
CSC153: Lab 1 - Forensic Investigation on Your Own Personal Computer

Ryan Kozak



2019-10-06

Lab 1: Forensic Investigation on Your Own Personal Computer

CSC 153 - Computer Forensics Principles and Practice

Introduction

For this lab we were to conduct a forensics investigation on our own PC, using any case management tools we so choose. Options presented to us included OSForensics, FTK, Sleuthkit/Autopsy_Browser, Encase, and ProDiscover Basic. I've used Sleuthkit and the Autopsy browser, only because I do not have a Windows machine on which to run any of the other software.

Note

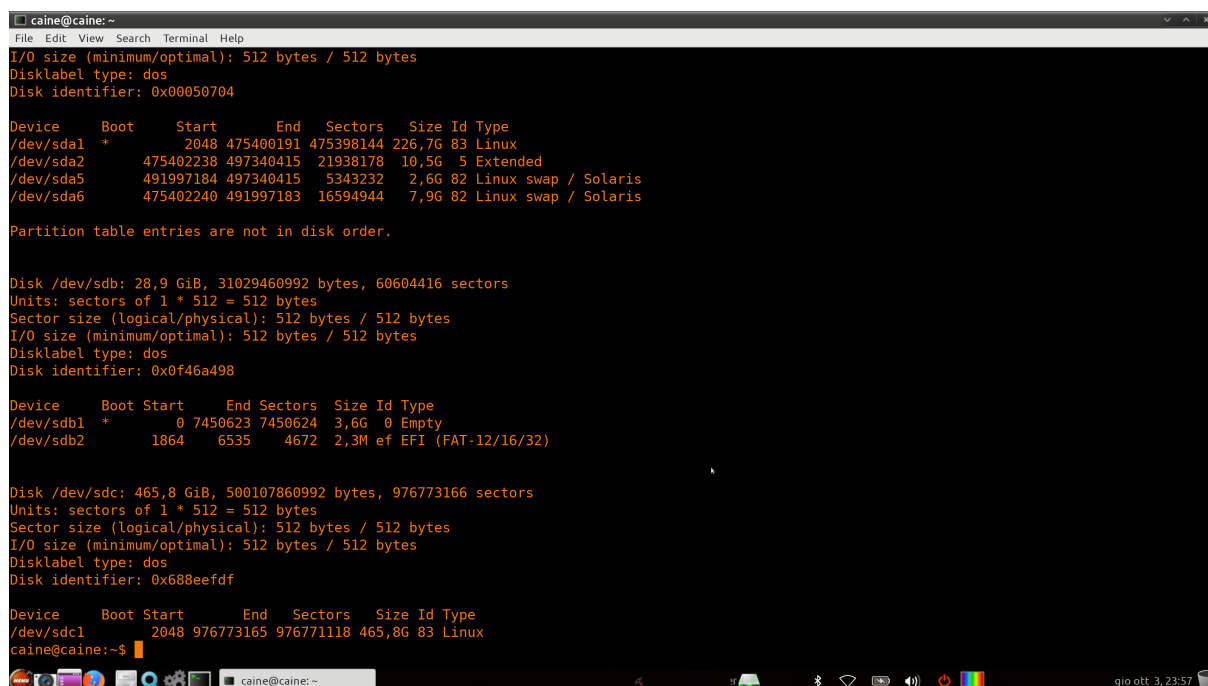
This lab took me **significantly** longer than it was intended to. Largely, this was because I have encrypted installations of Ubuntu Linux on all my machines, and I couldn't just install OSForensics and run a live acquisition like I'm assuming 99% of people did for this lab.

Data Acquisition

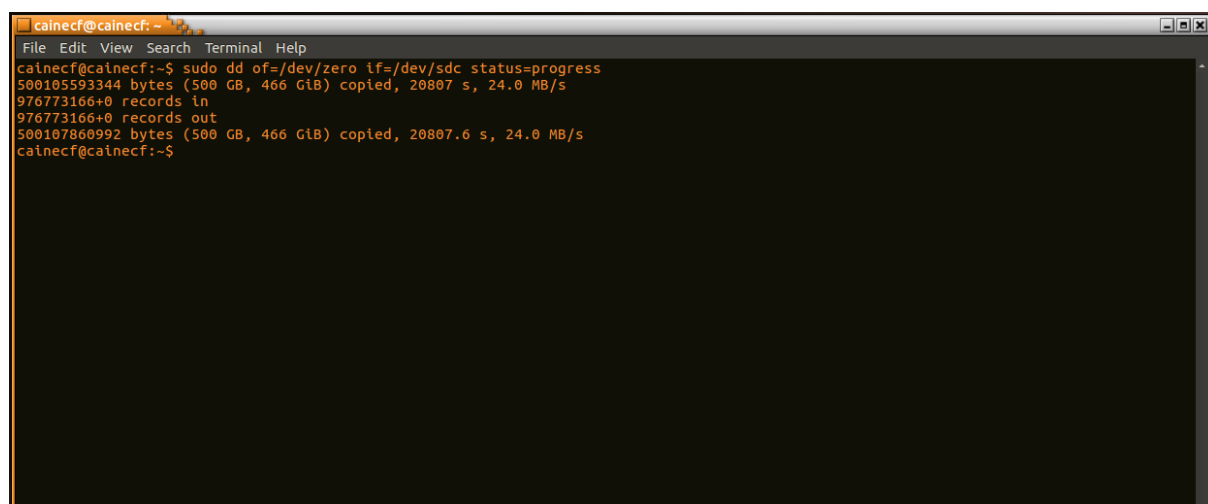
In preparation for this Lab I've installed CAIN 10.0 onto a flash drive, and this is the drive from which I ran the lab.

Preparing the Evidence Drive

The computer I'll be investigating for this lab is my laptop, which has a 256GB SSD with Ubuntu 16.04 installed. So, the first thing to do was prepare my evidence drive, which is a 500GB hard disk, large enough to fit an image of my laptop's entire drive. The drive on my laptop is `/dev/sda`, the CAIN live USB is `/dev/sdb`, and the evidence drive is `/dev/sdc`.



```
caine@caine:~  
File Edit View Search Terminal Help  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0x00050704  
  
Device      Boot      Start          End      Sectors   Size Id Type  
/dev/sda1   *            2048    475400191  475398144  226,7G 83 Linux  
/dev/sda2             475402238  497340415   21938178    10,5G  5 Extended  
/dev/sda5             491997184  497340415    5343232    2,6G 82 Linux swap / Solaris  
/dev/sda6             475402240  491997183   16594944    7,9G 82 Linux swap / Solaris  
  
Partition table entries are not in disk order.  
  
Disk /dev/sdb: 28,9 GiB, 31029460992 bytes, 60604416 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0x0f46a498  
  
Device      Boot Start          End      Sectors   Size Id Type  
/dev/sdb1   *            0 7450623 7450624    3,6G  0 Empty  
/dev/sdb2             1864      6535      4672    2,3M ef EFI (FAT-12/16/32)  
  
Disk /dev/sdc: 465,8 GiB, 500107860992 bytes, 976773166 sectors  
Units: sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disklabel type: dos  
Disk identifier: 0x688eefdf  
  
Device      Boot Start          End      Sectors   Size Id Type  
/dev/sdc1   *            2048  976773165  976771118  465,8G 83 Linux  
caine@caine:~$
```

Figure 1: Output of disks on machine.

```
cainecf@cainecf:~  
File Edit View Search Terminal Help  
cainecf@cainecf:~$ sudo dd of=/dev/zero if=/dev/sdc status=progress  
500105593344 bytes (500 GB, 466 GiB) copied, 20807 s, 24.0 MB/s  
976773166+0 records in  
976773166+0 records out  
500107860992 bytes (500 GB, 466 GiB) copied, 20807.6 s, 24.0 MB/s  
cainecf@cainecf:~$
```

Figure 2: Evidence drive zeroed out.

```
caine@caine: ~  
File Edit View Search Terminal Help  
caine@caine:~$ sudo fdisk /dev/sdc  
  
Welcome to fdisk (util-linux 2.31.1).  
Changes will remain in memory only, until you decide to write them.  
Be careful before using the write command.  
  
Command (m for help): n  
Partition type  
   p   primary (0 primary, 0 extended, 4 free)  
   e   extended (container for logical partitions)  
Select (default p): p  
Partition number (1-4, default 1): 1  
First sector (2048-976773165, default 2048):  
Last sector, +sectors or +size{K,M,G,T,P} (2048-976773165, default 976773165):  
  
Created a new partition 1 of type 'Linux' and of size 465,8 GiB.  
  
Command (m for help): w  
The partition table has been altered.  
Calling ioctl() to re-read partition table.  
Syncing disks.  
  
caine@caine:~$
```

Figure 3: Creating Linux partition on evidence drive.

```
caine@caine:~$ sudo mkfs.ext4 /dev/sdc1  
mke2fs 1.44.1 (24-Mar-2018)  
Creating filesystem with 122096389 4k blocks and 30531584 inodes  
Filesystem UUID: c1270a22-7568-4c70-b7b5-f35c8512e978  
Superblock backups stored on blocks:  
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,  
    4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,  
    102400000  
  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (262144 blocks): done  
Writing superblocks and filesystem accounting information: done  
  
caine@caine:~$
```

Figure 4: Creating Linux ext4 file system on newly created partition.

Creating an Image

After the evidence drive was ready I mounted it at `/mnt/sdc1`, created a new directory called `case1` and generated a pre-image hash of my laptop's drive. *mount_and_hash*

Figure 5: Mounting evidence drive and generating hash of target drive into case directory.

Next I created an image of my laptop's drive, which was also saved into the `case1` directory.

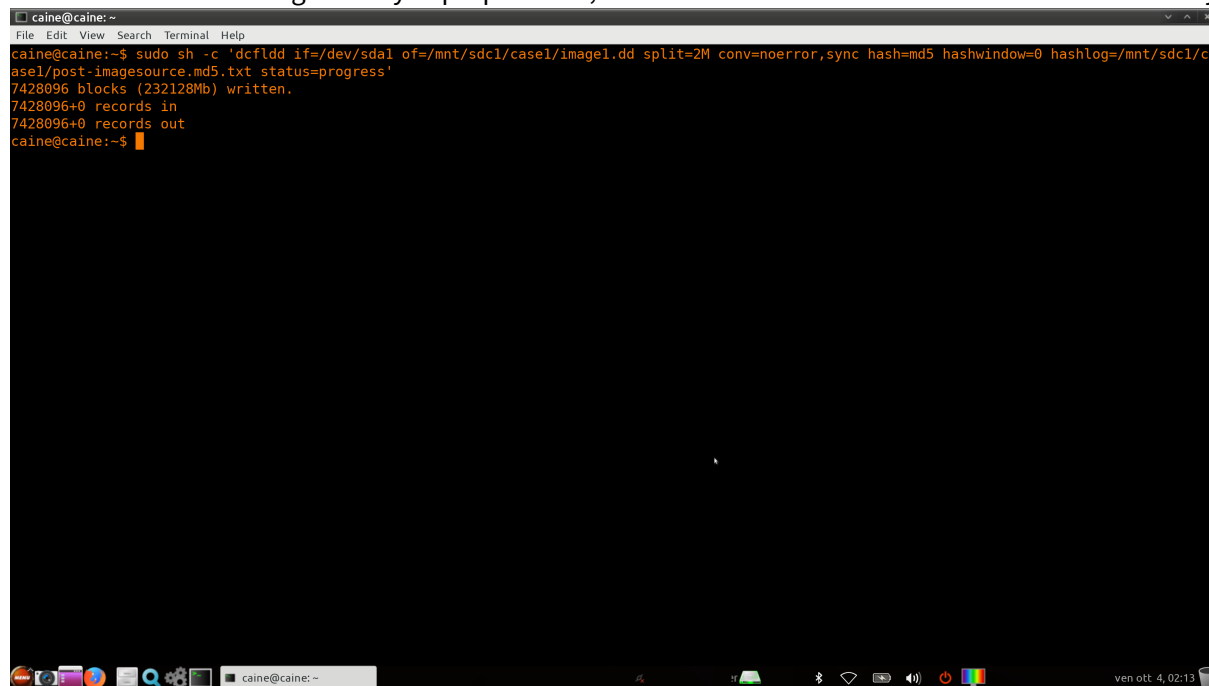
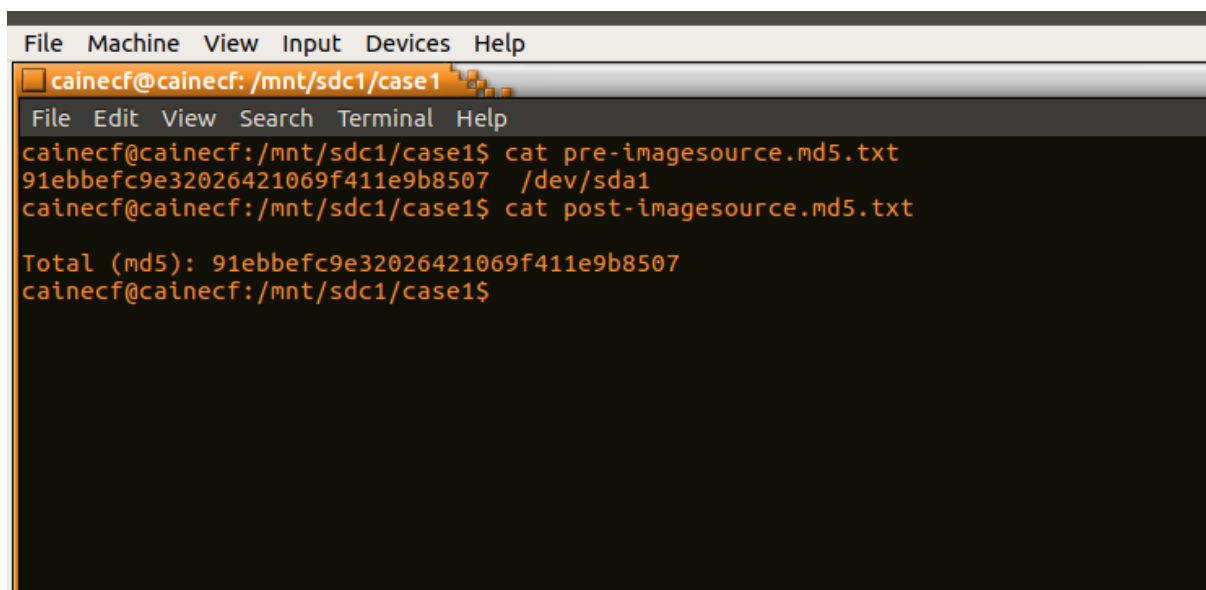
A screenshot of a terminal window titled 'caine@caine: ~'. The terminal shows the execution of a command to create a disk image using 'dcflddd'. The command is: 'sudo sh -c 'dcflddd if=/dev/sda1 of=/mnt/sdc1/case1/image1.dd split=2M conv=noerror,sync hash=md5 hashwindow=0 hashlog=/mnt/sdc1/case1/post-imagesource.md5.txt status=progress''. The output shows '7428096 blocks (232128Mb) written.' and '7428096+0 records in' followed by '7428096+0 records out'. The prompt returns to 'caine@caine:~\$'. The terminal window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The bottom of the window shows a taskbar with various icons and a system tray with the date and time 'ven ott 4, 02:13'.

Figure 6: Creating an image of the laptop's disk, placing it in `case1` directory.

Once the image of the disk was completed, I verified the integrity by comparing the md5 pre and post image hashes.



```
File Machine View Input Devices Help
caine cf@caine cf: /mnt/sdc1/case1
File Edit View Search Terminal Help
caine cf@caine cf: /mnt/sdc1/case1$ cat pre-imagesource.md5.txt
91ebbf9c9e32026421069f411e9b8507 /dev/sda1
caine cf@caine cf: /mnt/sdc1/case1$ cat post-imagesource.md5.txt

Total (md5): 91ebbf9c9e32026421069f411e9b8507
caine cf@caine cf: /mnt/sdc1/case1$
```

Figure 7: Integrity of image verified using md5.

Forensic Investigation

Analyzing the Disk Image

The first step to the investigation was opening Autopsy and creating a new case.

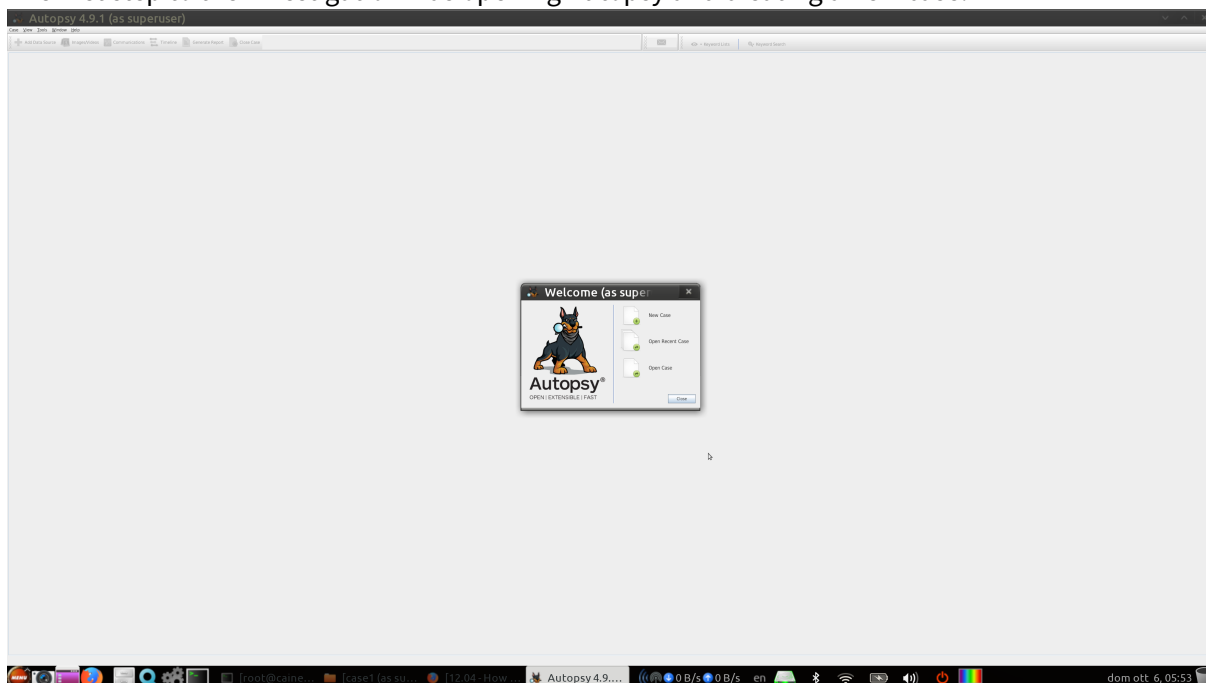


Figure 8: Opening Autopsy and creating new case.

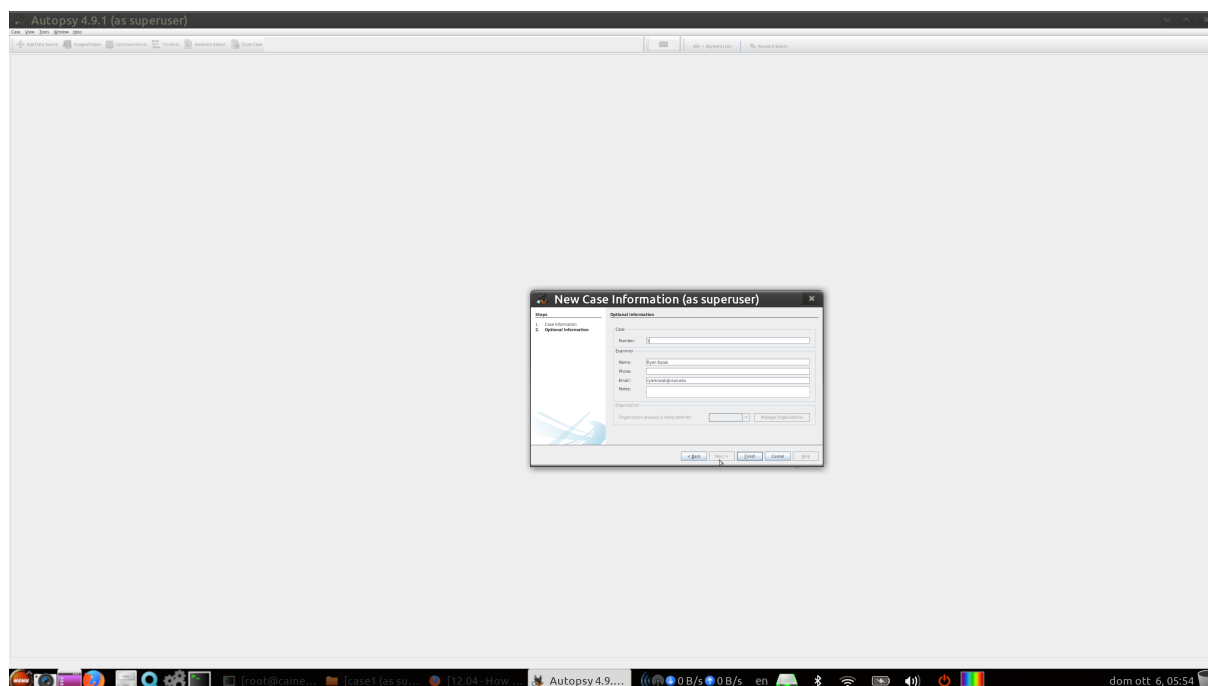


Figure 9: One of the basic info pages to be filled out creating the case.

Once the case was created, it was time to add a data source. Initially the image taken of the drive was added and analyzed.

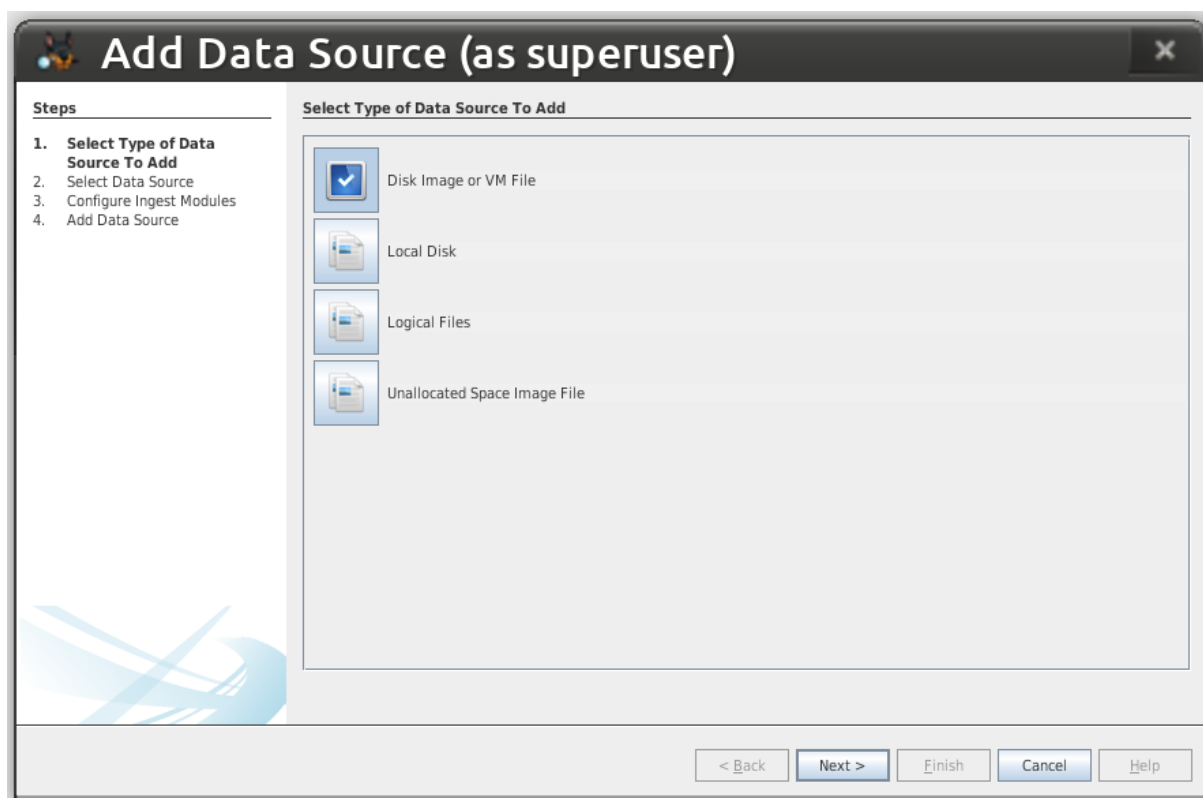


Figure 10: Adding data source type of “Disk Image”.

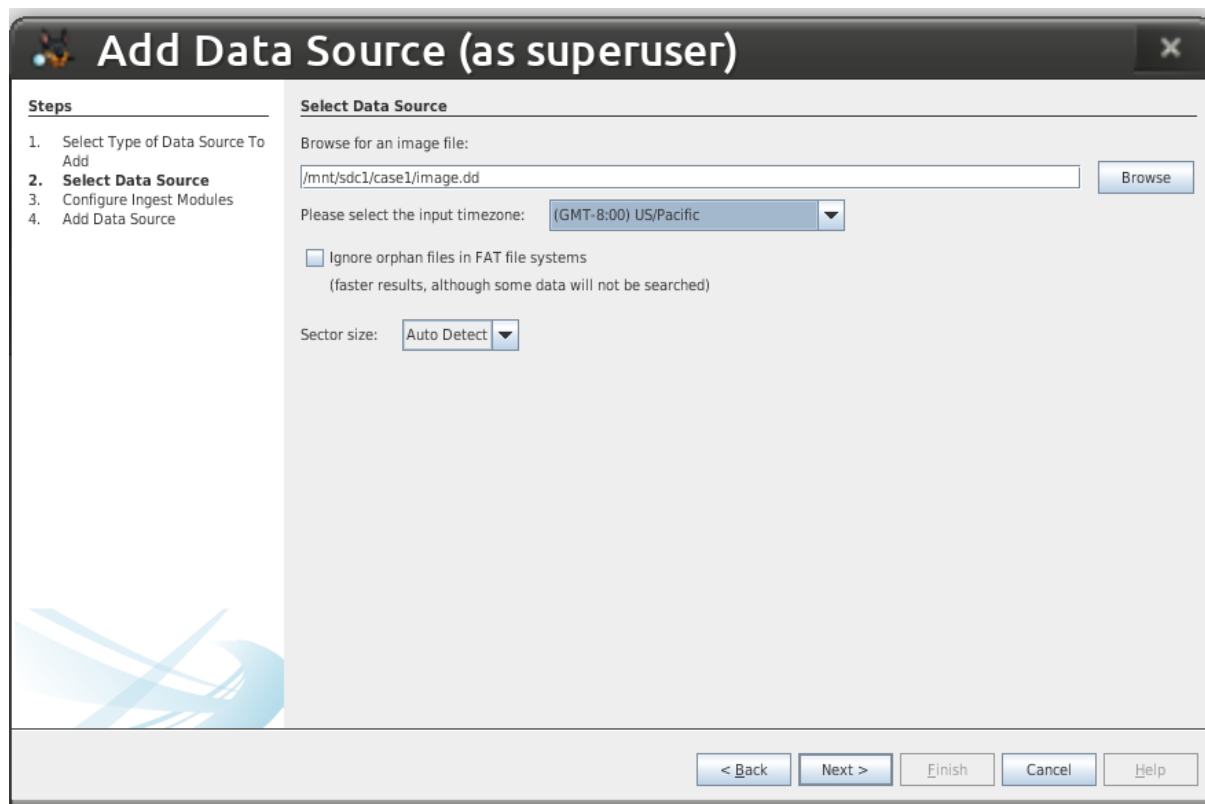


Figure 11: Adding `image.dd` as the disk image.

When the image is being analyzed we can see that it's going to be an encrypted file system (see figure 12).

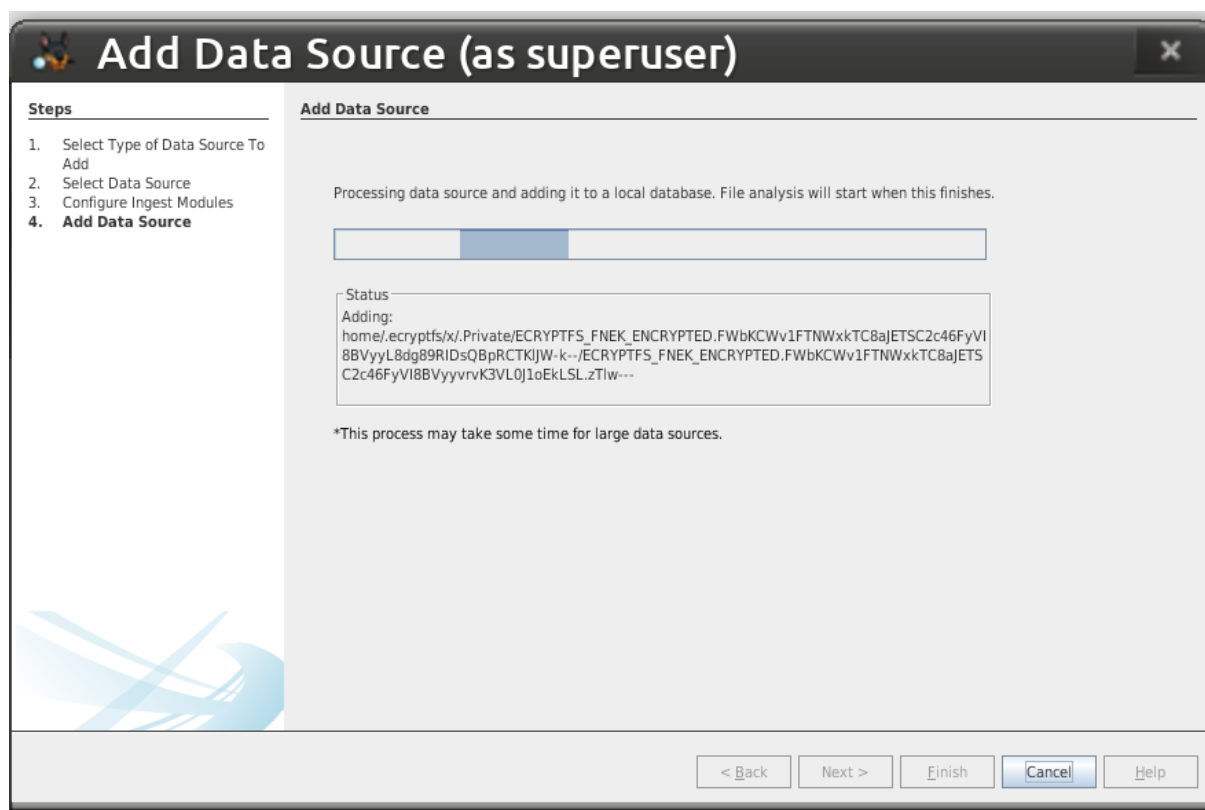


Figure 12: Encrypted file system on `image.dd`.

Now at this point I ran into a big issue. There was almost no useful information apparent. We need to decrypt the image, see below.

Mounting and Decrypting the Disk Image

In order to decrypt the disk image we first mount it under `/mnt/sda1`. Once the was mounted we're able to navigate it as though it's another disk on the system. I navigated to the encrypted home direc-

```
root@caine: /mnt/sda1/home/.ecryptfs/x
File Edit View Search Terminal Help
root@caine:/mnt/sda1# cd ./home
root@caine:/mnt/sda1/home# ls
tcp-ethereal-trace-1.pdf x
root@caine:/mnt/sda1/home# ls -lah
total 40K
drwxr-xr-x  4 root root 4,0K mag  4 2018 .
drwxr-xr-x 25 root root 4,0K ott  1 15:46 ..
drwxrwxr-x  3 root root 4,0K ott 15 2015 .ecryptfs
-rw-r--r--  1 root root 24K mag  4 2018 tcp-ethereal-trace-1.pdf
dr-x----- 2 1000 1000 4,0K ott  5 21:05 x
root@caine:/mnt/sda1/home# cd ../.ecryptfs
root@caine:/mnt/sda1/home/.ecryptfs# ls -lah
total 12K
drwxrwxr-x 3 root root 4,0K ott 15 2015 .
drwxr-xr-x 4 root root 4,0K mag  4 2018 ..
drwxrwxr-x 4 1000 1000 4,0K ott 15 2015 x
root@caine:/mnt/sda1/home/.ecryptfs# cd ../x
root@caine:/mnt/sda1/home/.ecryptfs/x# ls -lah
total 44K
drwxrwxr-x  4 1000 1000 4,0K ott 15 2015 .
drwxrwxr-x  3 root root 4,0K ott 15 2015 ..
drwx-----  2 1000 1000 4,0K ago  8 2016 .ecryptfs
drwx----- 136 1000 1000 28K ott  5 06:07 .Private
root@caine:/mnt/sda1/home/.ecryptfs/x#
```

tory to begin the decryption process.

Figure 13: Encrypted directory.

Note: If I did not have my own 32 character decryption key the rest of this lab would end here, and there would be no way to analyze anything. This is the point at which police typically need to forcibly get the key out of the suspect (beat them, threaten them, whatever).

```
root@caine:/mnt/sdc1/case1# cd /mnt/sda1/home/.ecryptfs/x
root@caine:/mnt/sda1/home/.ecryptfs/x# ls -lah
total 44K
drwxrwxr-x  4 1000 1000 4,0K ott 15  2015 .
drwxrwxr-x  3 root root 4,0K ott 15  2015 ..
drwx-----  2 1000 1000 4,0K ago  8  2016 .ecryptfs
drwx----- 136 root root 28K ott  5 06:07 .Private
root@caine:/mnt/sda1/home/.ecryptfs/x# ecryptfs-recover-private --rw ../.Private
INFO: Found [../.Private].
Try to recover this directory? [Y/n]: y
INFO: Found your wrapped-passphrase
Do you know your LOGIN passphrase? [Y/n] n
INFO: To recover this directory, you MUST have your original MOUNT passphrase.
INFO: When you first setup your encrypted private directory, you were told to record
INFO: your MOUNT passphrase.
INFO: It should be 32 characters long, consisting of [0-9] and [a-f].

Enter your MOUNT passphrase:
INFO: Success! Private data mounted at [/tmp/ecryptfs.gVp3gNa0].
root@caine:/mnt/sda1/home/.ecryptfs/x#
```

Figure 14: File system decryption using key.

After this step my home folder was mounted to `/tmp/ecryptfs.gVp3gNa0`. This was then added as a new Data Source to Autopsy for analysis.

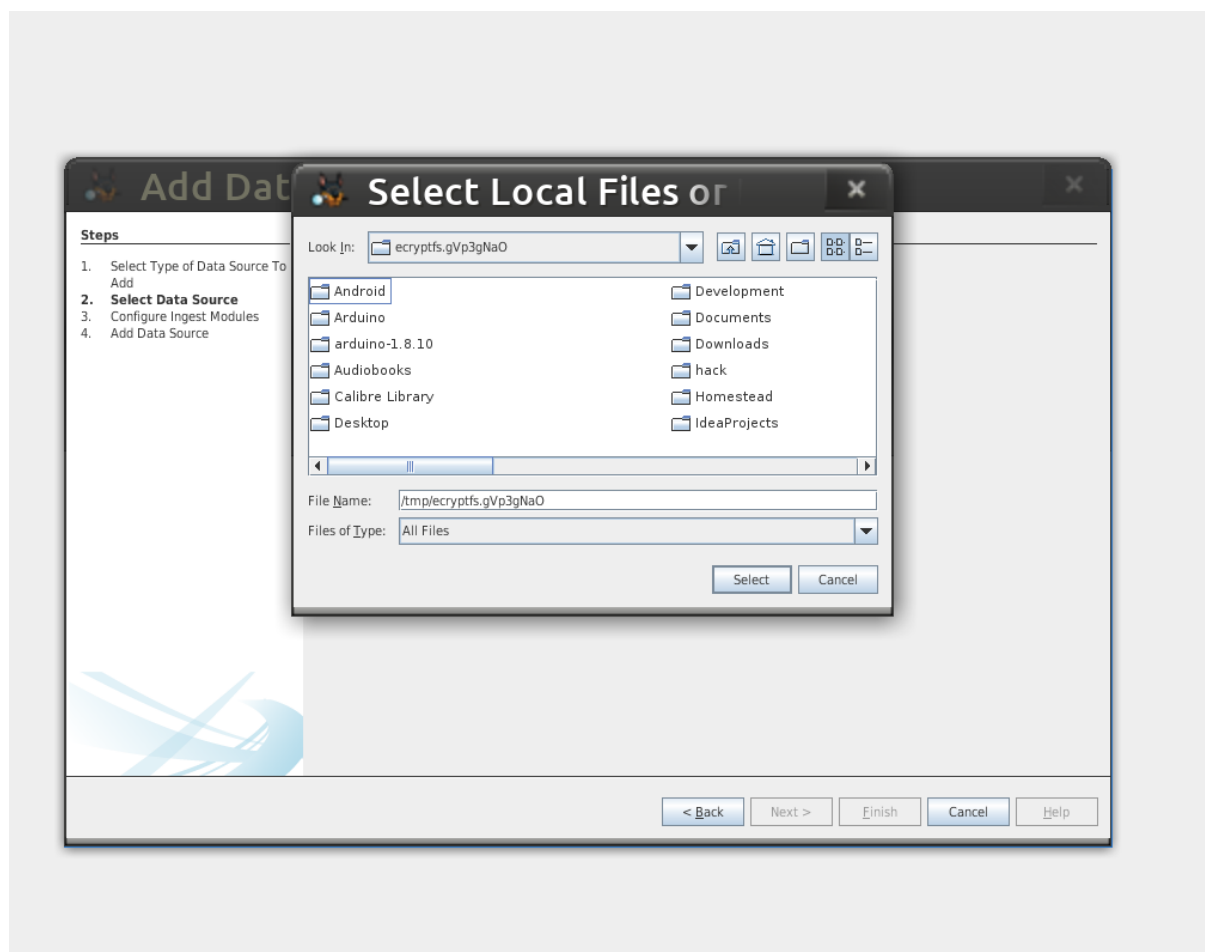


Figure 15: Adding the decrypted directory to as the new data source.

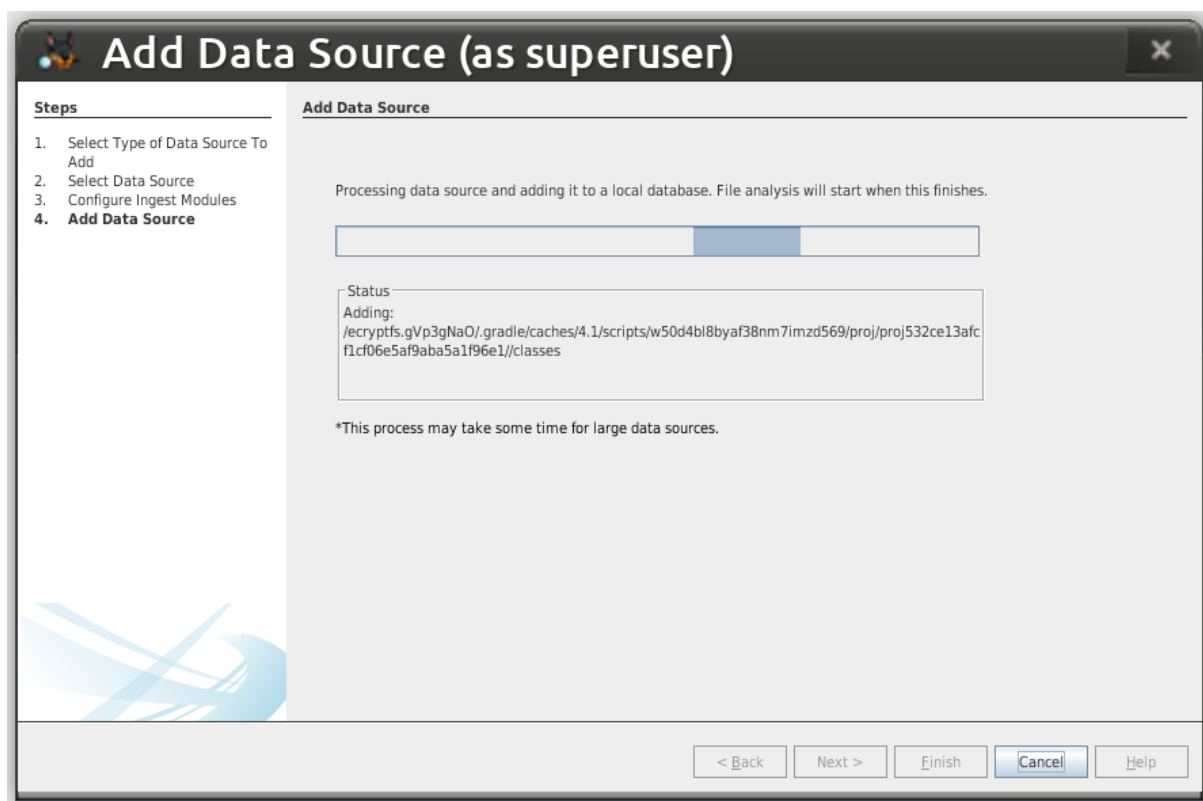


Figure 16: Autopsy going through the decrypted files for analysis, `.gradle` directory being parsed in image.

After this ran for a few hours (more like 9 or 10) it finished. Below are the results.

Investigation Results

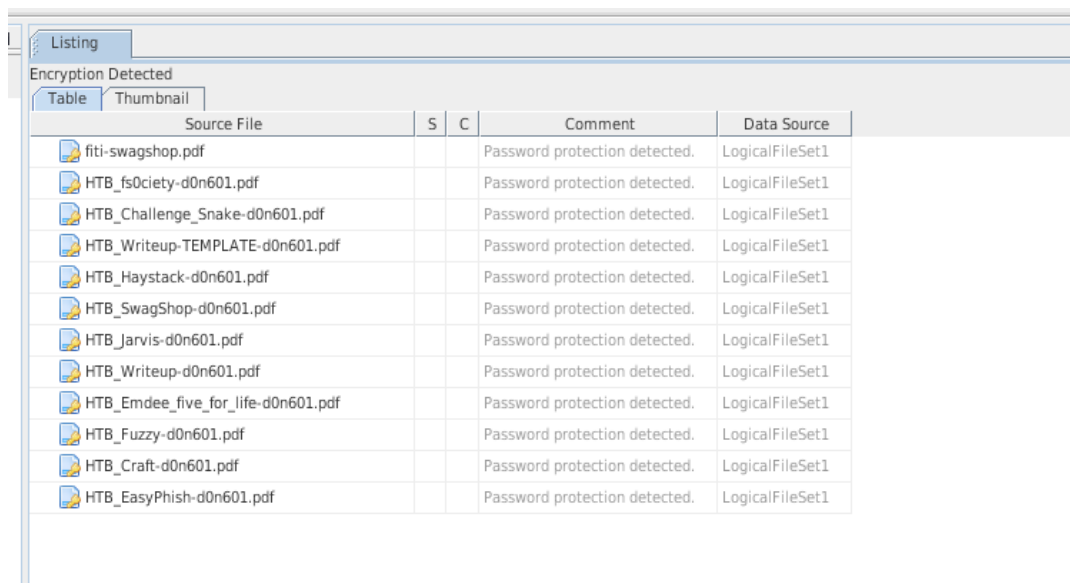
With my setup, some of these bullet points are simply not possible to determine.

1. Provide screenshots of the following information
 - Number and type of documents (Word, Power Point, Excel, etc).
 - The total number of documents is **957**. This was determined by opening up the file tree, and selecting the "Documents" option. As you can see in figure 17 the breakdown goes 190 HTML documents, 124 office documents, 211 PDFs, 426 plain text files, and 6 rich text files.



- [illegible]

- Number and types of encrypted files.
- Autopsy only detected **12** encrypted files on my machine, which I know as a fact is wrong. These are PDF's for CTF writeups that I've password protected using *PDFtk*. It considered my *TrueCrypt* container to be “suspected as encrypted”, which is funny. It also totally missed my KeePassX database.

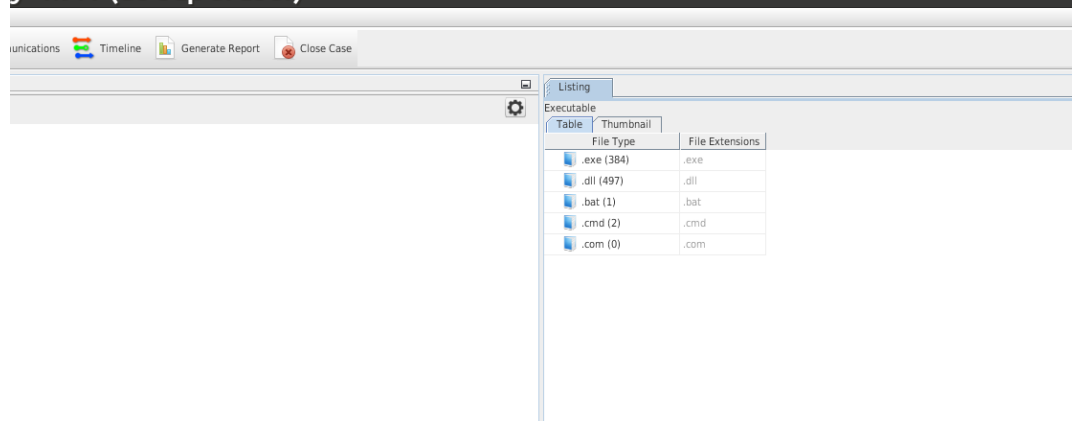


The screenshot shows the Autopsy interface with the 'Listing' tab selected. A table titled 'Encryption Detected' displays a list of PDF files. Each row includes a file icon, the source file name, status (S), comment (C), and data source.

Source File	S	C	Comment	Data Source
fiti-swagshop.pdf			Password protection detected.	LogicalFileSet1
HTB_fs0ciety-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Challenge_Snake-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Writeup-TEMPLATE-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Haystack-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_SwagShop-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Jarvis-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Writeup-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Emdex_five_for_life-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Fuzzy-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_Craft-d0n601.pdf			Password protection detected.	LogicalFileSet1
HTB_EasyPhish-d0n601.pdf			Password protection detected.	LogicalFileSet1

Figure 19: Autopsy only found my password protected PDFs, not all the encrypted files :)

- Number of executable files.
- There were **884** executable files. Most of these are Windows files. I'd assume this is because it scanned Wine, or my Virtual Machines? It's breaking it down by file extension and Linux executables mostly have no extension, perhaps that's another reason?
y 4.9.1 (as superuser)



The screenshot shows the Autopsy interface with the 'Listing' tab selected. A table titled 'Executable' displays a list of file types and their extensions.

File Type	File Extensions
.exe (384)	.exe
.dll (497)	.dll
.bat (1)	.bat
.cmd (2)	.cmd
.com (0)	.com

Figure 20: Only 3 deleted files were found?

- Number of deleted files.
- Autopsy only detected 3 deleted files. I'm not sure if this is because of the SSD, or due to encryption, or some flaw in the Autopsy scan. This number seems low.

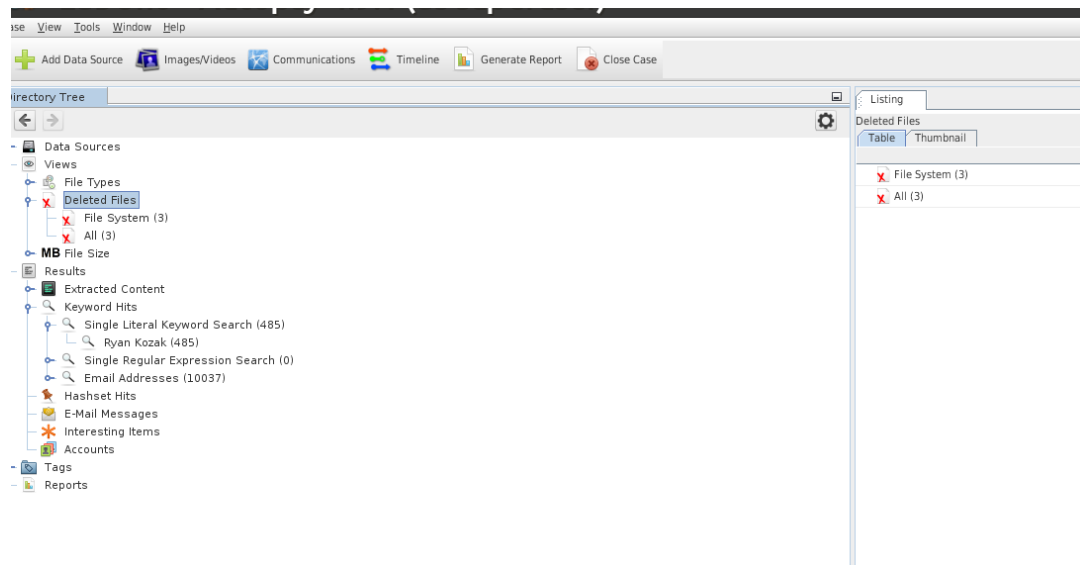


Figure 21: Only 3 deleted files were found?

- Number of files in slack space
- This cannot be determined. In order to analyze the encrypted home directory we must first decrypt it, and then add it as a “Logical File” source. According to Autopsy’s documentation logical file sources cannot find files in unallocated space, see quote and reference below.

It will not look at unallocated space or deleted files. Autopsy will only be able to see the allocated files. You should add the device as a “Logical Drive” to analyze the unallocated space. **User Docs 4.13.**

- Size of unallocated space.
- This cannot be determined for the same reason as stated above.
- Top 20 websites visited.
- I was fairly certain this would yield nothing, and it didn’t. I only use Firefox in private mode, no cookies or history stored. Although Autopsy analyzed my `~/ .mozilla/firefox` directory, it was unable to find any websites visited, or settings stored. Again, I’ve never stored this information because I was already very aware people could analyze it if my computer is ever stolen.
- Types of external devices (eg. USB, printer, etc.) connected to the computer
- This was not possible to determine. I don’t know if Autopsy offers it as a feature, especially given that this wasn’t a live acquisition. There was nothing to report in the category of external devices.

2. Do a search to determine the number of times your name, and typical places your name appeared.
 - My name appears 485 places. Typically this is in files for school assignments (paper, source code etc). Most of them are from my NextCloud folder or directly from my documents.

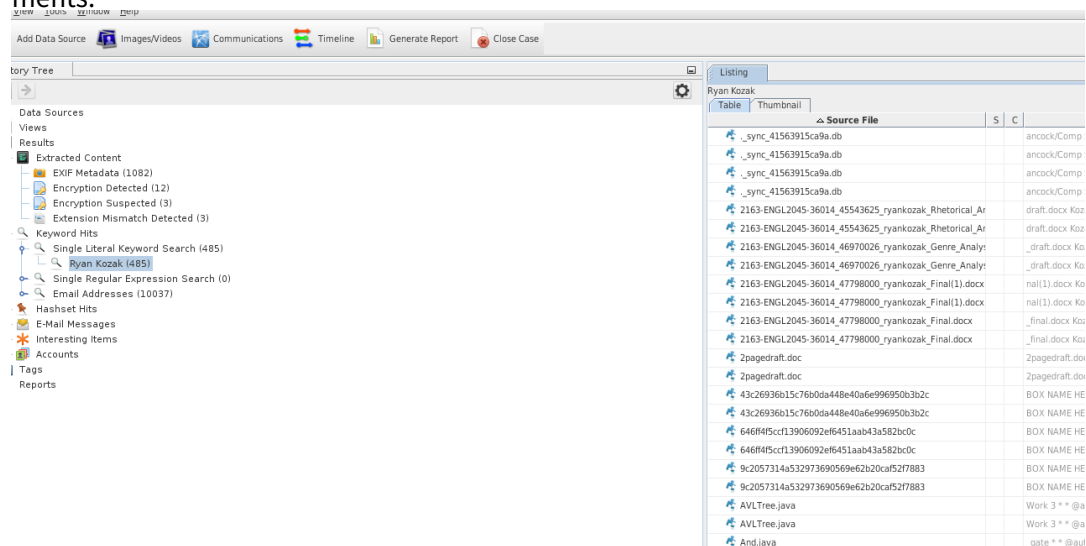


Figure 22: Number of times my name appears.

3. Do a search to determine the number of times CSUS or Sac State appeared and typical places where it appeared.
 - The phrase **CSUS** appeared 15 times. It appeared in my working assignment directory for CSC153 and CSC154, exactly where I'd have expected to find that keyword (see figure 23).

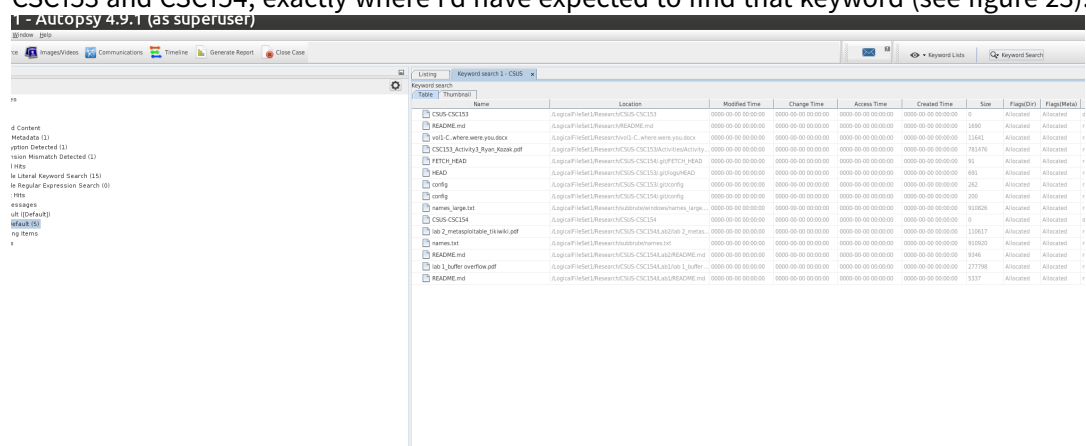


Figure 23: The phrase CSUS, appears 15 places.

- I'm a bit surprised Autopsy didn't detect my KeePassX database as an encrypted file, and

only considered my TrueCrypt container to be “Suspected” as an encrypted file. The number of images was higher than I anticipated too. What surprised me more was how much file system encryption actually protects you from this type of analysis. Even knowing my own encryption key didn’t make this process as easy as it would have been had the drive not been encrypted at all.

Conclusion

Encryption sure does keep you safe from prying eyes, but if you take a forensics class that tells you to analyze your own drive then your at war with yourself. I could have installed Windows and created dummy data for this Lab, and at a certain point I really considered it. However, I’d come so far in the process of actually doing this exercise on my real machine that I did not want to throw all that work away and turn in a boring lab report. I learned **a lot** doing this lab, and I think that was the whole point. d files were found?