Encrypted Disk

CTF: CSAW 2022
Category: Forensics
Author: Declan Oberzan (TNAR5)

Data Discovery

To start the challenge we are given two images:

```
forensic.img
memory.raw
```

This is a forensics challenge so lets begin by determining the type of data we are dealing with. For this we will look for 'magic bytes' that will help discover the structures in the files.

There are some tools that can do this: `file` and `binwalk` are what I ended up using.

```
root@ubuntu:~/edisk# file forensic.img
forensic.img: LUKS encrypted file, ver 1 [aes, ecb, sha1] UUID: 61478c6d-b04e-
4f1a-aff6-b4bb3a29cfbe

root@ubuntu:~/edisk# binwalk memory.raw
memory.raw: data
```

With this information we can tell that the `forensic.img` file is LUKS encrypted filesystem. To decrypt this file we will need a password or the master key.

```
root@ubuntu:~/edisk# cryptsetup open --type luks ./forensic.img forensic
Enter passphrase for ./forensic.img:
No key available with this passphrase.
```

Luckily, for CTF sake it seems we have a memory dump of the computer that is actively mounted to the LUKS filesystem. This is important because in order for the operating system to use the partition it has to concurrently decrypt the data with a master key it will store in memory. We can locate this key by its `magic bytes`, we know from the find command that the LUKS file system is using ver 1 [aes, ecb, sha1] so we will be looking for an AES-128 master key.

```
root@ubuntu:~/edisk# wget --no-check-certificate
https://cfhcable.dl.sourceforge.net/project/findaes/findaes-1.2.zip
root@ubuntu:~/edisk# unzip findaes-1.2.zip
root@ubuntu:~/edisk# cd findaes-1.2/
root@ubuntu:~/edisk/findaes-1.2# make
root@ubuntu:~/edisk/findaes-1.2# ./findaes ../memory.raw
Searching ../memory.raw
Found AES-128 key schedule at offset 0xad9f450:
8d 3f 52 7d e5 14 87 2f 59 59 08 95 8d bc 0e d1
root@ubuntu:~/edisk/findaes-1.2# echo '8d3f527de514872f595908958dbc0ed1' | xxd -r
-p > ../key.raw
```

Now that we have the master key we will attempt to decrypt and mount the filesystem.

```
root@ubuntu:~/edisk# cryptsetup --master-key-file key.raw luksOpen ./forensic.img
forensic-mount
root@ubuntu:~/edisk# mount /dev/mapper/forensic-mount /mnt/
root@ubuntu:~/edisk# cd /mnt/
root@ubuntu:/mnt# ls
dir2  lost+found
root@ubuntu:/mnt# cd dir2/
root@ubuntu:/mnt/dir2# ls
end.png  findme.txt.gpg  readme.txt
```

It worked!

Now we get to do some more forensics. Lets take an unencrypted copy of this partition using `dd`.

```
root@ubuntu:~/edisk# dd if=/dev/mapper/forensic-mount of=dec_forensic.img
49152+0 records in
49152+0 records out
25165824 bytes (25 MB, 24 MiB) copied, 0.0772391 s, 326 MB/s
```

I will admit there being just a dir2 in the filesystem is a little suspicious. Lets check out the files to see what everything is.

```
root@ubuntu:/mnt/dir2# ls
end.png  findme.txt.gpg  readme.txt
root@ubuntu:/mnt/dir2# file *
end.png:        PNG image data, 850 x 300, 8-bit/color RGB, non-interlaced
findme.txt.gpg: GPG symmetrically encrypted data (AES cipher)
readme.txt:     ASCII text
```

We aren't doing stenography today so let's see what that png is hiding and turn it into something useful.

```
root@ubuntu:/mnt/dir2# binwalk end.png -e

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 850 x 300, 8-bit/color RGB, non-
interlaced
320           0x140           Zlib compressed data, best compression
917           0x395           Zlib compressed data, best compression
493886        0x7893E         Zip archive data, at least v2.0 to extract,
compressed size: 61917, uncompressed size: 61907, name: end.zip.gpg
555953        0x87BB1         End of Zip archive

root@ubuntu:/mnt/dir2# cd _end.png.extracted/
root@ubuntu:/mnt/dir2/_end.png.extracted# ls
140  140.zlib  395  395.zlib  7893E.zip  end.zip.gpg
root@ubuntu:/mnt/dir2/_end.png.extracted# cp end.zip.gpg ~/edisk/
root@ubuntu:/mnt/dir2/_end.png.extracted# cd ~/edisk
root@ubuntu:~/edisk# file end.zip.gpg
end.zip.gpg: GPG symmetrically encrypted data (CAST5 cipher)
```

So it seems as though we have `2` GPG encrypted blobs one `(CAST5 cipher)` and one `(AES cipher)`. Both will need a password to decrypt. So lets do some more digging to find how these were encrypted. Lets do some volatility magic. First identify the memory image operating system and kernel. (We will need this info to determine that we will need to build a custom volatility profile)

```
root@ubuntu:~/edisk# strings memory.raw| grep -i 'Linux version' | uniq
Linux version 4.4.0-72-lowlatency (buildd@lcy01-17) (gcc version 5.4.0 20160609
(Ubuntu 5.4.0-6ubuntu1~16.04.4) ) #93-Ubuntu SMP PREEMPT Fri Mar 31 15:25:21 UTC
2017 (Ubuntu 4.4.0-72.93-lowlatency 4.4.49)
 o  The intent is to make the tool independent of Linux version dependencies,
```

This output tells us that the kernel version is `4.4.0-72.93-lowlatency 4.4.49` and the OS is `Ubuntu 16.04.4`, this is not a supported profile in volatility 2 so we will have to build a custom one. This will require gathering headers and symbols from a similar machine running the same kernel and operating system.

SEPARATE UBUNTU 16.04.4 VM - Install kernel and headers then reboot to apply.

```
root@ubuntu:~/# sudo apt install linux-image-4.4.0-72-lowlatency linux-headers-
4.4.0-72-lowlatency
root@ubuntu:~/# sudo apt install build-essential dwarfdump
root@ubuntu:~/# reboot

# Make the Ubuntu custom profile in the machine using volatility and dwarfdump
root@ubuntu:~/# git clone --depth=1
https://github.com/volatilityfoundation/volatility
root@ubuntu:~/# cd volatility/tools/linux
root@ubuntu:~/volatility/tools/linux# make
root@ubuntu:~/volatility/tools/linux# sudo zip Ubuntu1604.zip module.dwarf
/boot/System.map-4.4.0-72-lowlatency
```

You can copy this Ubuntu1604.zip to the `volatility/plugins/overlays/linux` folder on your ctf machine now. Let's dump the bash history of the machine from the memory dump.

```
root@ubuntu:~/volatility# python vol.py -f ../memory.raw --
profile=LinuxUbuntu1604x64 linux_bash
Volatility Foundation Volatility Framework 2.6.1
Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
    1229 bash                 2017-04-14 07:58:36 UTC+0000   history
    1229 bash                 2017-04-14 07:58:36 UTC+0000   apt-get install
linux-image-4.4.0-72-lowlatency linux-headers-lowlatency
    1229 bash                 2017-04-14 07:58:36 UTC+0000   reboot
    1229 bash                 2017-04-14 07:58:36 UTC+0000   apt-get insta
    1229 bash                 2017-04-14 07:59:07 UTC+0000   history
    1229 bash                 2017-05-05 12:04:44 UTC+0000   apt-get install lynx
gnupg
    1229 bash                 2017-05-05 12:06:54 UTC+0000   nano /etc/fstab
    1229 bash                 2017-05-05 12:06:58 UTC+0000   nano /etc/crypttab
    1229 bash                 2017-05-05 12:07:08 UTC+0000   cd /mnt/
    1229 bash                 2017-05-05 12:07:29 UTC+0000   cp -R
/media/sf_DUMP/dir* .
    1229 bash                 2017-05-05 12:07:38 UTC+0000   ping 8.8.8.8
    1229 bash                 2017-05-05 12:09:14 UTC+0000   gpg --quick-gen-key
'Troll <abuse@nothere.com>' rsa4096 cert 1y
    1229 bash                 2017-05-05 12:09:49 UTC+0000   lynx -
accept_all_cookies "https://www.google.com/?=password+porno+collection"
    1229 bash                 2017-05-05 12:10:27 UTC+0000   gpg --yes --batch --
passphrase=1m_4n_4dul7_n0w -c findme.txt
    1229 bash                 2017-05-05 12:10:37 UTC+0000   lynx -
accept_all_cookies "https://www.google.com/?=password+troll+memes"
    1229 bash                 2017-05-05 12:11:04 UTC+0000   gpg --yes --batch --
passphrase=Troll_Tr0ll_TrOll -c end.zip
    1229 bash                 2017-05-05 12:11:20 UTC+0000   nano
dir1/dic_fr_l33t.txt
    1229 bash                 2017-05-05 12:11:28 UTC+0000   rm findme.txt
    1229 bash                 2017-05-05 12:11:35 UTC+0000   rm -rf dir1/
    1229 bash                 2017-05-05 12:11:55 UTC+0000   dd if=/dev/sdb
of=/media/sf_DUMP/forensic.img bs=2048
```

Here we can see the original password in plaintext for both encrypted files `1m_4n_4dul7_n0w` and `Troll_Tr0ll_TrOll`. Now decrypt them:

```
root@ubuntu:~/edisk# cp /mnt/dir2/findme.txt.gpg .
root@ubuntu:~/edisk# gpg --yes --batch --decrypt --passphrase=1m_4n_4dul7_n0w
findme.txt.gpg
gpg: AES encrypted data
gpg: gpg-agent is not available in this session
gpg: encrypted with 1 passphrase
The flag is not here of course !!!
You must find it :-)
Troll one day troll always ........
root@ubuntu:~/edisk# gpg --yes --batch --decrypt --passphrase=Troll_Tr0ll_TrOll
end.zip.gpg > end.zip
gpg: CAST5 encrypted data
gpg: gpg-agent is not available in this session
gpg: encrypted with 1 passphrase
gpg: WARNING: message was not integrity protected
root@ubuntu:~/edisk# file end.zip
end.zip: Zip archive data, at least v2.0 to extract
root@ubuntu:~/edisk# unzip end.zip
Archive:  end.zip
[end.zip] flag.gif password:
```

This challenge seems to never end :\ lets look for a password I guess. Looking at the command history it looks like maybe dir1 has it in dir1/dic_fr_l33t.txt?

```
root@ubuntu:~/volatility# python vol.py -f ../edisk/memory.raw --
profile=LinuxUbuntu1604x64 linux_find_file -L | grep dic_fr_l33t.txt
Volatility Foundation Volatility Framework 2.6.1
               3 0xffff88000c135cd8 /media/sf_DUMP/dir1/dic_fr_l33t.txt
```

Looking through a hex editor we can find that the file starts at `0x200400` and ends at `0x5102EB` (difference `0x30FEEB`). Extract it:

```
root@ubuntu:~/edisk# dd if=dec_forensic.img of=dic_fr_l33t.txt skip=$((0x200400))
count=$((0x30FEEB)) bs=1
3210987+0 records in
3210987+0 records out
3210987 bytes (3.2 MB, 3.1 MiB) copied, 2.83577 s, 1.1 MB/s
```

Time to crack the zip. We will use `john` and `zip2john` for this.

```
root@ubuntu:~/john/run# ./zip2john ../../edisk/end.zip > ../../edisk/endzip.hash
ver 2.0 efh 5455 efh 7875 end.zip/flag.gif PKZIP Encr: TS_chk, cmplen=61563,
decmplen=90082, crc=18A1C7A3 ts=8050 cs=8050 type=8

root@ubuntu:~/john/run# ./john ../../edisk/endzip.hash --
wordlist=../../edisk/dic_fr_l33t.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Cyb3rs3curit3    (end.zip/flag.gif)
1g 0:00:00:00 DONE (2022-09-11 13:36) 100.0g/s 5734Kp/s 5734Kc/s 5734KC/s
Couronnassi3z..D3activons
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

# Using the password Cyb3rs3curit3
root@ubuntu:~/john/run# cd ~/edisk/
root@ubuntu:~/edisk# unzip end.zip
Archive:  end.zip
[end.zip] flag.gif password:
  inflating: flag.gif
```

Imma be honest with you chief I was so tired by this point that I just extracted all the frames from the gif and scanned each to get the characters for the flag.

DATA: The_flag_is:1_Lik3_F0r3nS1c_4nd_y0u?
FLAG: 1_Lik3_F0r3nS1c_4nd_y0u?

THE END