
CSC153: Lab 2 - Data Acquisition and Basic Forensic Analysis

Ryan Kozak



2019-10-28

Objectives

- Perform Data Acquisition using FTK imager and Linux dd, dcfldd commands.
- Learn to use the Windows version Autopsy (version 4).
- Locate deleted/hidden files.
- Perform a dirty word search.
- Create a case report with any evidence you find.
- Understand the difference between disk formatting in Windows and zero-out.

Tasks

Task 1: Software Preparation.

The first task is to get our forensic tools downloaded and installed. FTK Imager and Autopsy 4 are to be installed on Windows and CAINE 9 is installed as a virtual machine. For this lab I'll be using an Ubuntu 18 host, running Virtual Box with virtual machines for Windows 10 and CAIN 9.

FTK Imager 4.2.1 was downloaded from [here](#) onto the Windows 10 virtual machine, and installed.

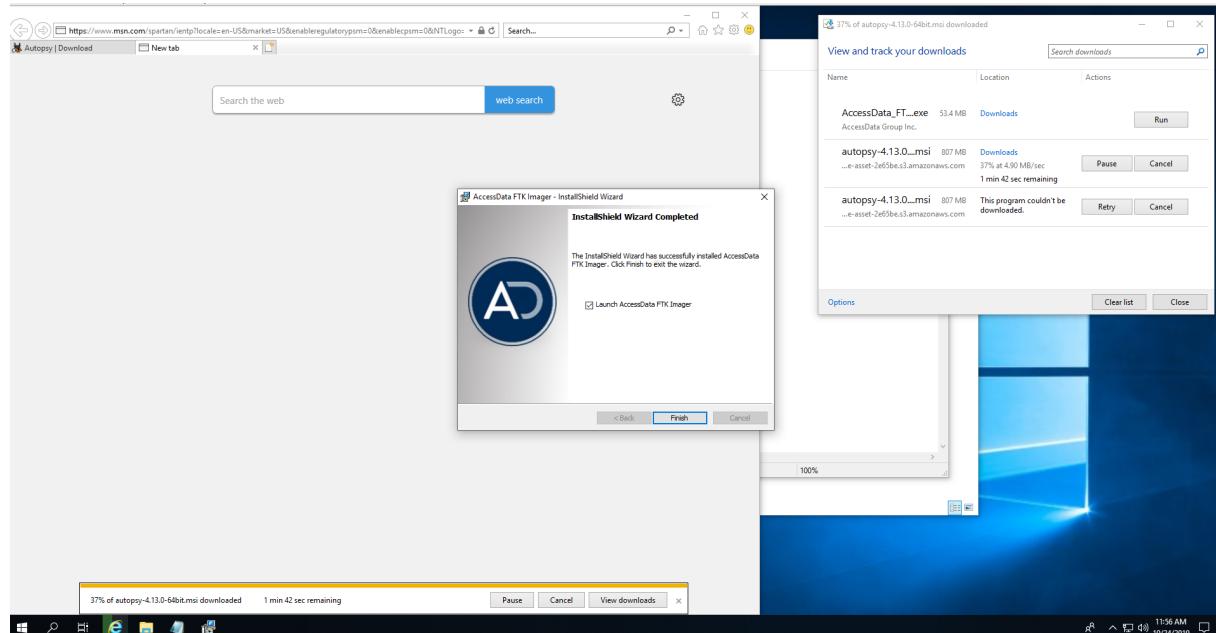


Figure 1: Installation of FTK Imager.

Autopsy 4.13 was downloaded from [here](#) onto the Windows 10 virtual machine, and installed.

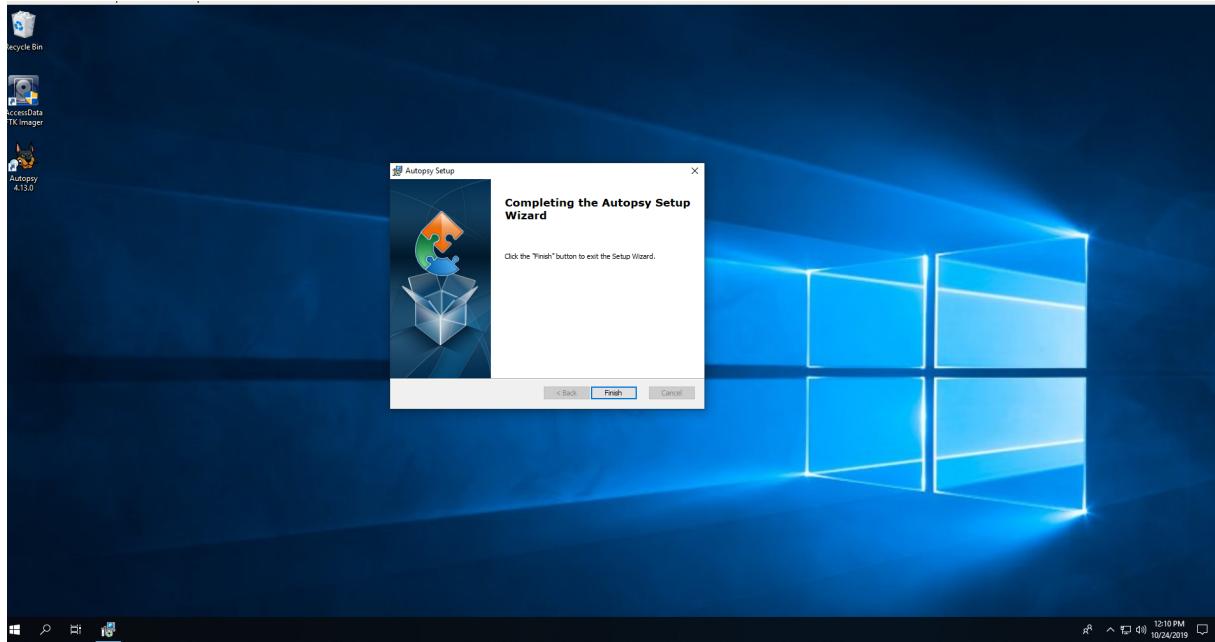


Figure 2: Installation of Autopsy.

The lab states “*Install a CAINE or Kali Linux virtual machine following the instructions of our in-class activity 1. You may skip this step if you decide to use the lab machines in RVR2009 to perform the steps involving Linux.*”. Because I completed activity 1 on my home machine, between the first and second class periods we were allowed to work on it, a CAINE 9 virtual machine was ready to go for me before beginning this lab.

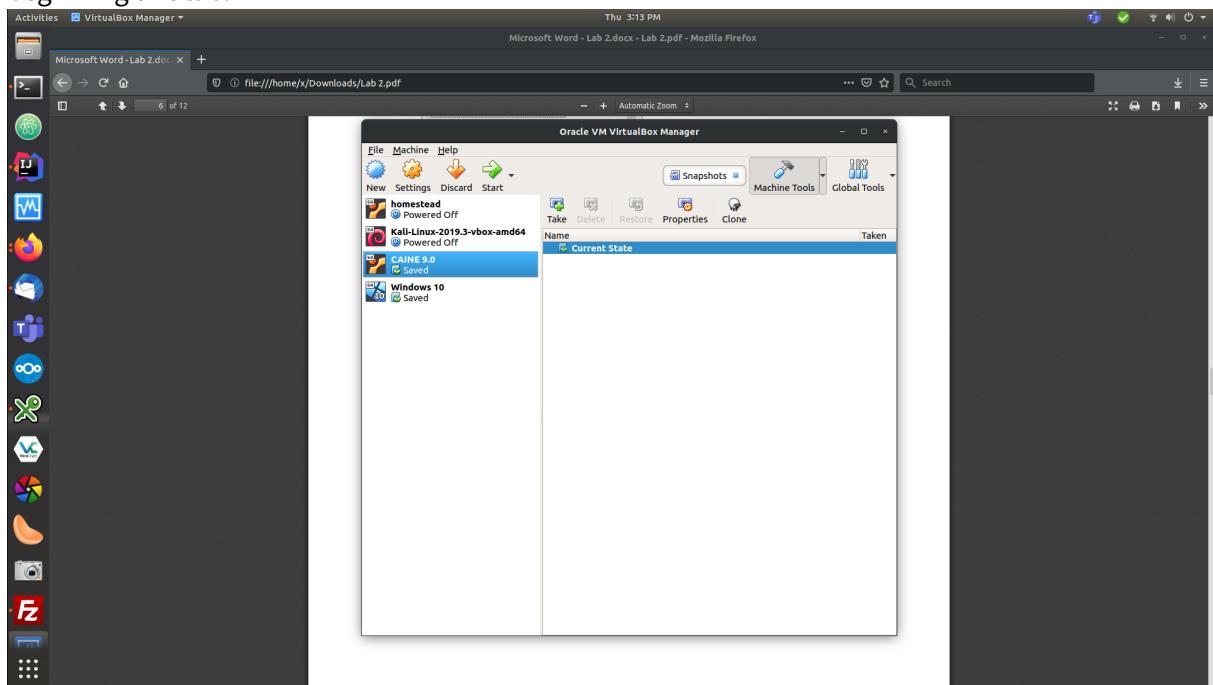


Figure 3: CAIN 9 ready to go on Virtual Box.

Task 2: Prepare a suspect drive.

The second task was to find a USB drive with the size of 500MB-2GB to work as the suspect drive. For this lab I'll be using a 1GB flash drive. **Note:** I have zeroed out this drive a few times already for this class, and just installed the file system prior to this lab.

We first connect the drive to the Windows virtual machine, and a file named `test.docx` is created via Microsoft Word. The contents of this file are the following.

> Life does not provide Warranties and Guarantees it only provides possibilities and opportunities for those who there to make best use of it!

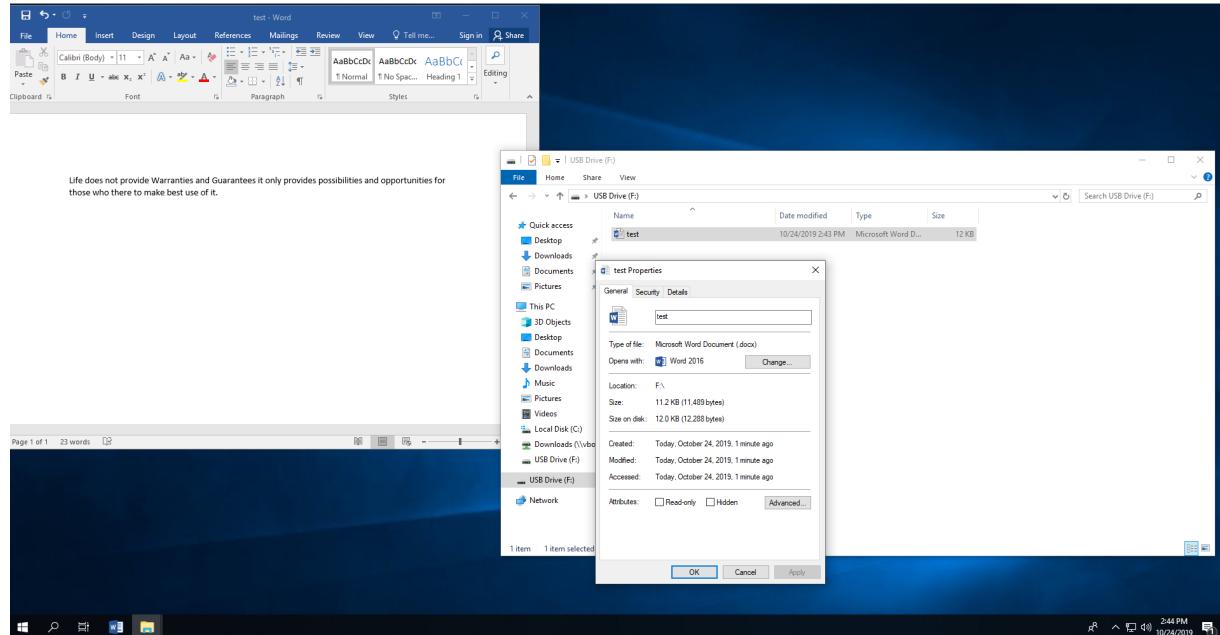


Figure 4: Creating `test.docx` on the Windows machine.

After this file was saved onto the USB drive, we right click on the drive and perform a disk format.

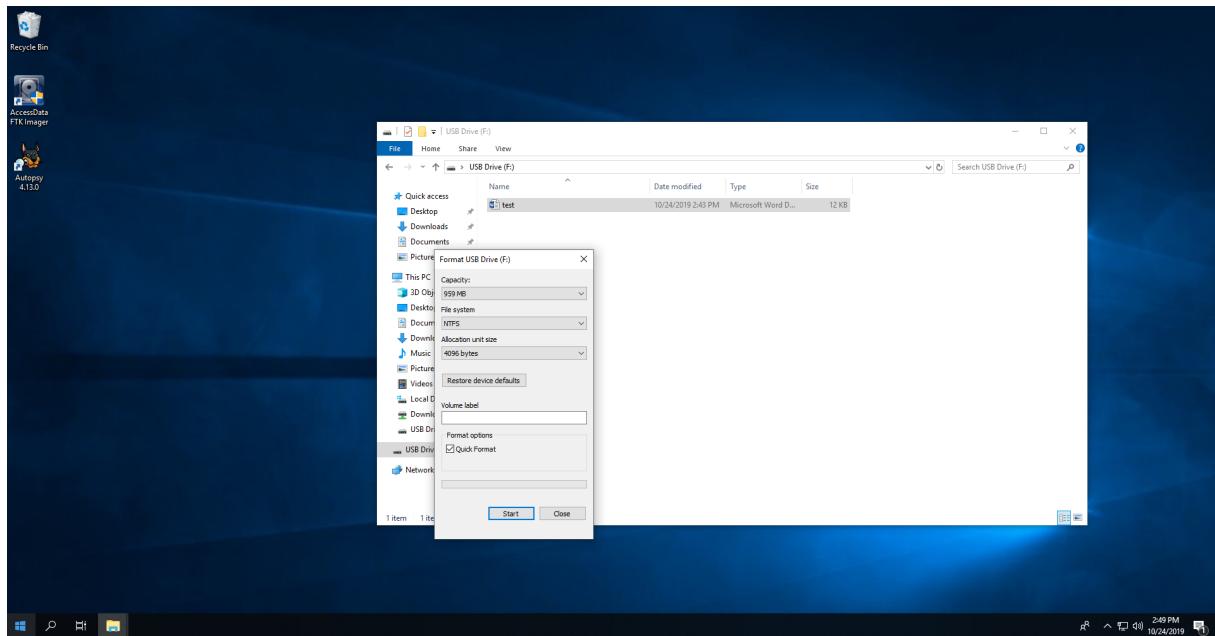


Figure 5: Formatting the USB drive on Windows, right after creating our `test.docx`.

Task 3: Perform a data acquisition with FTK Imager.

Next, we perform data acquisition with FTK Imager, which was installed in Step 1, and choose [File->Create Disk Image](#). In the Select Source dialog box, we click the Physical Drive option, and then click Next.

In the Select Drive dialog box, we click the Source Drive Selection list arrow, click on suspect drive, and then click Finish.

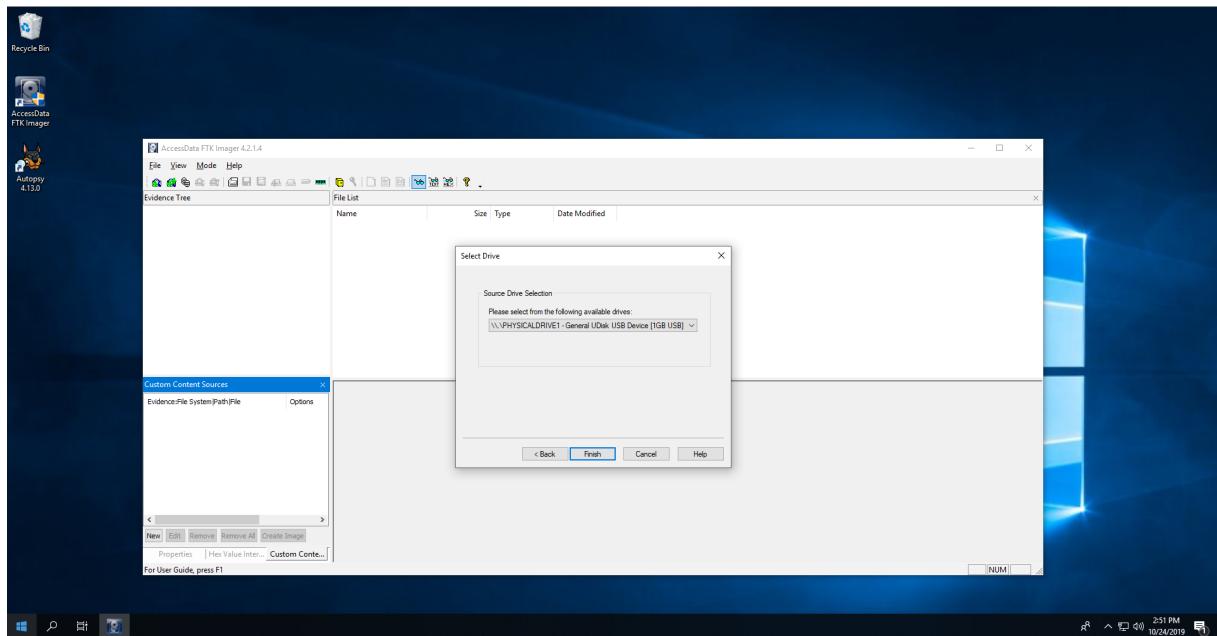


Figure 6: Selecting the suspect drive as physical source on FTK Imager.

In the Create Image dialog box, we select “*Verify images after they are created*”, and then click “Add”. Then, we select “*Raw*” as the Destination Image Type.

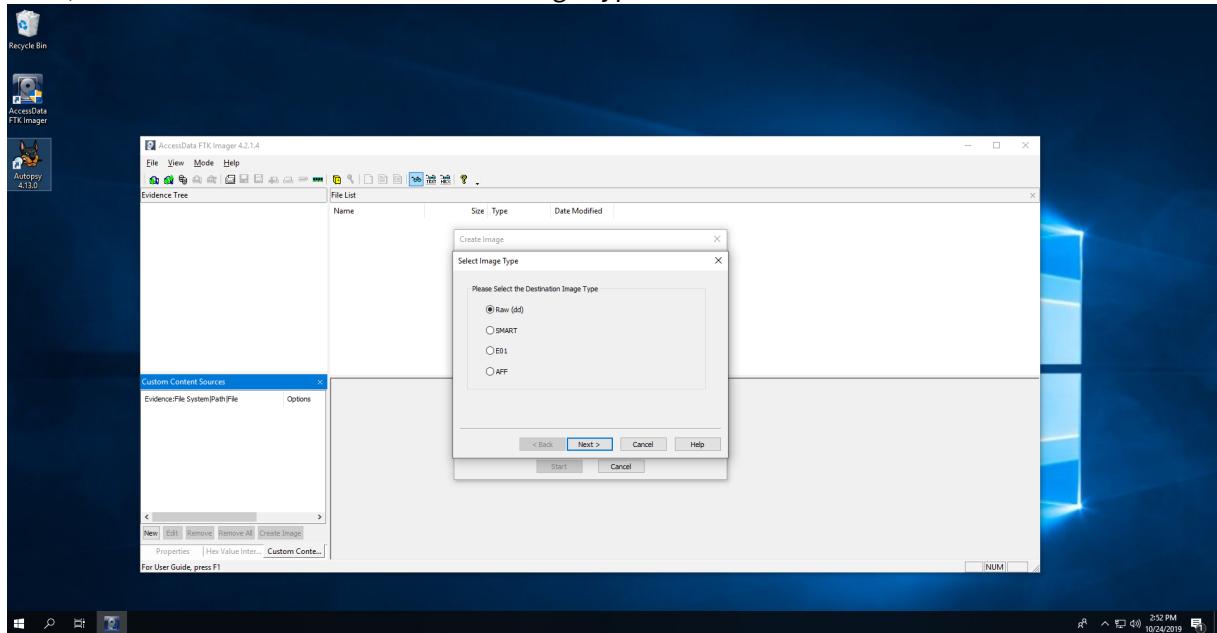


Figure 7: Raw data selected as destination image type on FTK Imager.

Complete the case information, and then click Next.

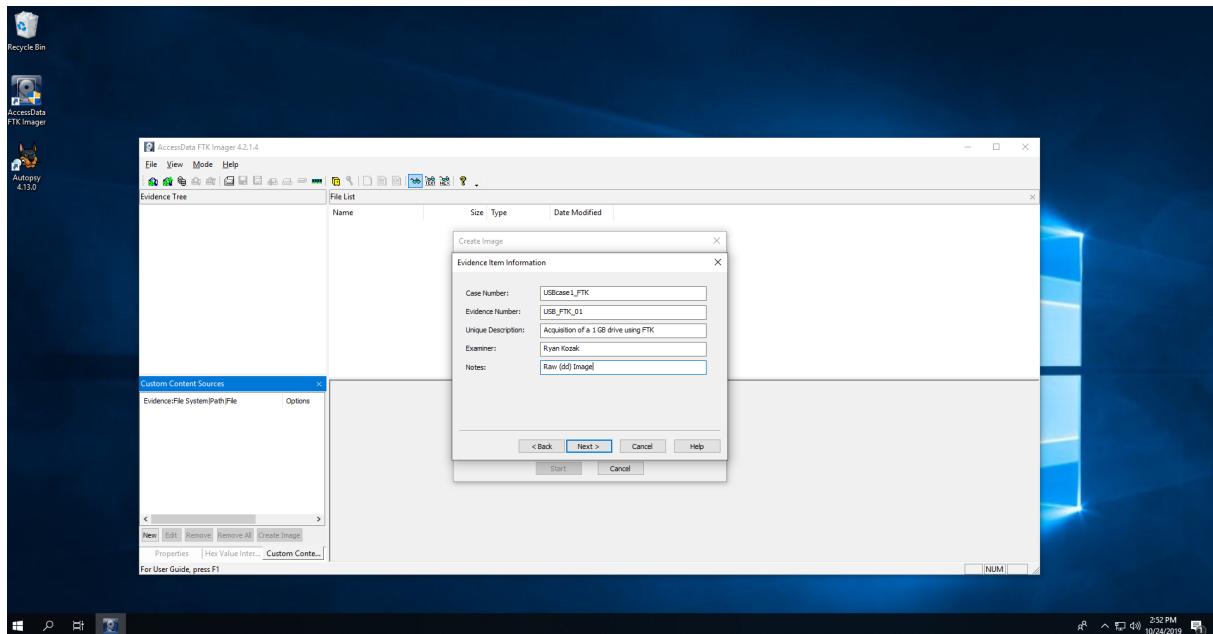


Figure 8: Completing evidence item information.

Next we click to the Select Image Destination dialog box, click Browse and specify the location we're storing the image. Also, we click to clear the Use AD Encryption check box.

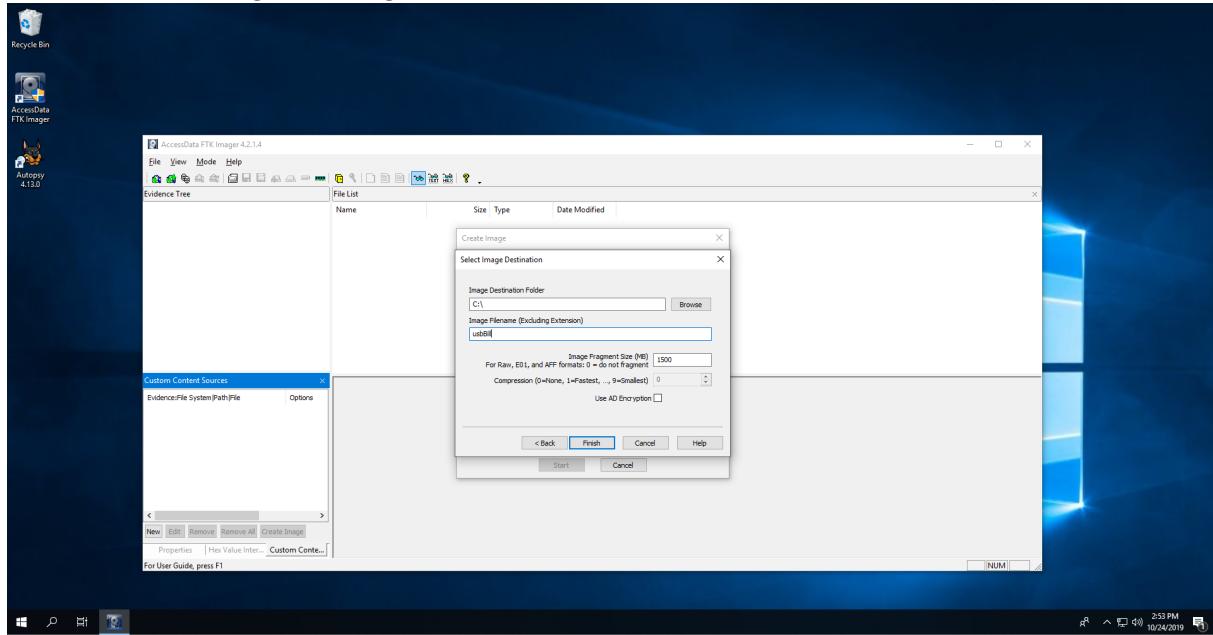


Figure 9: Selecting where to store the image file.

It's now time we start to initiate the acquisition.

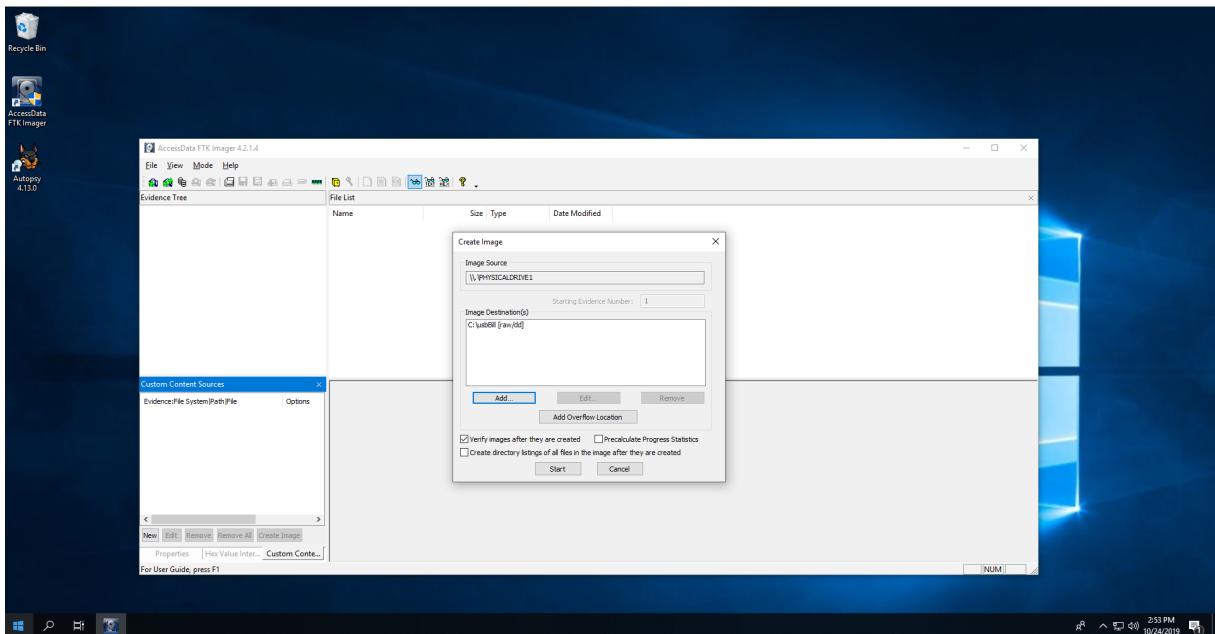


Figure 10: Ready to start the image creation.

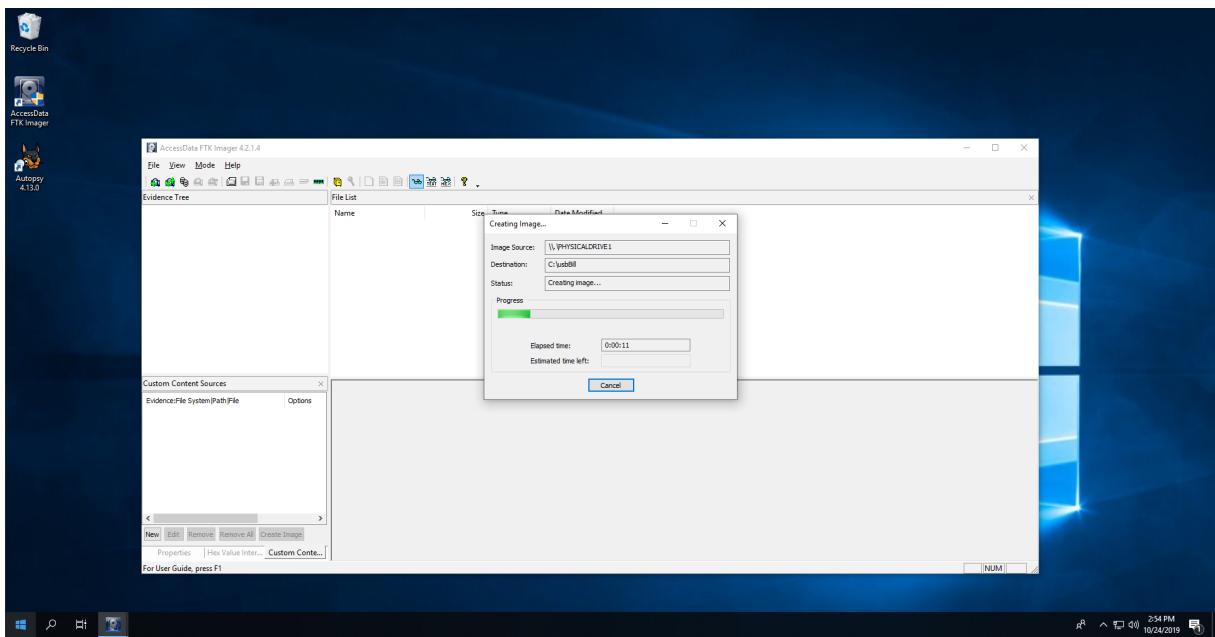


Figure 11: FTK creating the image.

Lastly, we review the information in the Drive/Image Verify Results dialog box.

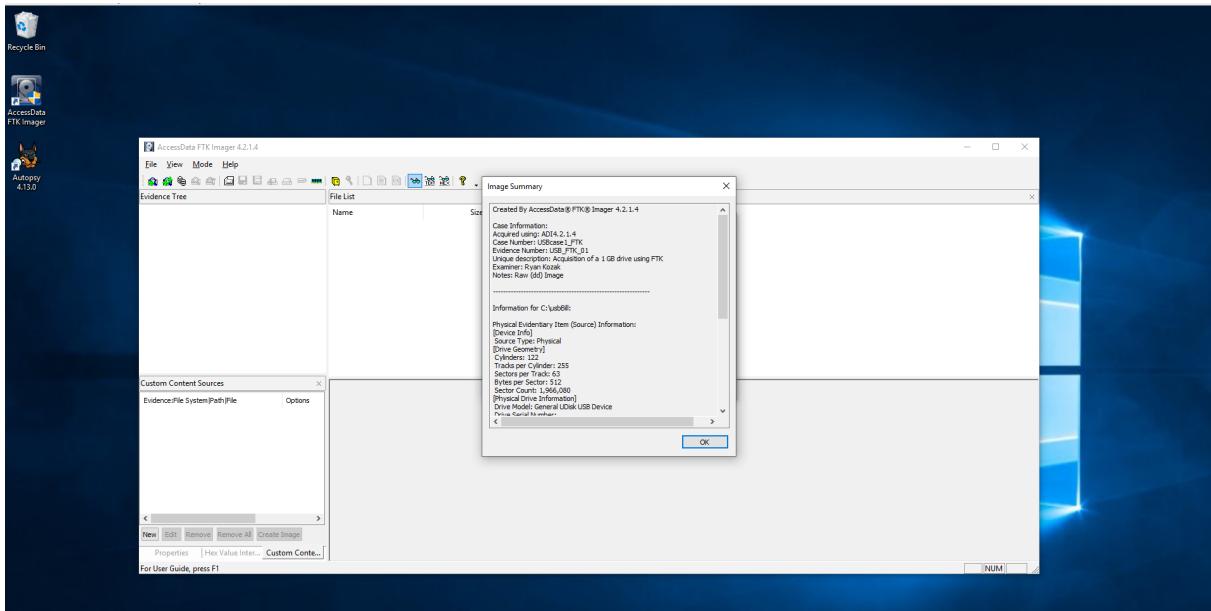


Figure 12: Image Summary.

Task 4: Perform a data acquisition with Linux dd/dcfldd command.

Next we open up the CAINE 9 virtual machine, and follow the instructions of in-class activity 3 to perform data acquisition of the USB drive using Linux dd/dcfldd commands.

The first step in activity 3 was to locate and then zero the drive on which we're to copy our evidence. I've included this part because it was part of activity 3.

```
[root@cainecf:~]# cat /etc/fstab
File Edit View Search Terminal Help
caine[fc1]# su root
Password:
[root@cainecf:~]# fdisk -l
Disk /dev/sda: 960 MiB, 1006632960 bytes, 62914560 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0eb2c8

Device Boot Start End Sectors Size Id Type
/dev/sda1 2848 62914559 62912152 30G 83 Linux

Disk /dev/sdb: 960 MiB, 1006632960 bytes, 19668800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc295ac

Device Boot Start End Sectors Size Id Type
/dev/sdb1 2848 62914559 62912152 30G 83 Linux

Disk /dev/sdc: 960 MiB, 1006632960 bytes, 62914560 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
Disk identifier: 0x0eb2c8

Device Boot Start End Sectors Size Id Type
/dev/sdc1 2848 62914559 62912152 30G 83 Linux

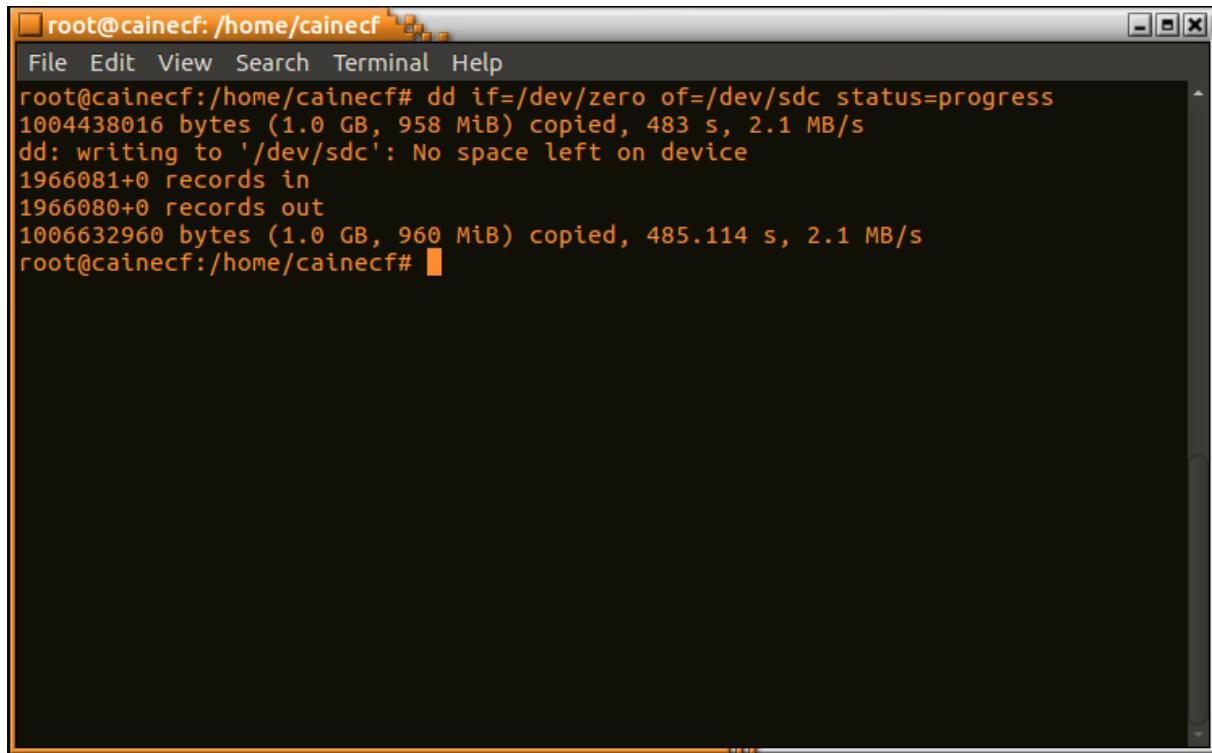
Disk /dev/sdb: 960 MiB, 1006632960 bytes, 19668800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0eb2c8

Device Boot Start End Sectors Size Id Type
/dev/sdh1 2848 1956679 1964032 559M c W55 FAT32 (LBA)

Disk /dev/sdc: 960 MiB, 1006632960 bytes, 62914560 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
Disk identifier: 0x0ddcc9

Device Boot Start End Sectors Size Id Type
/dev/sdc1 2848 1956679 1964032 559M 7 HPFS/NTFS/exFAT
[root@cainecf:~]#
```

Figure 13: /dev/sdb1 is our evidence drive, /dev/sdc1 is our target.



A terminal window titled "root@cainecf: /home/cainecf". The window contains the following text:

```
File Edit View Search Terminal Help
root@cainecf:/home/cainecf# dd if=/dev/zero of=/dev/sdc status=progress
1004438016 bytes (1.0 GB, 958 MiB) copied, 483 s, 2.1 MB/s
dd: writing to '/dev/sdc': No space left on device
1966081+0 records in
1966080+0 records out
1006632960 bytes (1.0 GB, 960 MiB) copied, 485.114 s, 2.1 MB/s
root@cainecf:/home/cainecf#
```

Figure 14: Zero out the target drive before acquisition.

Next we create a partition, and a file system on the target drive.

```

root@cainecf:/home/cainecf# fdisk /dev/sdc

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognised partition table.
Created a new DOS disklabel with disk identifier 0x2caa71a5.

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 (2048-1966079, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-1966079, default 1966079):

Created a new partition 1 of type 'Linux' and of size 959 MiB.

Command (m for help):

```

Figure 15: Creating a partition on `/dev/sdc`.

Code	Description
a OS/2 Boot Manag	OS/2 Boot Manager
b W95 FAT32	W95 FAT32
c W95 FAT32 (LBA)	W95 FAT32 (LBA)
e W95 FAT16 (LBA)	W95 FAT16 (LBA)
f W95 Ext'd (LBA)	W95 Extended (LBA)
10 OPUS	OPUS
11 Hidden FAT12	Hidden FAT12
12 Compaq diagnost	Compaq diagnostics
14 Hidden FAT16 <3	Hidden FAT16 <3
16 Hidden FAT16	Hidden FAT16
17 Hidden HPFS/NTF	Hidden HPFS/NTFS
18 AST SmartSleep	AST SmartSleep
1b Hidden W95 FAT3	Hidden W95 FAT3
1c Hidden W95 FAT3	Hidden W95 FAT3
1e Hidden W95 FAT1	Hidden W95 FAT1
50 OnTrack DM	OnTrack DM
51 OnTrack DM6 Aux	OnTrack DM6 Aux
52 CP/M	CP/M
53 OnTrack DM6 Aux	OnTrack DM6 Aux
54 OnTrackDM6	OnTrackDM6
55 EZ-Drive	EZ-Drive
56 Golden Bow	Golden Bow
5c Priam Edisk	Priam Edisk
61 SpeedStor	SpeedStor
63 GNU HURD or Sys	GNU HURD or Sys
64 Novell Netware	Novell Netware
65 Novell Netware	Novell Netware
70 DiskSecure Mult	DiskSecure Mult
75 PC/IX	PC/IX
80 Old Minix	Old Minix
94 Amoeba BBT	Amoeba BBT
9f BSD/OS	BSD/OS
a0 IBM Thinkpad hi	IBM Thinkpad hi
a5 FreeBSD	FreeBSD
a6 OpenBSD	OpenBSD
a7 NeXTSTEP	NeXTSTEP
a8 Darwin UFS	Darwin UFS
a9 NetBSD	NetBSD
ab Darwin boot	Darwin boot
af HFS / HFS+	HFS / HFS+
b7 BSDI fs	BSDI fs
b8 BSDI swap	BSDI swap
bb Boot Wizard hid	Boot Wizard hid
bc Acronis FAT32 L	Acronis FAT32 L
be Solaris boot	Solaris boot
ff BBT	BBT

```

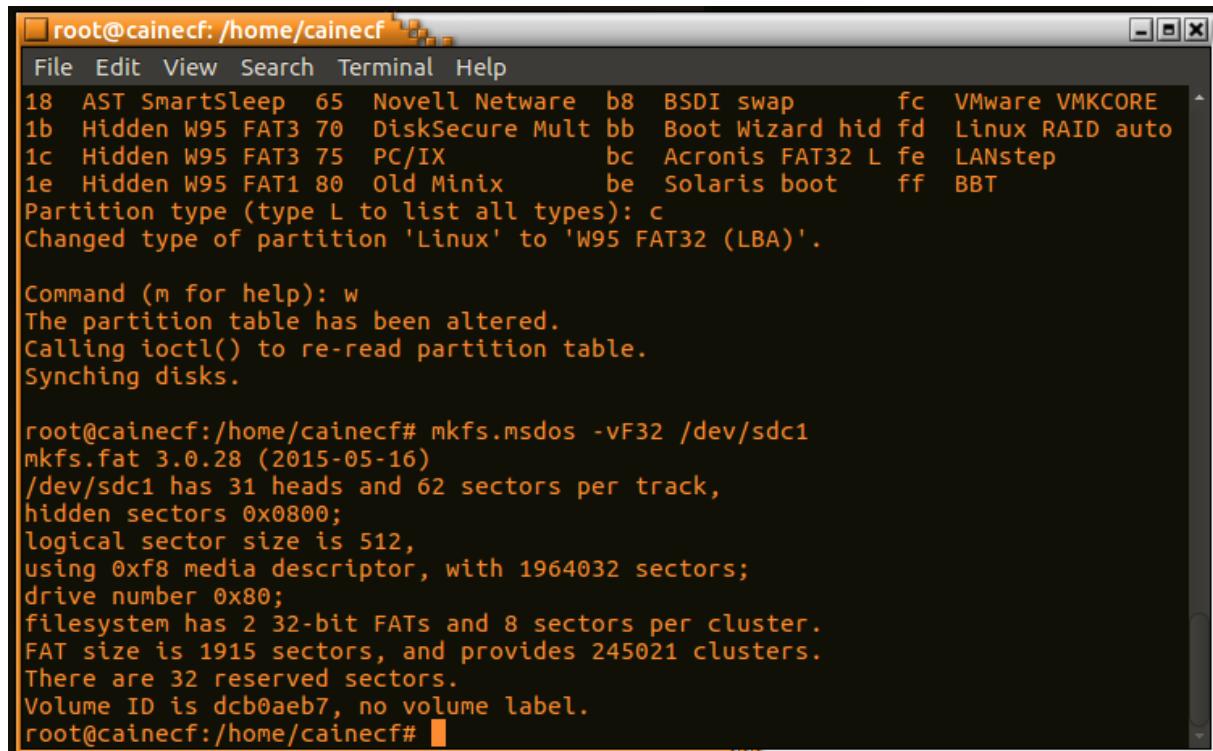
File Edit View Search Terminal Help
a OS/2 Boot Manag 50 OnTrack DM      94 Amoeba BBT      e3 DOS R/O
b W95 FAT32       51 OnTrack DM6 Aux 9f BSD/OS        e4 SpeedStor
c W95 FAT32 (LBA) 52 CP/M          a0 IBM Thinkpad hi ea Rufus alignment
e W95 FAT16 (LBA) 53 OnTrack DM6 Aux a5 FreeBSD       eb BeOS fs
f W95 Ext'd (LBA) 54 OnTrackDM6    a6 OpenBSD       ee GPT
10 OPUS           55 EZ-Drive       a7 NeXTSTEP     ef EFI (FAT-12/16/
11 Hidden FAT12    56 Golden Bow    a8 Darwin UFS   f0 Linux/PA-RISC b
12 Compaq diagnost 5c Priam Edisk  a9 NetBSD       f1 SpeedStor
14 Hidden FAT16 <3 61 SpeedStor    ab Darwin boot  f4 SpeedStor
16 Hidden FAT16    63 GNU HURD or Sys af HFS / HFS+  f2 DOS secondary
17 Hidden HPFS/NTF 64 Novell Netware b7 BSDI fs      fb VMware VMFS
18 AST SmartSleep 65 Novell Netware b8 BSDI swap    fc VMware VMKCORE
1b Hidden W95 FAT3 70 DiskSecure Mult bb Boot Wizard hid fd Linux RAID auto
1c Hidden W95 FAT3 75 PC/IX         bc Acronis FAT32 L fe LANstep
1e Hidden W95 FAT1 80 Old Minix    be Solaris boot  ff BBT
Partition type (type L to list all types): c
Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

root@cainecf:/home/cainecf#

```

Figure 16: Write W95 FAT 32 (LBA).



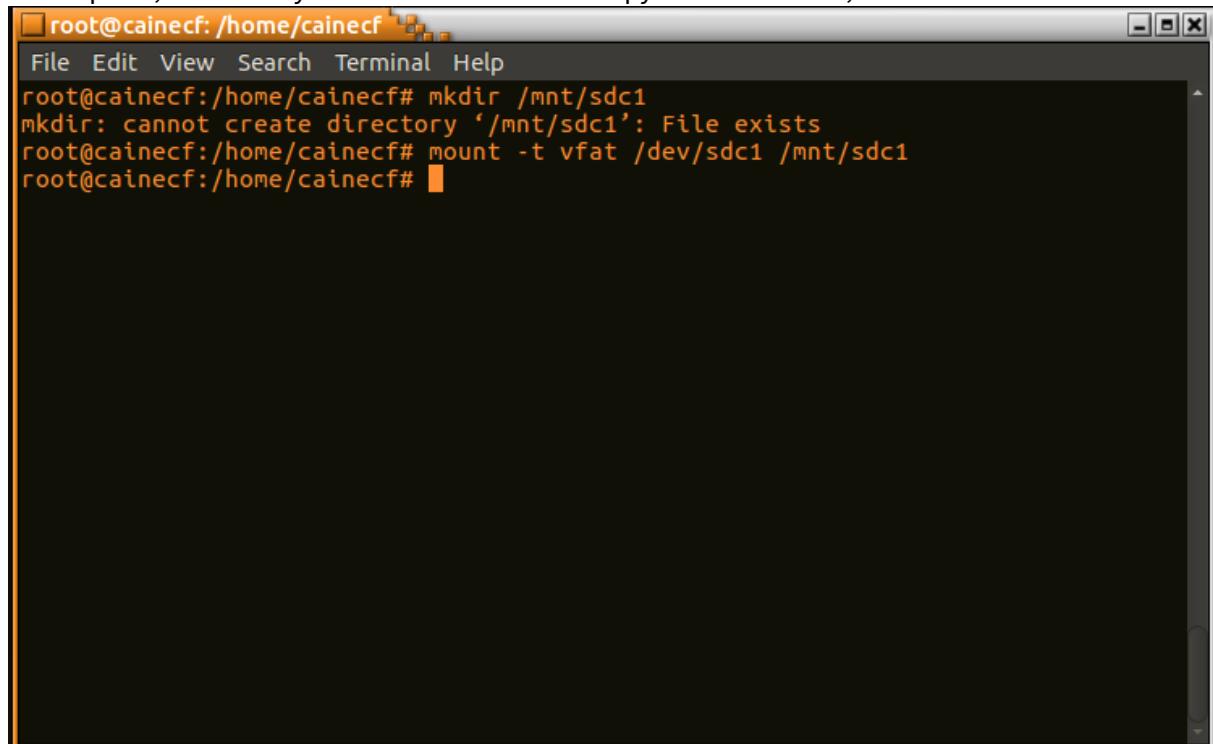
```
root@cainecf:/home/cainecf
File Edit View Search Terminal Help
18 AST SmartSleep 65 Novell Netware b8 BSDI swap      fc VMware VMKCORE
1b Hidden W95 FAT3 70 DiskSecure Mult bb Boot Wizard hid fd Linux RAID auto
1c Hidden W95 FAT3 75 PC/IX          bc Acronis FAT32 L fe LANstep
1e Hidden W95 FAT1 80 Old Minix     be Solaris boot    ff BBT
Partition type (type L to list all types): c
Changed type of partition 'Linux' to 'W95 FAT32 (LBA)'.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

root@cainecf:/home/cainecf# mkfs.msdos -vF32 /dev/sdc1
mkfs.fat 3.0.28 (2015-05-16)
/dev/sdc1 has 31 heads and 62 sectors per track,
hidden sectors 0x0800;
logical sector size is 512,
using 0xf8 media descriptor, with 1964032 sectors;
drive number 0x80;
filesystem has 2 32-bit FATs and 8 sectors per cluster.
FAT size is 1915 sectors, and provides 245021 clusters.
There are 32 reserved sectors.
Volume ID is dcbaeb7, no volume label.
root@cainecf:/home/cainecf#
```

Figure 17: Make FAT 32 file system on the new partition.

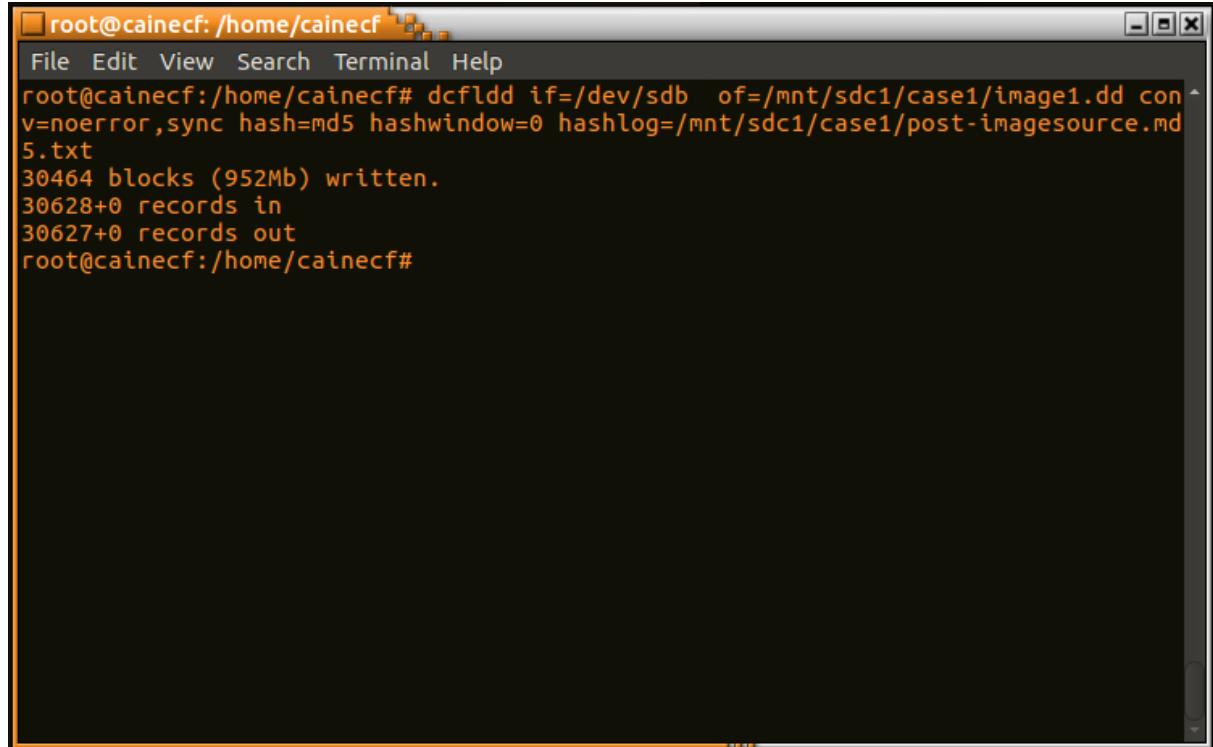
At this point, we're ready to mount the to store a copy of our evidence, and evidence hashes.



```
root@cainecf:/home/cainecf
File Edit View Search Terminal Help
root@cainecf:/home/cainecf# mkdir /mnt/sdc1
mkdir: cannot create directory '/mnt/sdc1': File exists
root@cainecf:/home/cainecf# mount -t vfat /dev/sdc1 /mnt/sdc1
root@cainecf:/home/cainecf#
```

Figure 18: Mounting drive to copy our evidence to.

Once our target drive is mounted, we create a `case1` directory, and acquire the data.

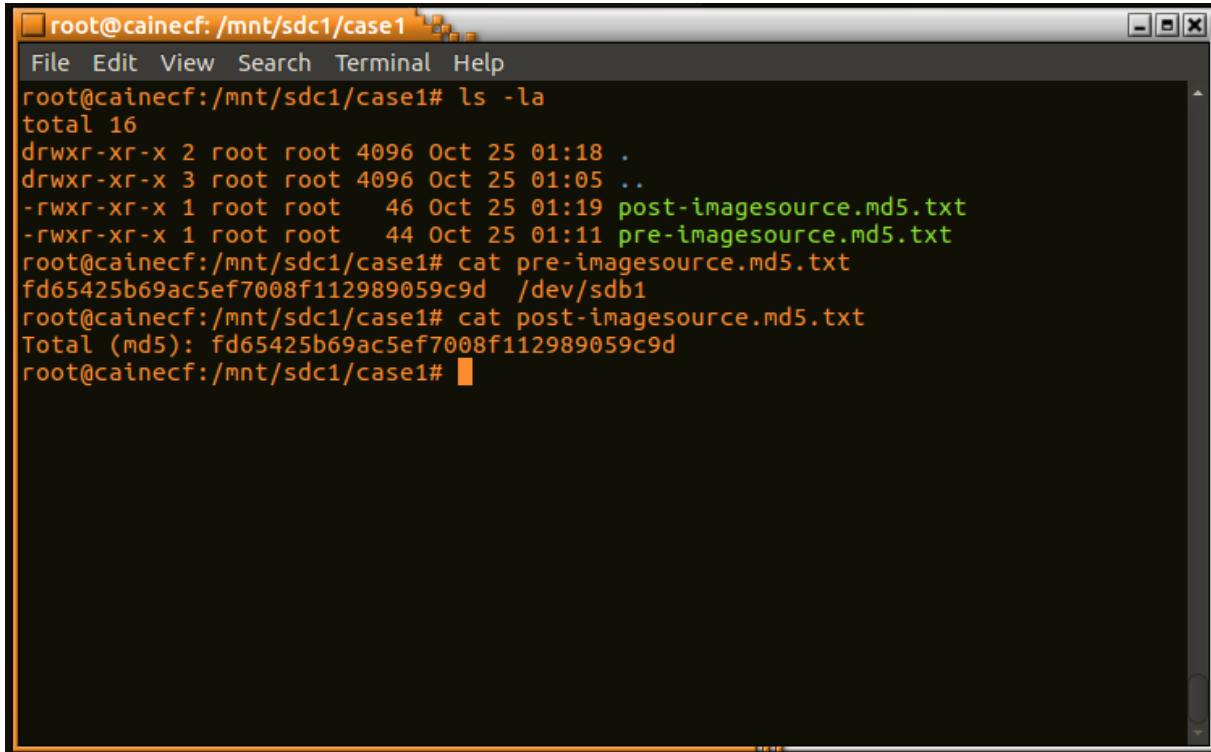


A screenshot of a terminal window titled "root@cainecf: /home/cainecf". The window contains the following text:

```
root@cainecf:/home/cainecf# dcfldd if=/dev/sdb of=/mnt/sdc1/case1/image1.dd con-
v=noerror,sync hash=md5 hashwindow=0 hashlog=/mnt/sdc1/case1/post-imagesource.md
5.txt
30464 blocks (952Mb) written.
30628+0 records in
30627+0 records out
root@cainecf:/home/cainecf#
```

Figure 19: Acquisition of data to `case1` directory with `dcfldd`.

After acquisition is complete we verify the image by checking the the values of the `pre-imagesource.md5.txt` and `post-imagesource.md5.txt` contain the same hash value.

A screenshot of a terminal window titled "root@cainecf: /mnt/sdc1/case1". The window shows a command-line session where the user runs "ls -la" to list files, then "cat pre-imagesource.md5.txt" and "cat post-imagesource.md5.txt" to compare their contents. Both files show the same MD5 hash: "fd65425b69ac5ef7008f112989059c9d".

```
root@cainecf: /mnt/sdc1/case1# ls -la
total 16
drwxr-xr-x 2 root root 4096 Oct 25 01:18 .
drwxr-xr-x 3 root root 4096 Oct 25 01:05 ..
-rw-r--r-- 1 root root 46 Oct 25 01:19 post-imagesource.md5.txt
-rw-r--r-- 1 root root 44 Oct 25 01:11 pre-imagesource.md5.txt
root@cainecf: /mnt/sdc1/case1# cat pre-imagesource.md5.txt
fd65425b69ac5ef7008f112989059c9d
root@cainecf: /mnt/sdc1/case1# cat post-imagesource.md5.txt
Total (md5): fd65425b69ac5ef7008f112989059c9d
root@cainecf: /mnt/sdc1/case1#
```

Figure 20: Verified acquisition by checking `md5sum` results.

Now there is an `image1.dd` file created from our evidence drive. It's time to analyze this in Autopsy, so we transfer the evidence drive to our Windows virtual machine.

Task 5: Analyze the acquired data.

We open Autopsy in our Windows VM that we installed in Task 1, and create a new case.

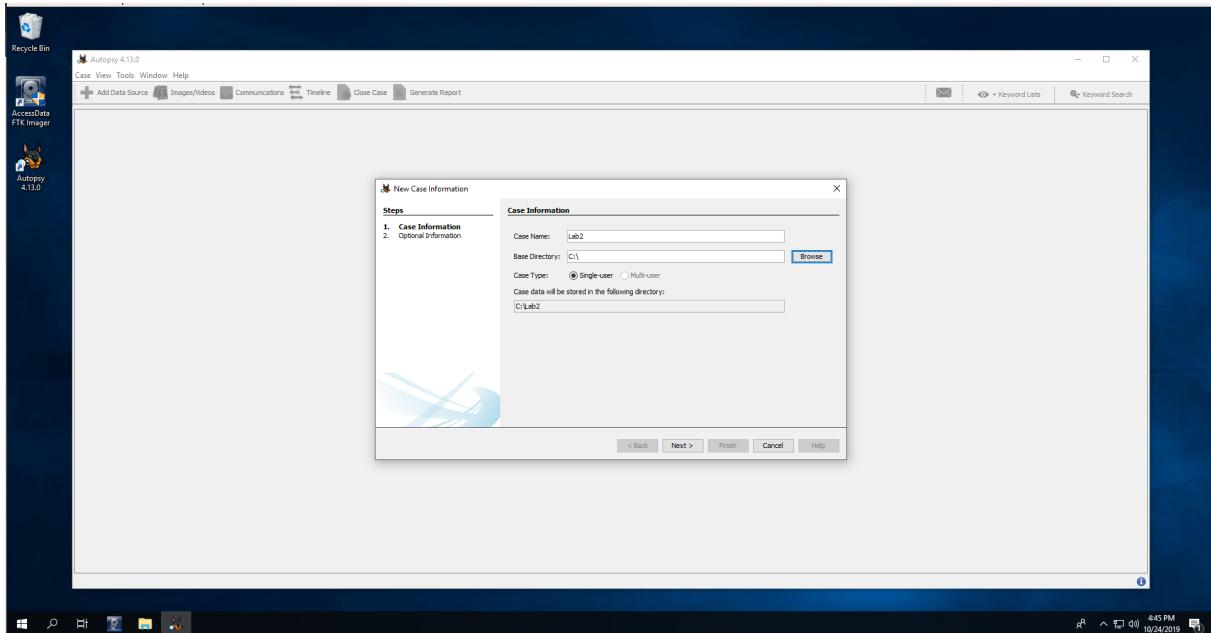


Figure 21: Creating Lab2 case in Autopsy.

Next, we add the image by choosing Disk Image or VM File.

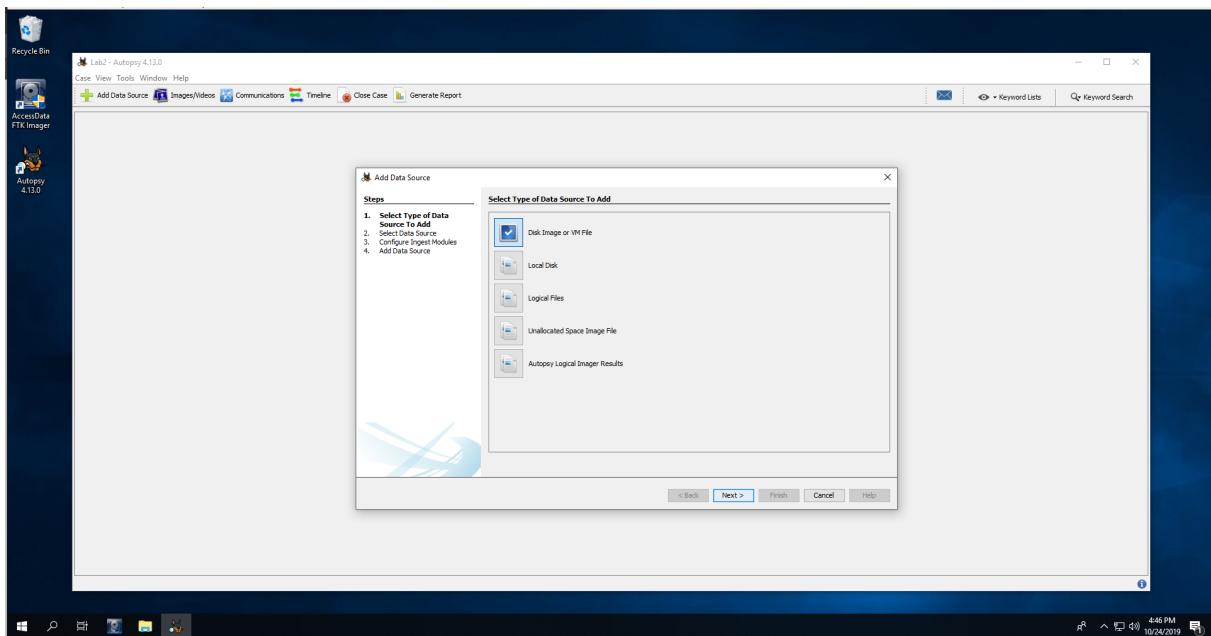


Figure 22: Select to add a disk image file.

For our data source, we select the raw image acquired via `dcfldd`.

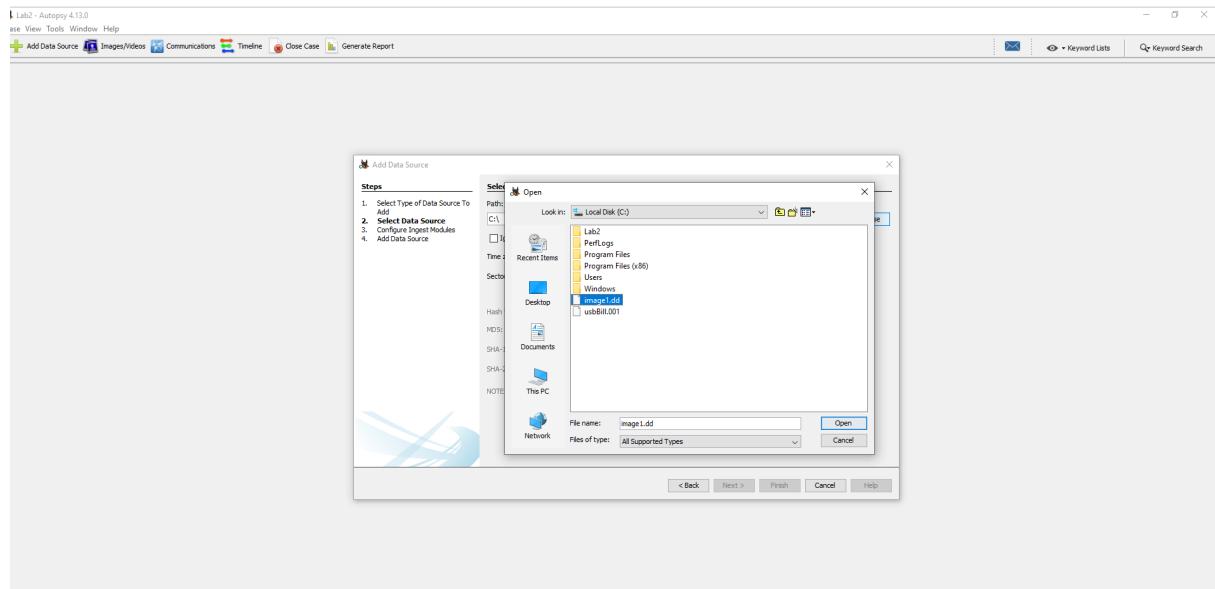


Figure 23: Data source is image `image1.dd`.

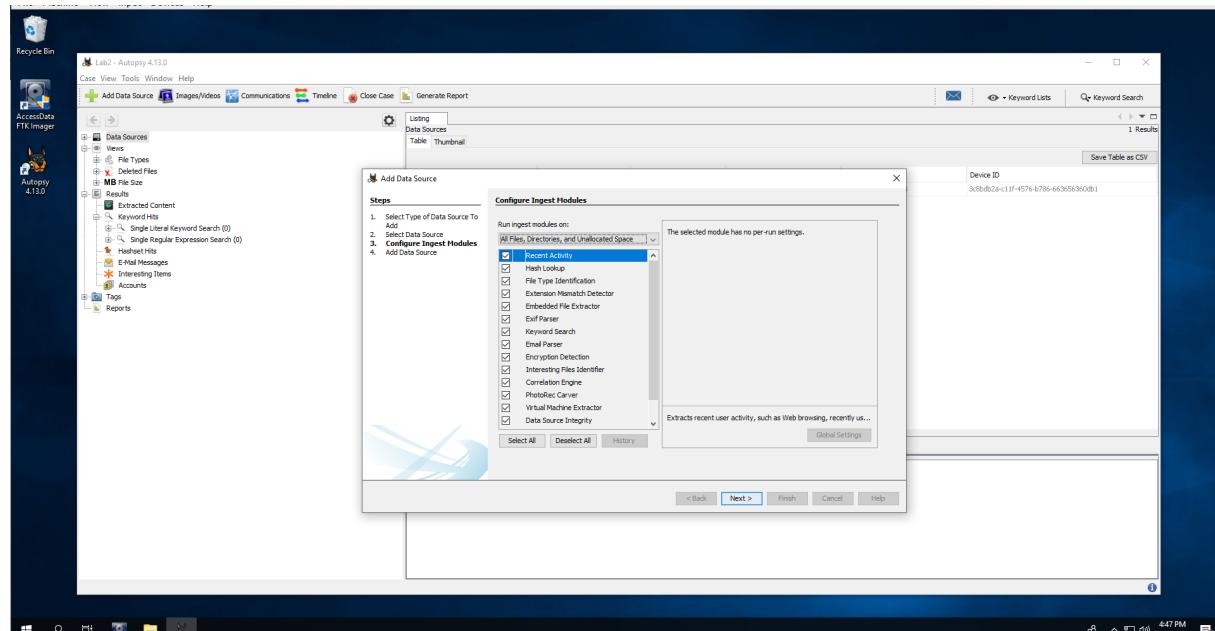


Figure 24: All modules configured.

We click on *Data Sources* to view the image. Next we navigate to the tabs under *Views* to check the files on this USB drive. Because I've been using this drive for CSC 153 already, it's been zeroed a couple of times previously. The only file on this drive is the file we've placed on here in Task 2, `test.docx`. However, that appears as a deleted file, due to our formatting of the drive in Task 2 directly after creating the file. So, we click on deleted files to see the deleted files.

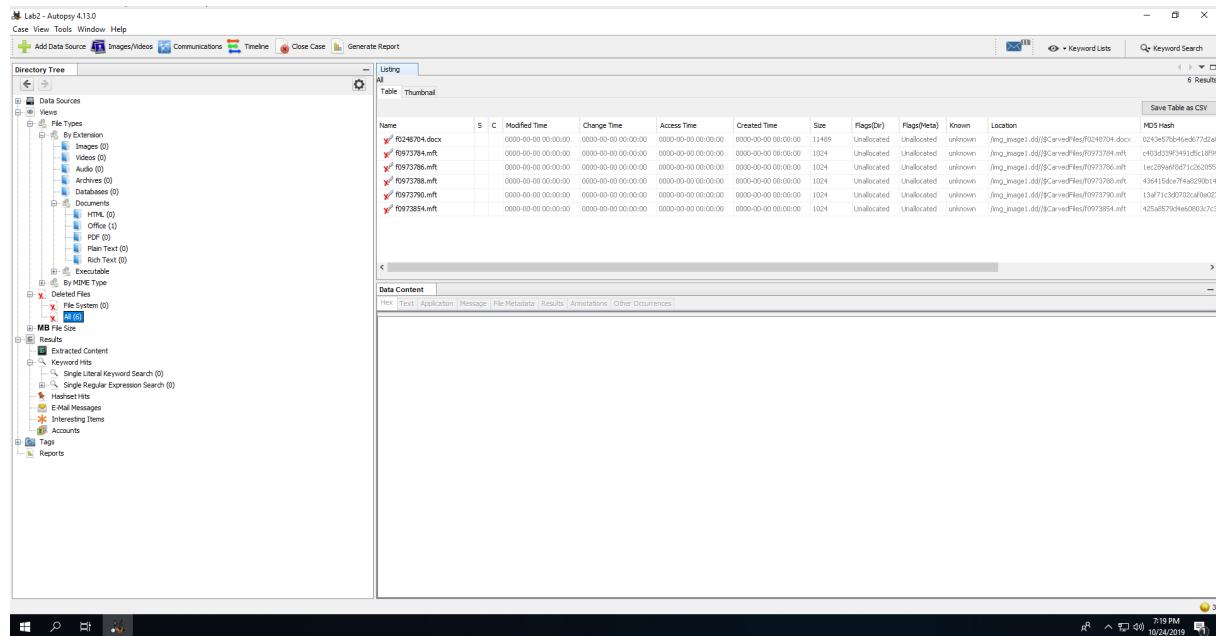


Figure 25: There is one deleted .docx file, and mft records, nothing more.

We can examine this deleted file `f0248704.docx` by right-clicking on the file, choosing `Extract file`.

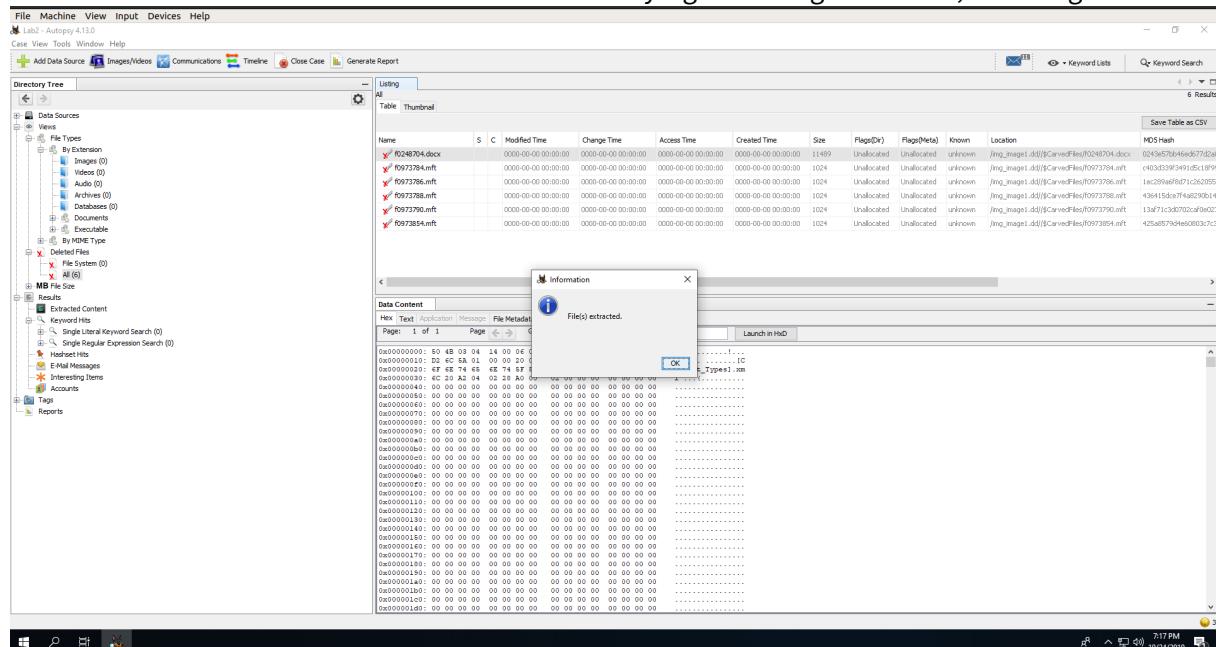


Figure 26: Extracted deleted .docx file.

When we open `f0248704.docx` in Microsoft Word, we see that the contents are that of our `test.docx` file. The file was deleted by the disk formatting performed in Task 2, but we've recovered it.

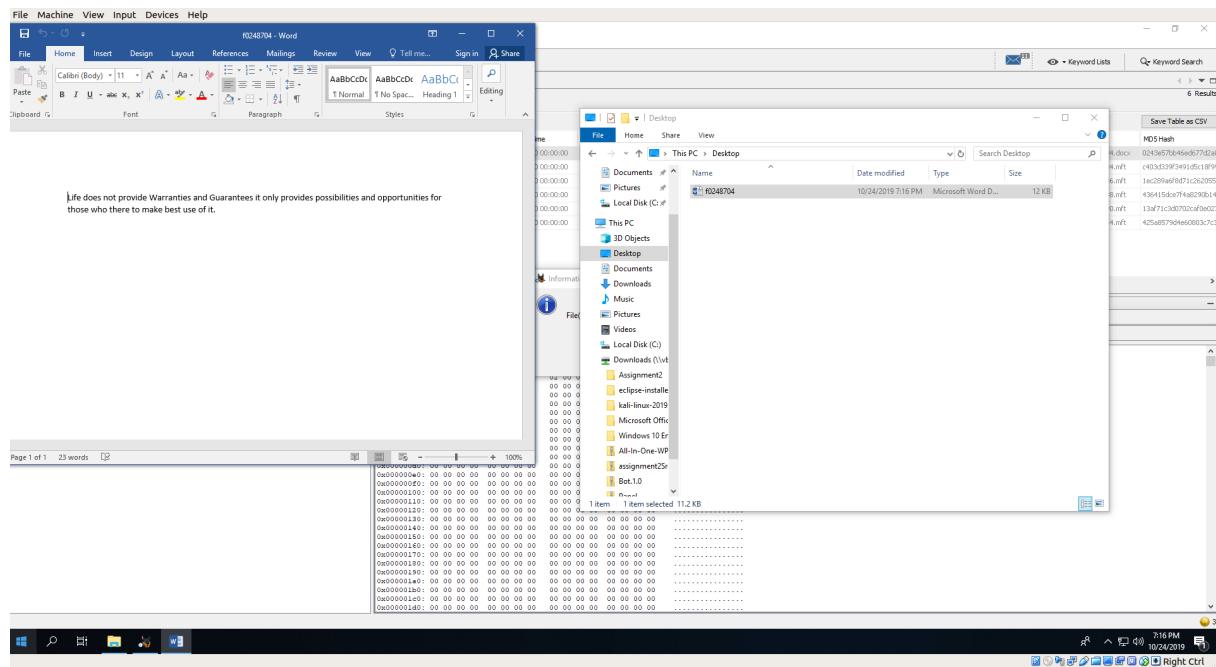


Figure 27: Contents of the deleted file match `test.docx`.

Task 6: Perform a dirty word search.

For the purposes of this lab we will just use *Warranties* as the key word for our dirty word search, as we're aware that this word is in our deleted file.

We click on the “Keyword Search” button, ensure ASCII, and Case Insensitive are selected, and type in *Warranties*.

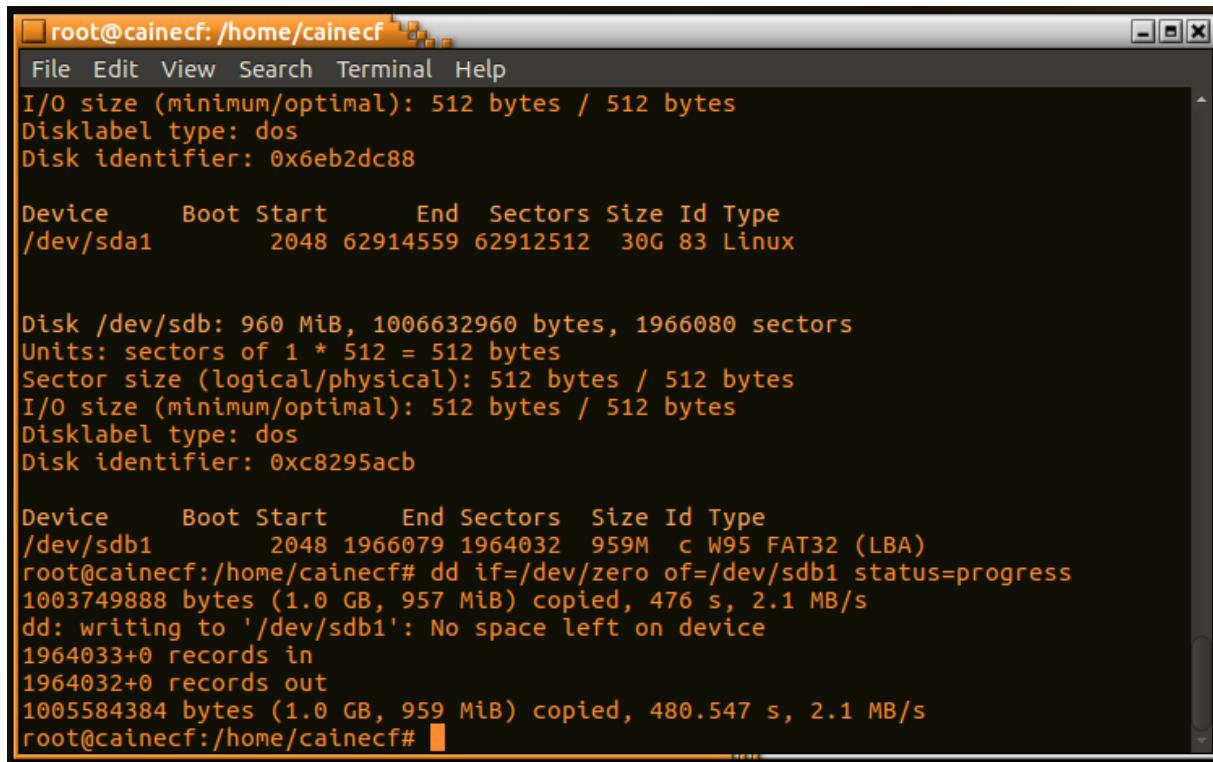
Name	Location	Modified Time	Change Time	Access Time	Created Time	Size	Flag(DF)	Flag(Md5)	Known	MD5 Hash
R0248704.docx	Img_Img1.dfru:\$CarvedFiles\$R0248704.docx	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	11469	Unallocated	Unallocated	Unknown	0248704e0ed77d2d03c

Figure 28: Keyword (dirty word) search results for *Warranties*.

Figure 29: Hit and file information from our dirty word search.

Task 7: Zero-out the suspect USB drive.

Now we are going to zero-out the suspect USB drive. First we connect the same suspect USB drive back up to our CAIN virtual machine. In our case it's at `/dev/sdb`. **Then, we zero-out the drive.**



The screenshot shows a terminal window titled "root@cainecf: /home/cainecf". The terminal displays the following information:

```
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x6eb2dc88

Device      Boot Start      End Sectors Size Id Type
/dev/sda1          2048 62914559 62912512 30G 83 Linux

Disk /dev/sdb: 960 MiB, 1006632960 bytes, 1966080 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: 0xc8295acb

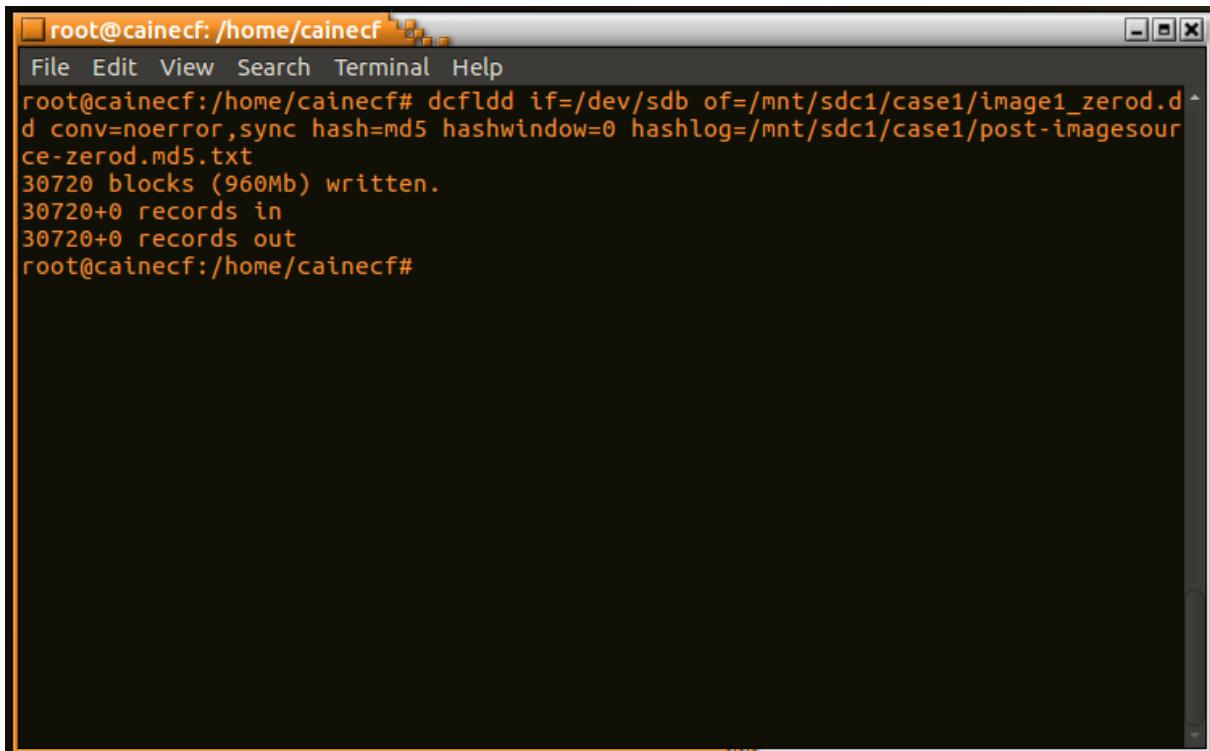
Device      Boot Start      End Sectors Size Id Type
/dev/sdb1          2048 1966079 1964032 959M c W95 FAT32 (LBA)
root@cainecf:/home/cainecf# dd if=/dev/zero of=/dev/sdb1 status=progress
1003749888 bytes (1.0 GB, 957 MiB) copied, 476 s, 2.1 MB/s
dd: writing to '/dev/sdb1': No space left on device
1964033+0 records in
1964032+0 records out
1005584384 bytes (1.0 GB, 959 MiB) copied, 480.547 s, 2.1 MB/s
root@cainecf:/home/cainecf#
```

Figure 30: Zero-out the suspect drive.

Task 8: Repeat Tasks 4-6.

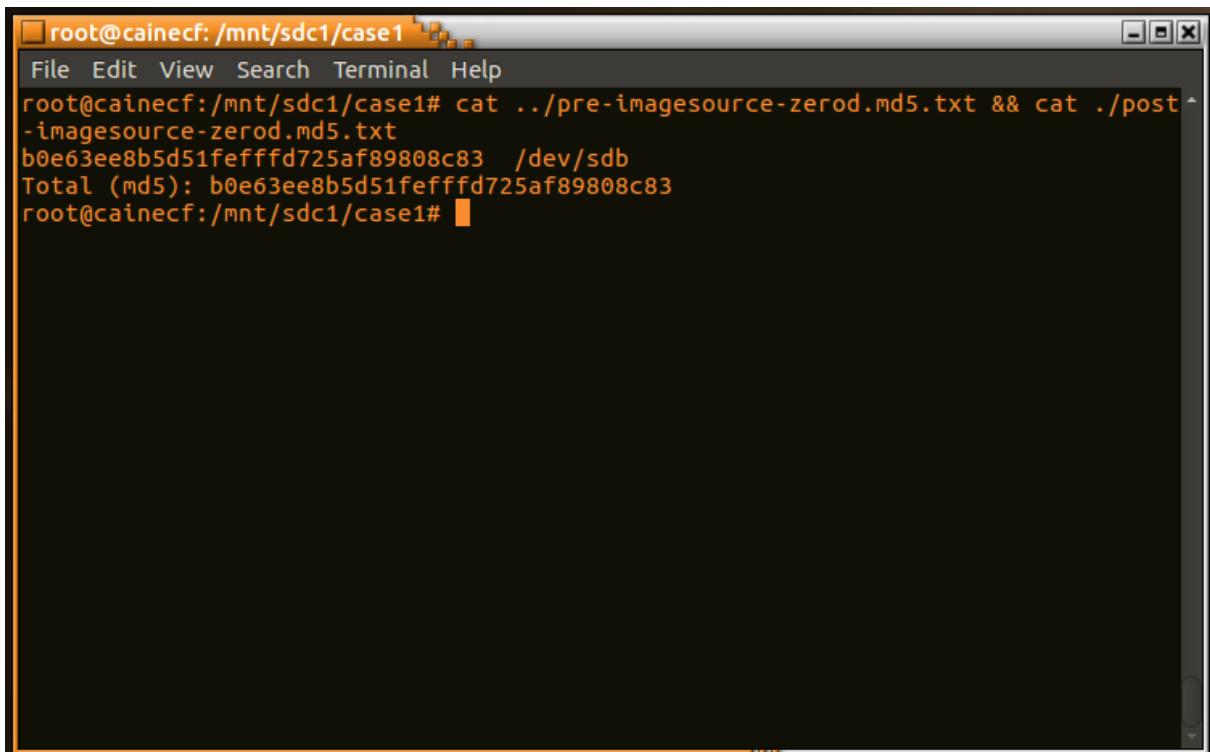
We now begin to repeat the activities 4-6 towards the zeroed-out USB drive. For activity 4, we do not need to zero out the drive on which we store evidence, or create a case directory. That part was to ensure a safe place to store evidence, and it still is.

So, at this point we acquire an image of the drive again using `dcfldd` and compare the hashes once more to confirm integrity.



```
root@cainecf: /home/cainecf# dd if=/dev/sdb of=/mnt/sdc1/case1/image1_zerod.dd conv=noerror,sync hash=md5 hashwindow=0 hashlog=/mnt/sdc1/case1/post-imagesource-zerod.md5.txt
30720 blocks (960Mb) written.
30720+0 records in
30720+0 records out
root@cainecf: /home/cainecf#
```

Figure 31: Acquiring `image1_zerod.dd` of suspect drive after zero-out.



```
root@cainecf: /mnt/sdc1/case1# cat ../pre-imagesource-zerod.md5.txt && cat ./post-imagesource-zerod.md5.txt
b0e63ee8b5d51feffffd725af89808c83  /dev/sdb
Total (md5): b0e63ee8b5d51feffffd725af89808c83
root@cainecf: /mnt/sdc1/case1#
```

Figure 32: Verify `image1_zerod.dd` with previously calculated hash of the zeroed drive.

Now `image1_zerod.dd` contains an image of the drive after we've run our zero-out. We begin to repeat Task 5 by opening up the Windows virtual machine once more, and then opening Autopsy. We then create a case and add `image1_zerod.dd` as a raw image.

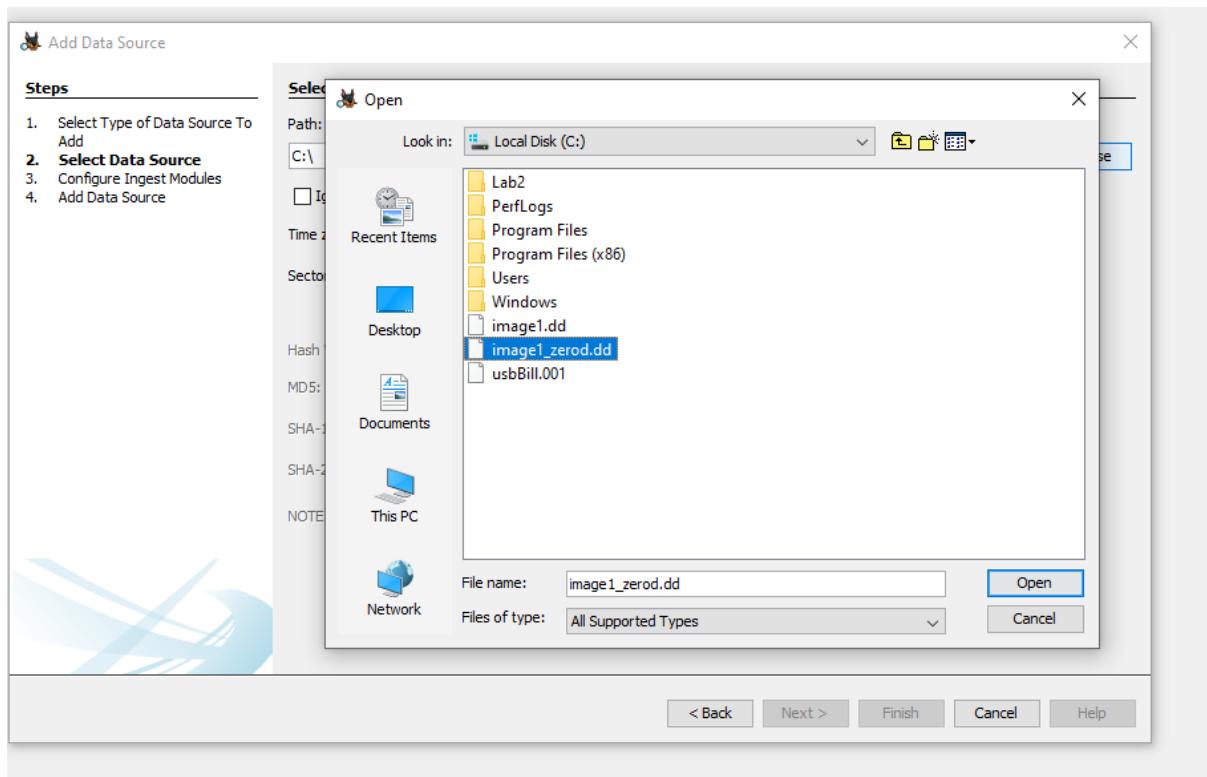


Figure 33: Adding raw image file of zeroed out suspect drive.

We must add the image as an “Unallocated Space Image File” this time, as there is no file system on the disk.

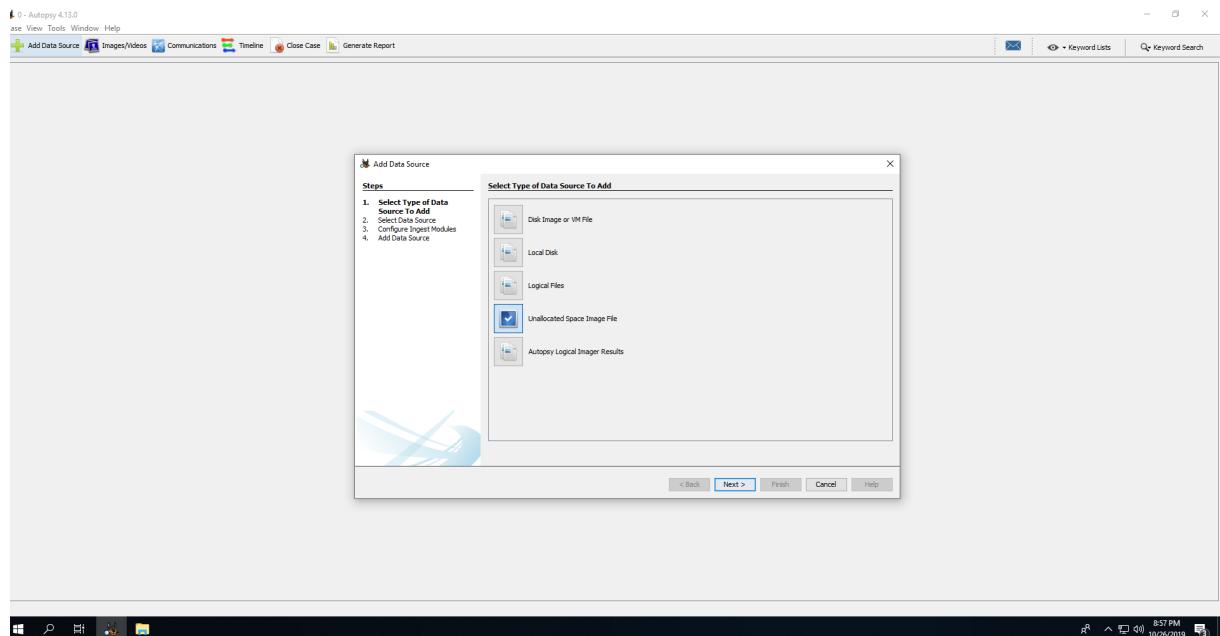


Figure 34: Select *Unallocated Space Image File* this time.

Now we can select `image1_zeroed.dd` and it will work.

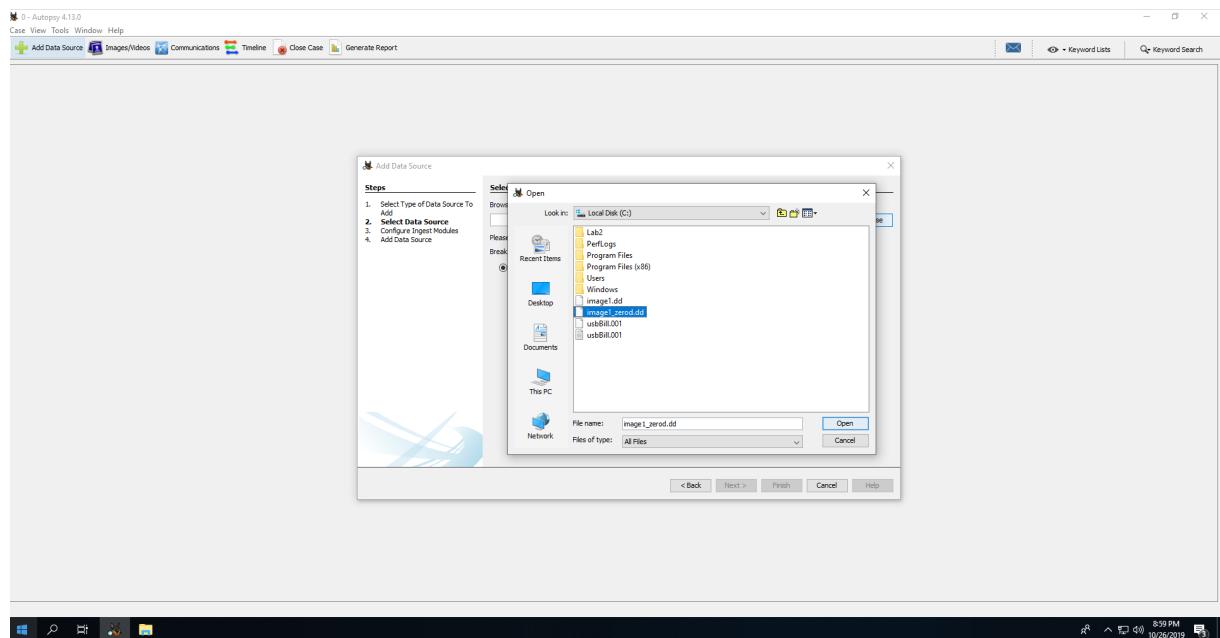


Figure 35: Selecting the zeroed-out image file for analysis.

Even after expanding the entire tree we see that there are no files on the disk, not even any deleted files.

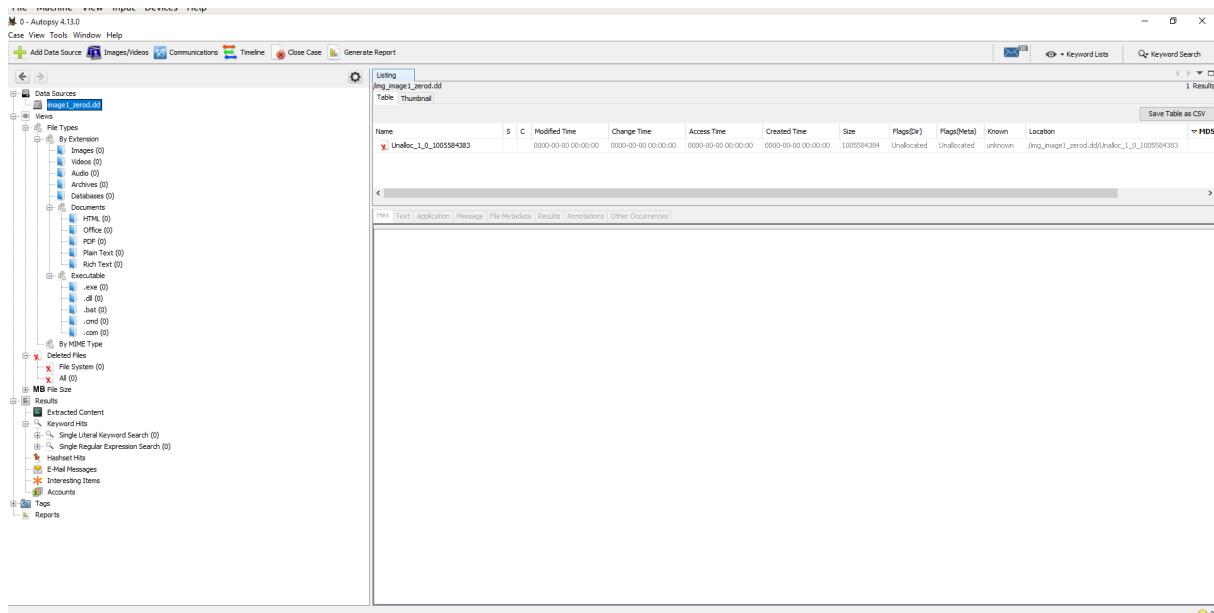


Figure 36: There aren't any files on the drive now, nor deleted files.

When we run our dirty word search once again, for the word *Warranties*, but nothing is found. There are no files on the drive, I didn't expect it would find anything.

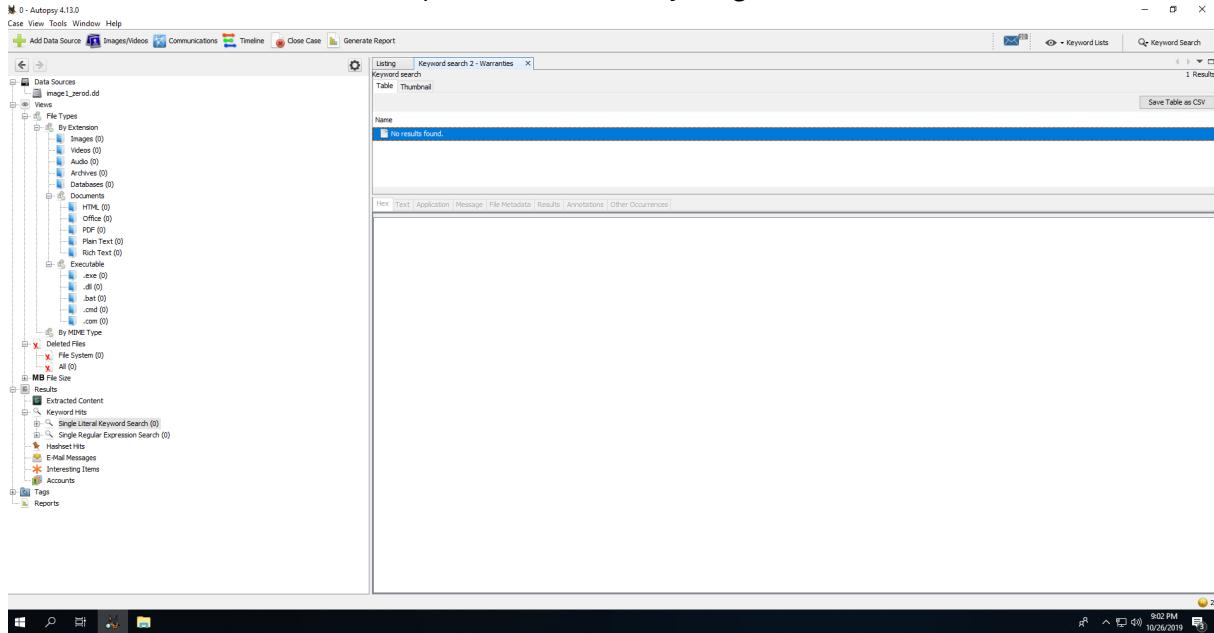


Figure 37: Nothing found in second dirty word search.

Task 9: Answer the questions. Please attach screenshots to prove your answers when necessary.**1. In Task 4, the acquired image has an extension of “.dd”. In Task 3, what is the extension for the image file?**

FTK uses an extension of .001. We can discover this by looking at the output of FTK Imager, and seeing the file we created is [usbBill.001](#).

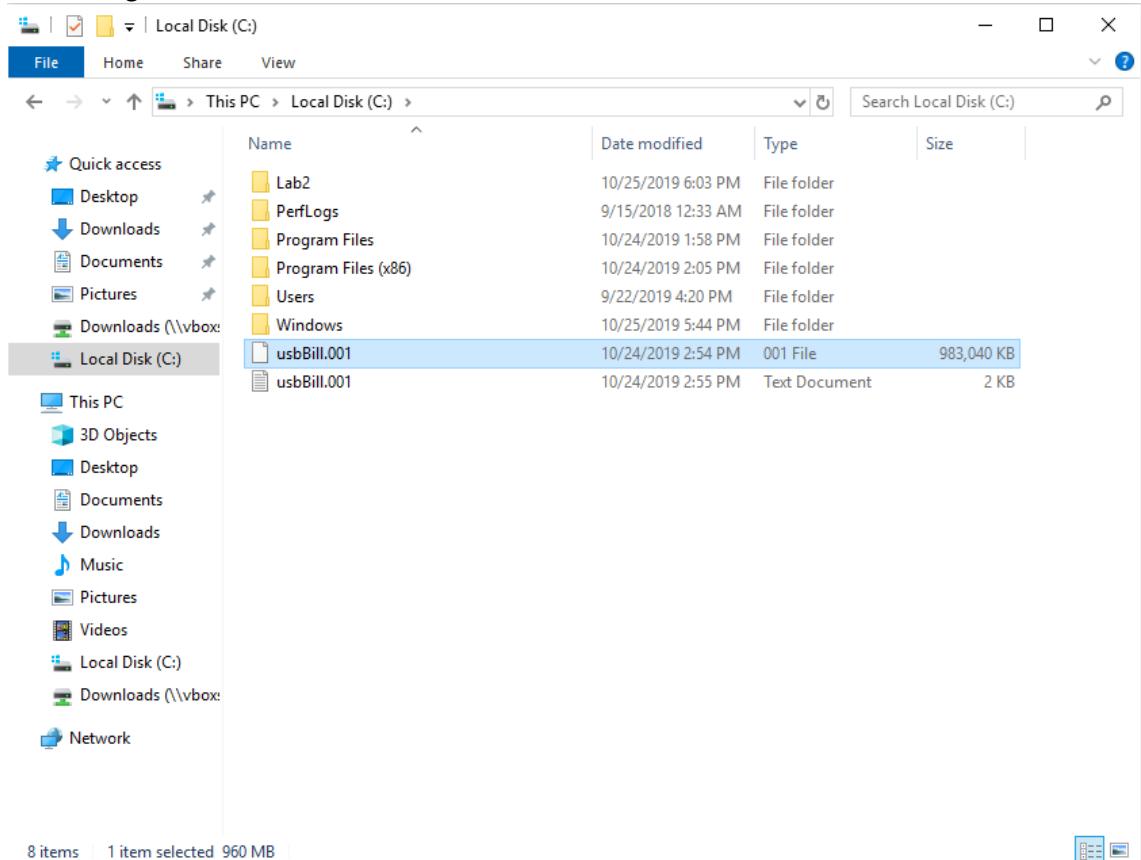


Figure 38: File created by FTK Imager has .001 extension, see [usbBill.001](#).

2. In Task 5, how many files are there on the USB drive? What are they?

- Six if you count deleted files, otherwise zero. Because the drive had been zeroed out twice due to previous activities in the class. All the files on the drive were deleted files, there were ** and all these files were .mft files except for the single .docx we created in task 2.

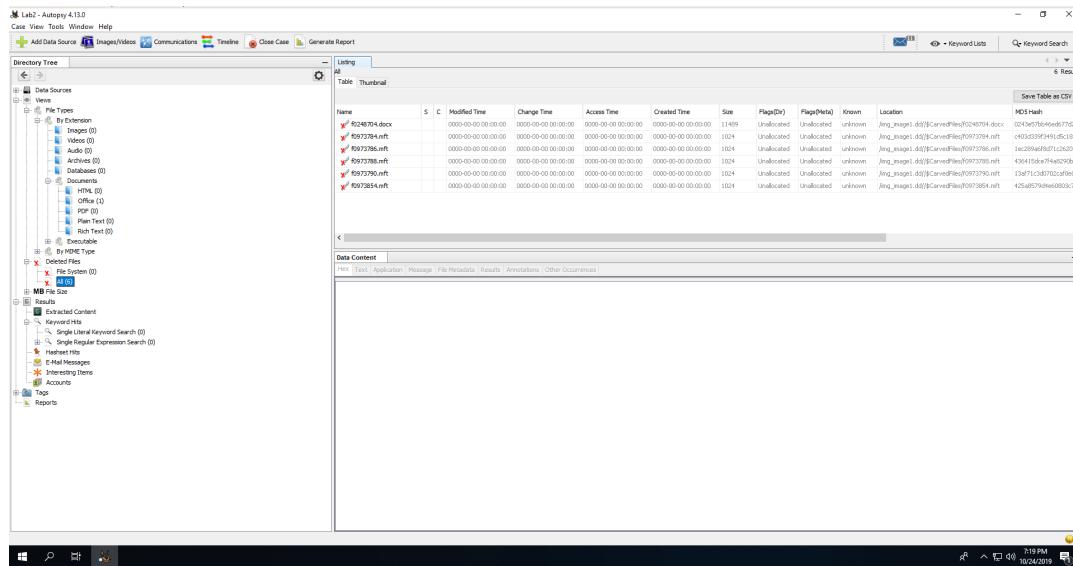


Figure 39: There are no files that aren't deleted existing on the drive.

3. In Task 5, which file/files are deleted?

- As I stated in question 2, because the drive had been zeroed out twice due to previous activities in the class. All the files on the drive were deleted files, and all these files were .mft files except for the single .docx we created in task 2.

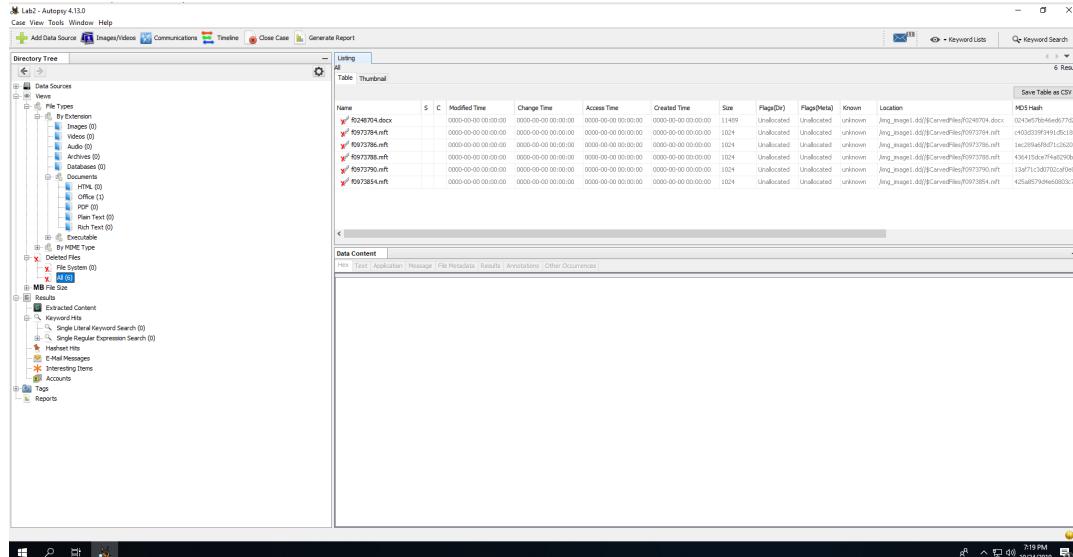


Figure 40: Deleted files found on the drive.

4. In Task 6, are you able to find any hit when you search *Warranties* as the key word? In which file is the key word located?

- I was able to get a hit for *Warranties*. The file the word was located in was f0248704.docx,

which was actually the remains of our `test.docx` we created before formatting the drive.

Name	Location	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Known	MD5 Hash
f0248704.docx	/img_image1.dd@\$carvedFiles/f0248704.docx	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	11489	Unallocated	Unallocated	unknown	0243e57bb46e6077d2a83

Figure 41: Search results for *Warranties*

5. In Task 8, how many files are there on the USB drive? What are they?

- There are no files on the drive at all after we've zeroed it out.

Name	\$	C	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Known	Location
Unalloc_1_0_1005584383			0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	1005584383	Unallocated	Unallocated	unknown	/img_image1_zeroed.dd@Unalloc_1_0_1005584383

Figure 42: No files exist on the drive after we've run a zero-out.

6. In Task 2, you performed a “disk format” operation towards the USB drive. Did this operation completely erase the “test.doc” (or “test.docx”) file in the USB drive? How do you

know?

- No, it did not delete the file completely. The MFT records for the file were deleted, the file is still recoverable though after the formatting, as you can see in the figure below.

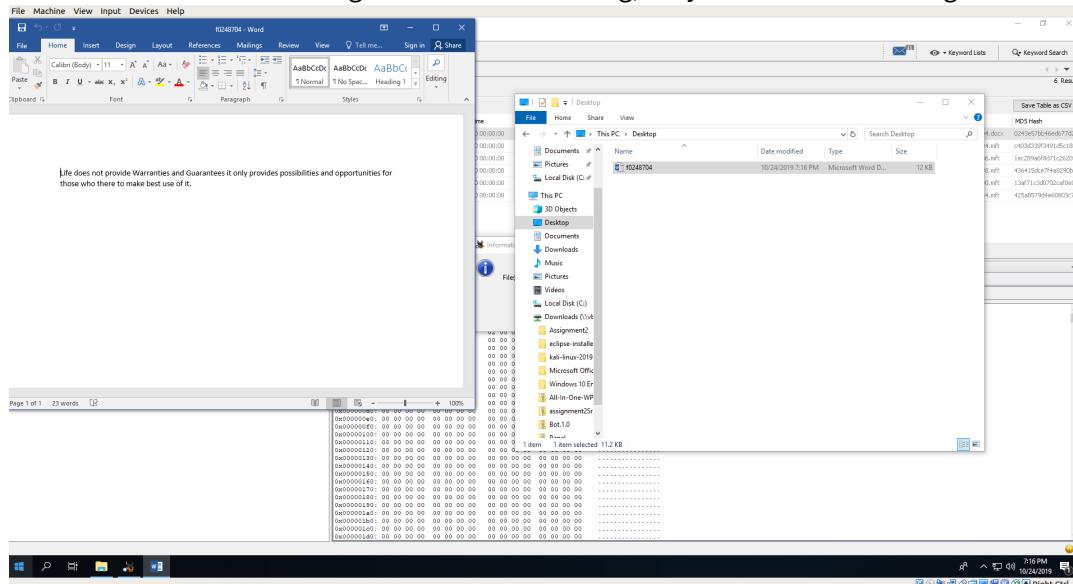


Figure 43: The contents of `f0248704.docx`, containing *Warranties*.

- 7. In Task 7, you performed a `zero out` operation towards the USB drive. Did this operation completely erase the `test.doc` (or `test.docx`) file in the USB drive? How do you know? Please provide a screenshot to prove your answer.**

- Yes, because there are no files on the drive at all after we've zeroed it out.

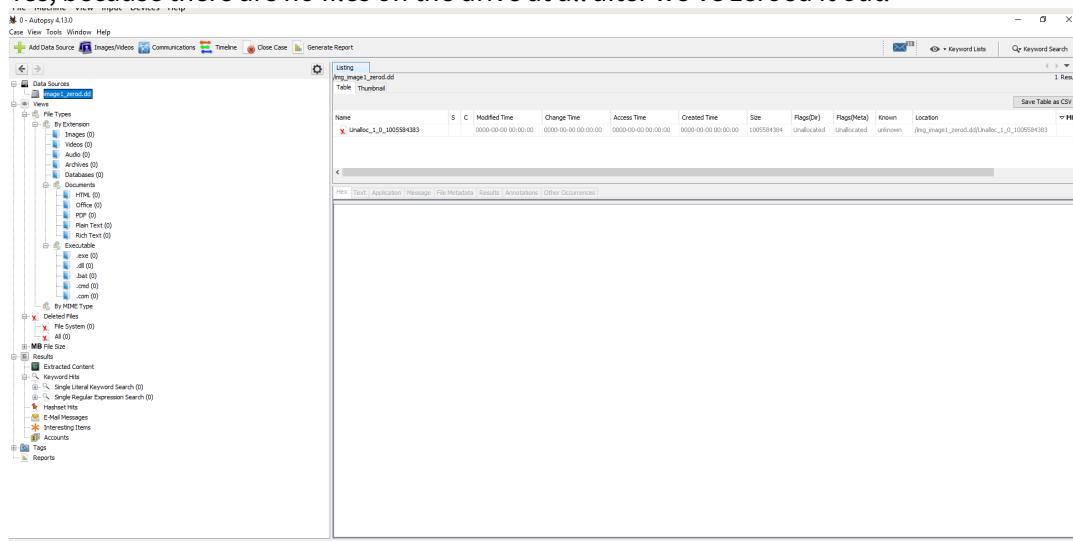
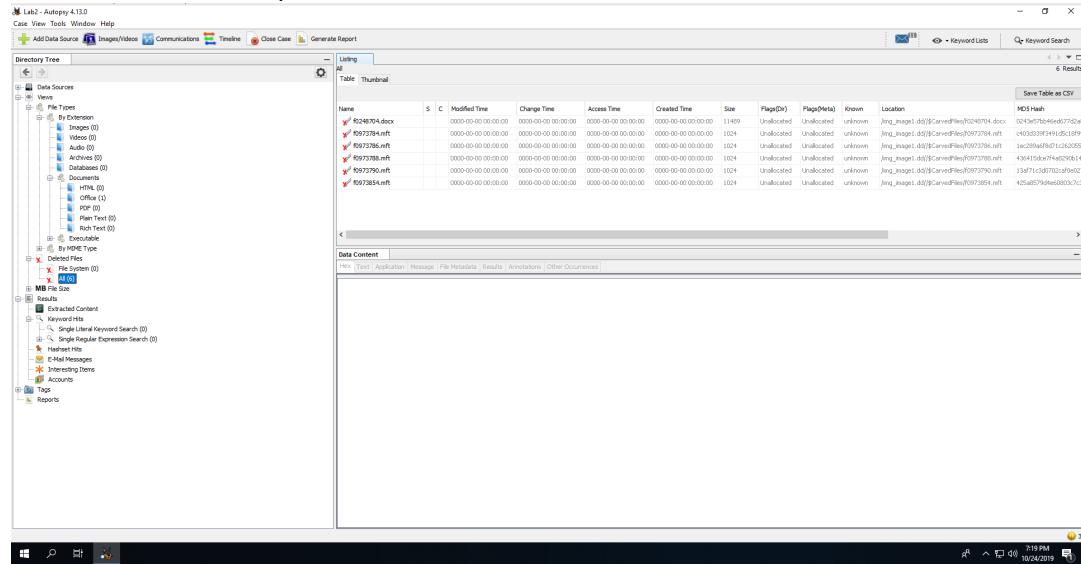


Figure 44: No files exist on the drive after we've run a zero-out

- 8. Did have any surprise in Task 5? Did you see any other files other than `test.doc` or `test.docx`.**

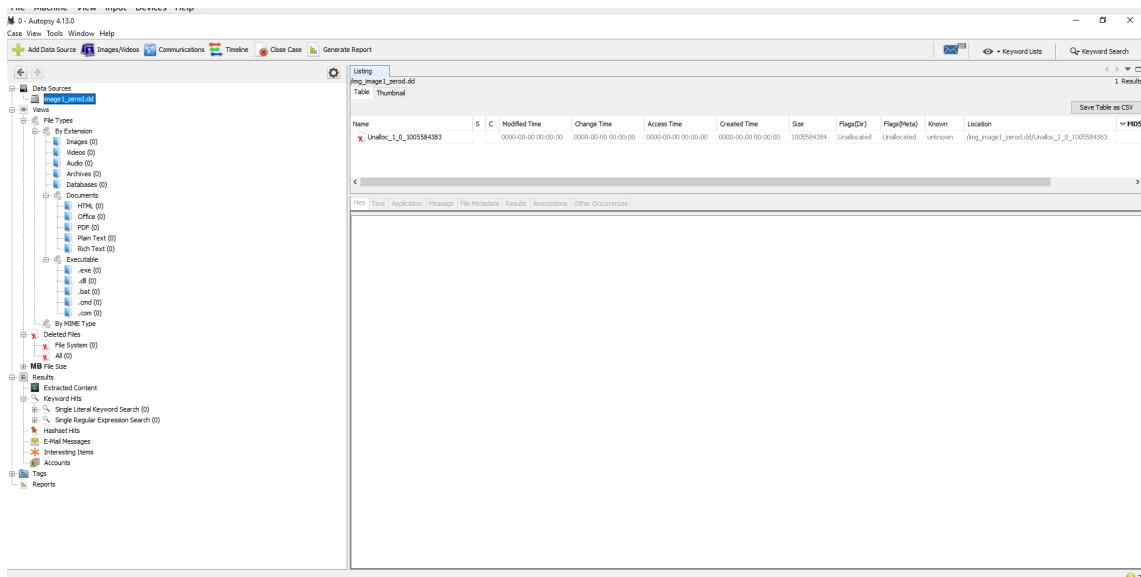
docx?

- The only files on the drive were the deleted remains of `test.docx` and some deleted `.mft` files. This wasn't a surprise.

**Figure 45:** We the only files on the drive are the 6 that were deleted.

- In Task 8, did you see any other files other than `test.doc` or `test.docx`? Please provide a screenshot to prove your answer.

- No, there were no files on the disk.

**Figure 46:** No files exist on the drive after we run a zero-out

- To summarize the questions above, what is the difference between disk formatting in Win-

What are the differences between formatting a drive and the zero-out operation?

- Formatting just deletes file system, so it no longer references the data, however the sectors on which the data is stored remain unchanged.
- The zero-out operation goes to each sector of the drive and sets the value of each bit to zero. This destroys all data that was stored in every sector on the drive.