenge, so we know we're onto something. Thinking back to the contents of `.nano_history`, we realize that Internet Relay Chat (IRC) servers typically run on port 6667, and that we should try connecting.

[illegible]

not done yet, since we need to re-compute the checksum of the new data that we maintained forensic integrity.

```

root@kali: /mnt/hard-chatting/home/mike
File Edit View Search Terminal Help
atoi@@GLIBC_2.2.5
sprintf@@GLIBC_2.2.5
getppid@@GLIBC_2.2.5
exit@@GLIBC_2.2.5
connect@@GLIBC_2.2.5
_TMC_END
disabled
_ITM_registerTMCloneTable
sock
ntohl@@GLIBC_2.2.5
nick
strdup@@GLIBC_2.2.5
sleep@@GLIBC_2.2.5
dispass
getspoofs
_init
fork@@GLIBC_2.2.5
rand@@GLIBC_2.2.5
move
socket@@GLIBC_2.2.5
killall
root@kali:/mnt/hard-chatting/home/mike# shalsum /dev/sda1
59349bcaf80e9f6047c2aa75ab9a846e25fe20c4 /dev/sda1
root@kali:/mnt/hard-chatting/home/mike#

```

The checksums are the same, so we know that we got ourselves a nice 400 points while also keeping our evidence untainted!