Workbook
4-6

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

Exercise 4.1—Volume Shadow Copy Triage Acquisition

Background

Volume Shadow Copies (VSC) provide a rich view into historical forensic data for storage volumes. Days, weeks, or months of previously existing data is often held within them. Being comfortable locating and accessing data in Volume Shadow Copies is a critical skill for the digital forensics practitioner.

In this exercise, we will walk through how to locate VSCs, gain access to them via the command line, and perform a manual triage collection of files from VSCs. By leveraging the historical data available in VSCs, a more complete picture of an investigation is available when looking for answers.

We will also use Shadow Explorer, a GUI program, to interact with and export files from VSCs on a volume.

Objectives

- Identify Volume Shadow Copies (VSC) using VSSAdmin
- Manually mount a VSC to a symbolic link
- Use VSCMount to mount all available VSCs
- Export files of interest from mounted VSCs using PowerShell
- Use Shadow Explorer to interact with VSCs and export files from VSCs

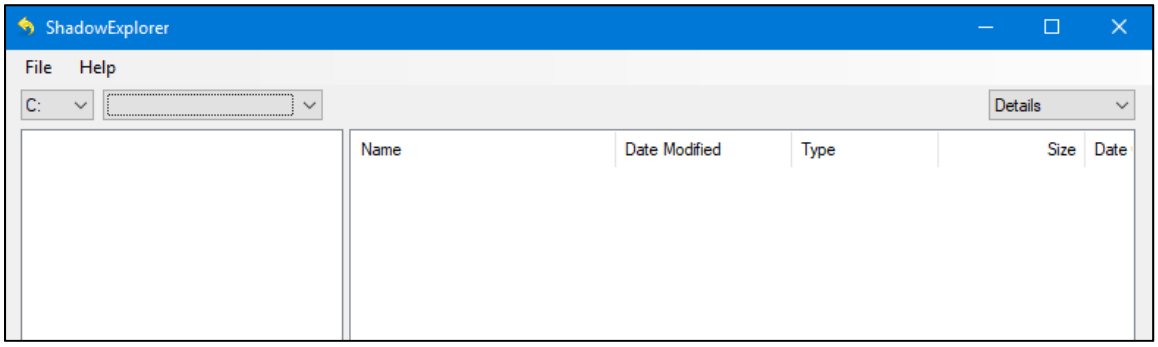
Exercise Preparation

1. Boot your **FOR498 Windows VM**

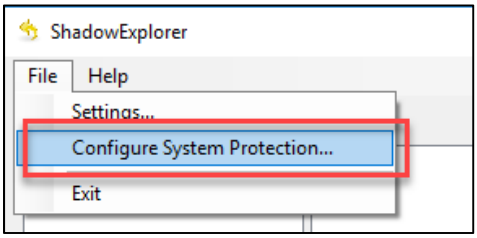
2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Verify **WindowsImage.E01** is mounted using **AIM**. Refer to the **Mounting Evidence** exercise for mounting instructions, if you need.

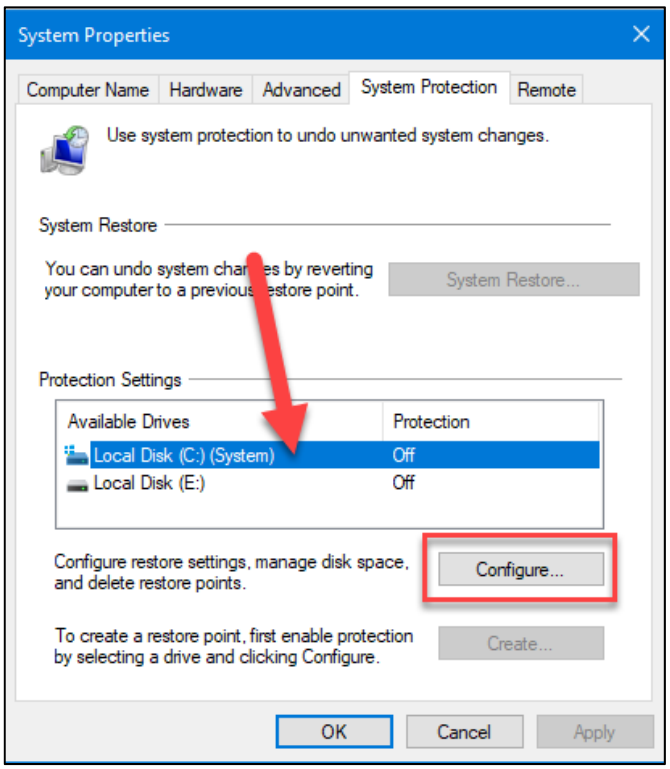
4. Double click the **Shadow Explorer** shortcut in the **Utilities** fence on the **Desktop** to start **Shadow Explorer**.



5. Before we start using **Shadow Explorer**, let's take a closer look at some of the system protection options related to **VSCs**. Click **File → Configure System Protection.**

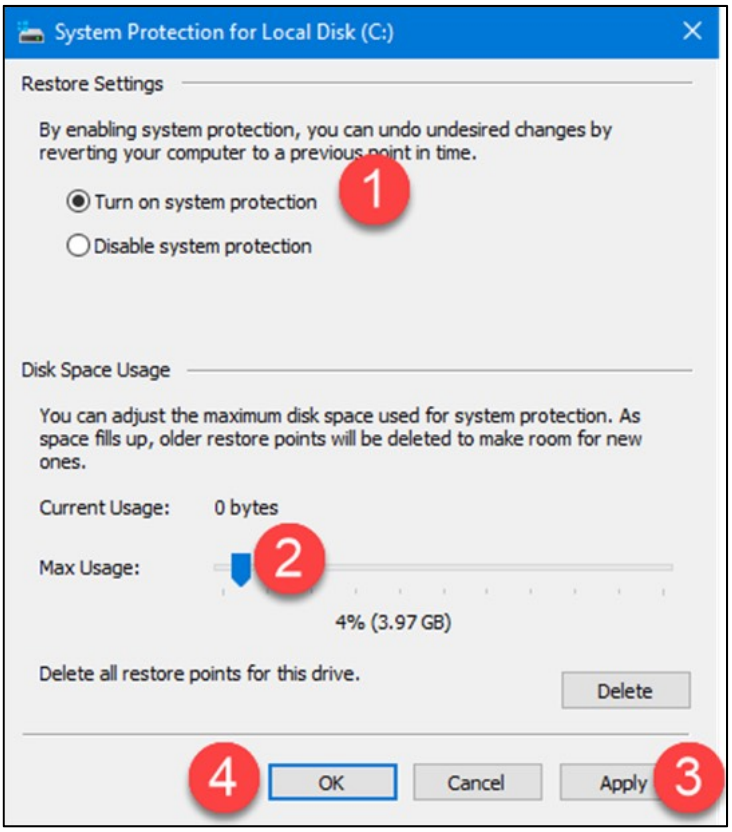


6. In the **Protection Settings** list, find the **C:** drive.

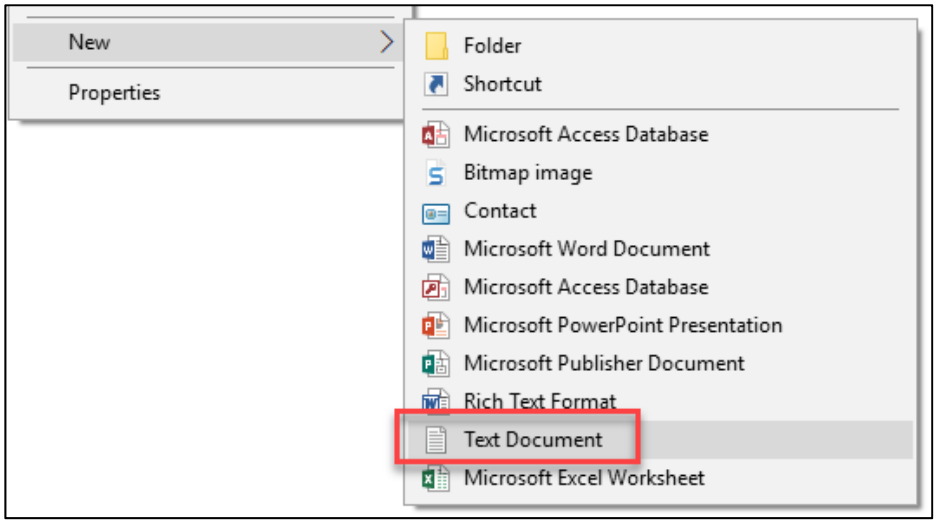


Once the **C:** drive is selected, click the **Configure** button and review the current settings for **System Protection** on the **C:** drive. Notice that **Protection** for the **C:** drive is currently **Off.**

7. Because shadow copies are turned off on the **VM**, enable them by configuring the settings as shown below. Set **Max Usage** to somewhere around **1-3%** of the drive size.



8. To get a better sense of how useful **VSCs** can be, let's create one manually. Before we do this however, we need to create a few new files so the Volume Shadow Copy we create will contain these new files. Open **File Explorer** and navigate to **C:\Temp** directory. Once there, right click on empty space, and choose **New -> Text Document** from the context menu.

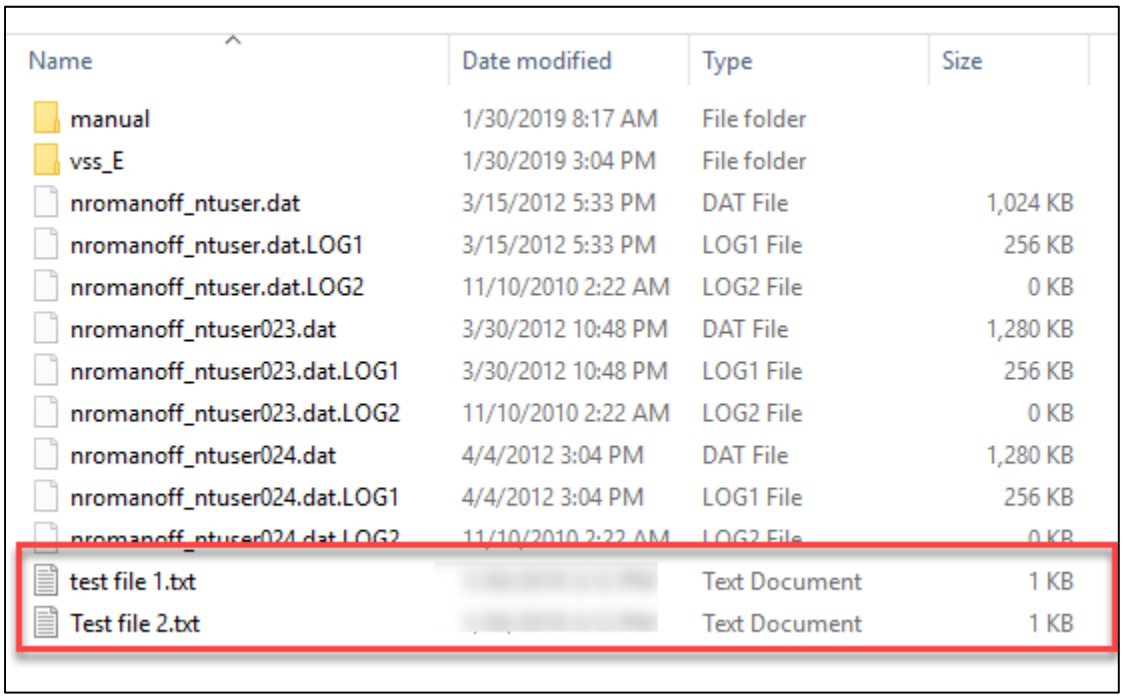


9. Using the previous steps, create two files as follows:

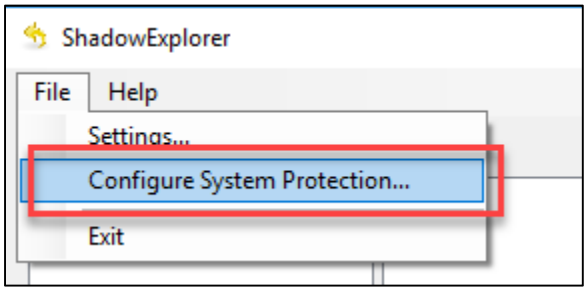
   a. Name the first text file **`test file 1.txt`**, then open it and enter your **first** and **last** name, then close the file, saving the changes.

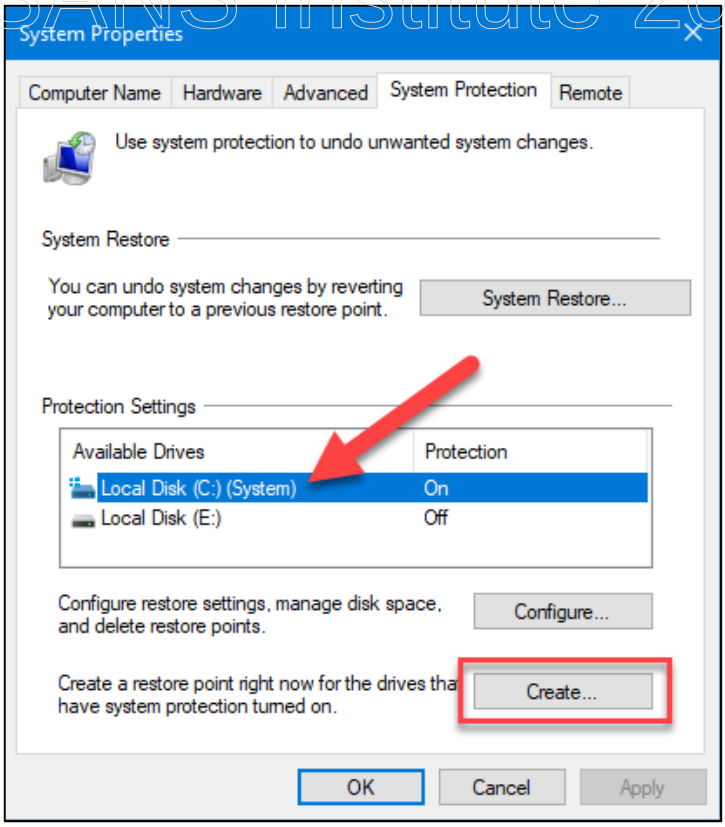
   b. Name the second file **`Test file 2.txt`**. Open this file and enter **FOR498**, then close the file, saving the changes.

   c. You should now have these two text files in your **C:\Temp** directory.

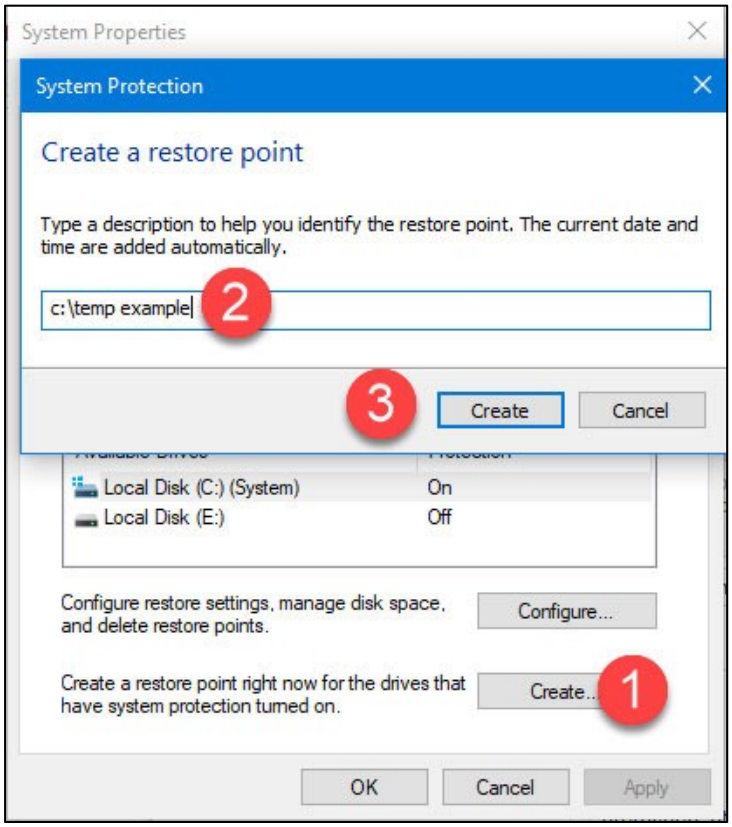


10. In **Shadow Explorer**, return to the **System Properties** window and make sure the **System Protection** tab is selected. Once here, make sure the **C:** drive is selected. If the **System Properties** window is not visible, open it again via the **File** menu in **ShadowExplorer**.

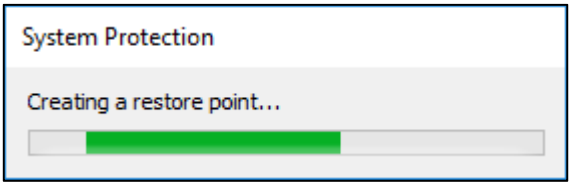




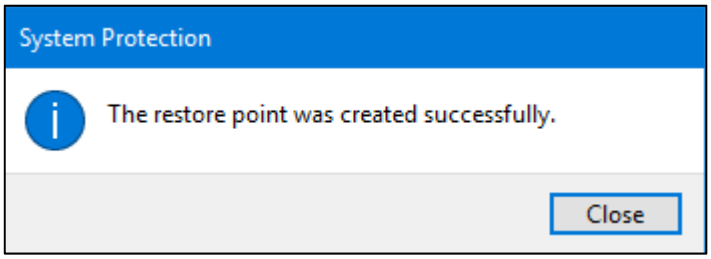
11. Click the **Create…** button to manually create a **VSC**. In the **Create a restore point** dialog, enter a description of **`c:\temp example`**, then click **Create**.



Windows will now create a new **VSC** and display a dialog box.



When the **VSC** is created, click the **Close** button.



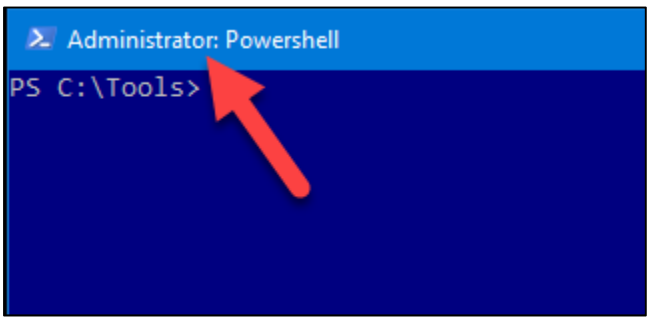
Finally, click the **OK** button in the **System Properties** dialog to close the window.

Record the current system time below. We will use this time to find the **VSC** we created above in a subsequent step.

_____

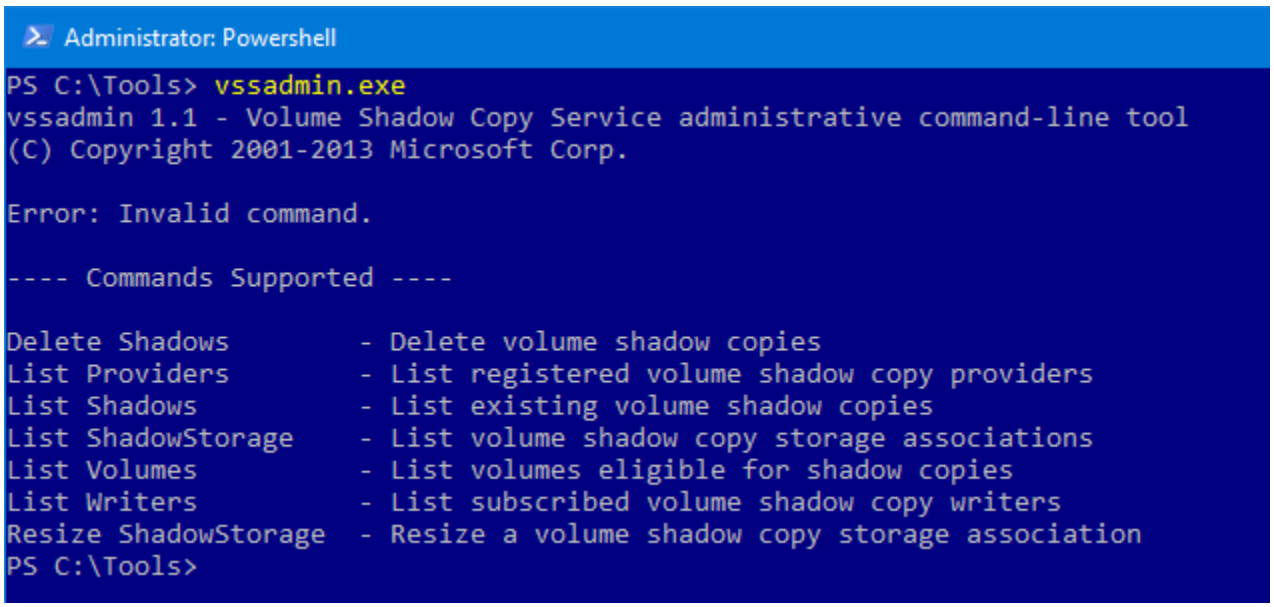
12. Close **Shadow Explorer** by clicking the **X** in the upper right corner.

13. Open a new **PowerShell** window using the shortcut on the **Desktop**. Administrator rights are required because the **vssadmin** tool we will be using requires admin rights to run properly.



14. In the **PowerShell** window, type the following and press **Enter**.

```
vssadmin.exe
```

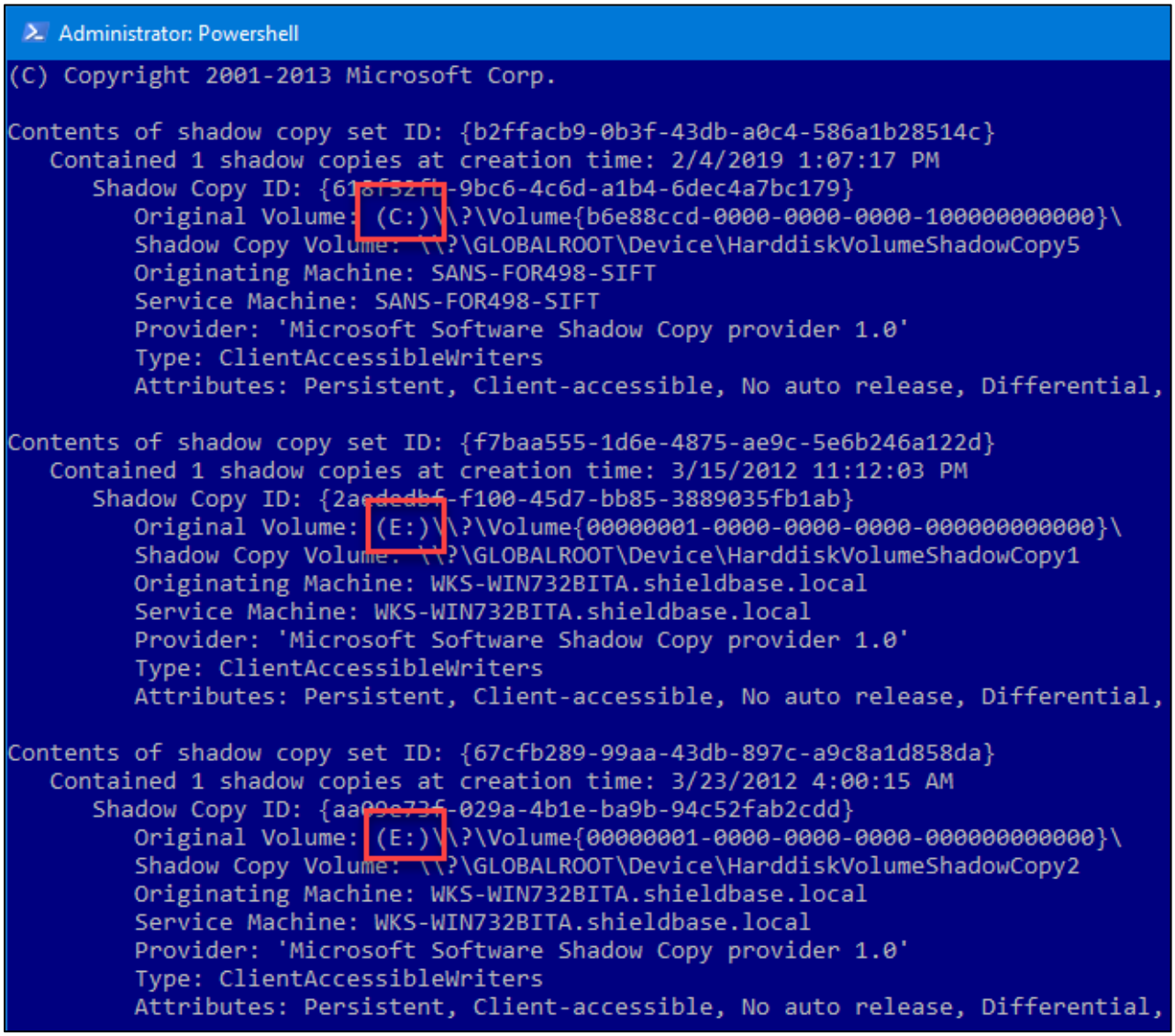


Notice that **vssadmin** displays an error about an invalid command, but also lists the available commands. We will use some of these commands going forward to interact with different volume shadow copies.

15. In the **PowerShell** window, use the **up** arrow to bring up the previous command and adjust the command as shown below, then press **Enter**.

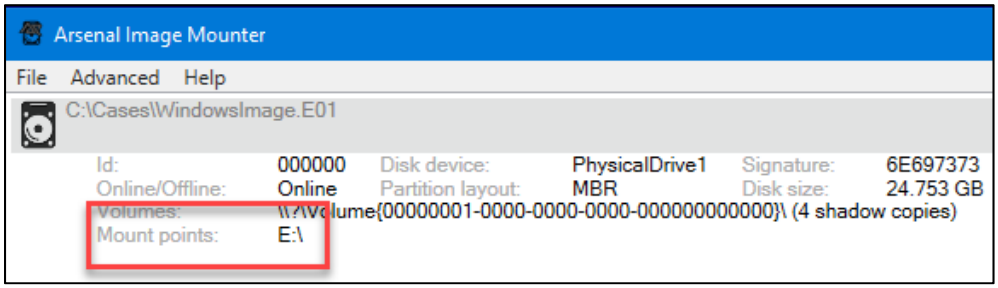
```
vssadmin.exe List Shadows
```

You will see the screen below…



The output on your system may look slightly different in your **VM** than what is shown above, but the important things to notice here are that we can see the **Shadow Copy ID**, its **creation time**, the **Shadow Copy Volume** name, the **Original Volume**, and the **Originating Machine** (i.e. where the **VSC** was generated).

The **E:** drive in the above screenshot reflects the drive letter where **AIM** mounted our E01. The drive letter **AIM** used can be seen in the **AIM** interface.

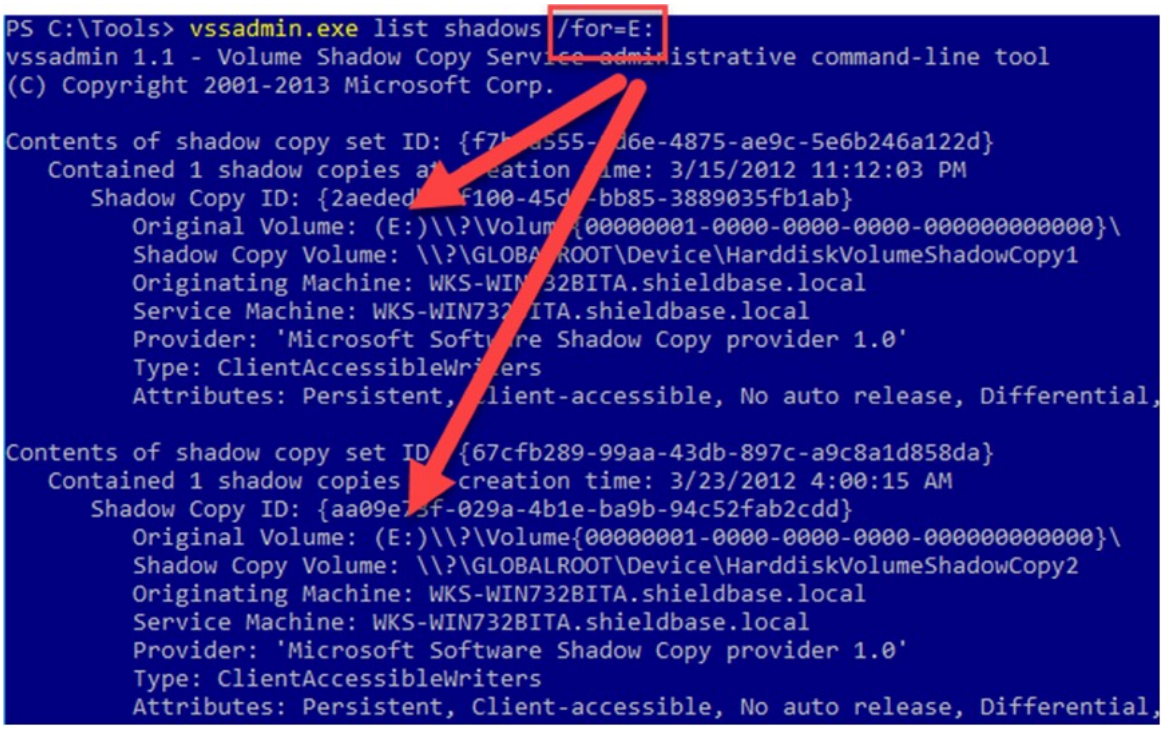


Depending on the machine you run this command on, you may see entries for several different drive letters (C:, E:, or J: for example). This is because **vssadmin** displays all **VSCs** for all available partitions by default. In most cases, we will only be interested in the **VSCs** found on one of the hard drives. Recall from our preparation steps, we used **AIM** to mount an E01. In the example output above, that E01 was mounted as the **E:** drive. Depending on what you have connected to your **VM**, you may have a different drive letter.

16. Rather than look at all the **VSCs** for all volumes, let's use **vssadmin** again to get back information about VSCs located on the **E:** drive. To do this, we need to add another argument to the command line, **`/for=<driveletter>`**, which filters what is shown to only include **VSCs** from <**driveletter**>. For the **E:** drive, it would look like what is shown below.

    Be sure to adjust the command to reflect the drive letter shown in **AIM** for your machine.

```
vssadmin.exe List Shadows /for=E:
```

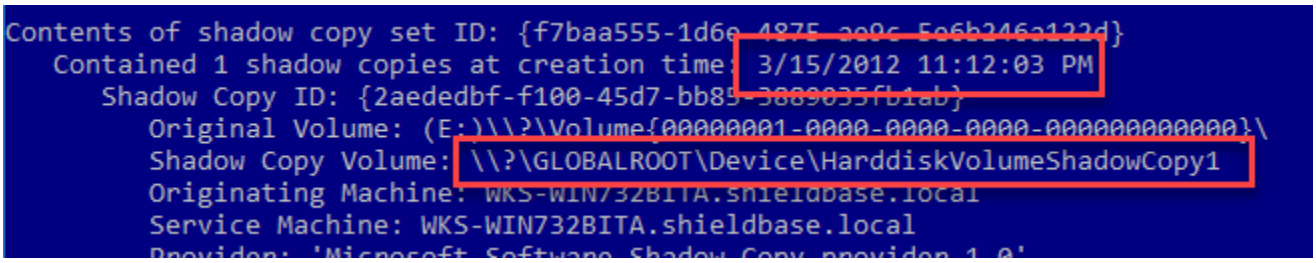


Now that we know the details for the **VSCs** on our drive of interest, let's get access to the contents of one of them.

17. To access a **VSC** manually, we need to map the **Shadow Copy Volume** to a directory. The tool we will use to do this is called **mklink**, which is built into the Windows Command Prompt. Because we already have a **PowerShell** window open, we will use that to make our symbolic link by simply invoking **cmd** and passing in the appropriate command. The syntax looks like this:

```
cmd.exe /c mklink /D <MountDirectory> <ShadowCopyVolume>
```

Where **<*MountDirectory*>** is the directory where we want the **VSC** to show up on our file system and **<*ShadowCopyVolume*>** is the value of "**Shadow Copy Volume**" property from **vssadmin** output:



PRO TIP: It is critical to have a trailing backslash at the end of the Shadow Copy Volume name. The command will not work without it! For example, in the command we run next, the Shadow Copy Volume name is the following (notice the trailing backslash):

\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1

NOTE: Verify that the **Shadow Copy** you are working with has a creation time of **3/15/2012 11:12:03 PM.** On your **VM**, the **Shadow Copy Volume** name for this particular **VSC** may end with a different number other than "1".

Once you find the correct shadow copy, select the **Shadow Copy Volume** value (shown in the red box above) with the mouse, then press **Enter** to copy it to the clipboard.

Putting it all together, we can mount the **VSC** from the previous stage using this command. Note that the command should all be on one line, with a space after '**vsc**' and before '****':

```
cmd /c mklink /D C:\Temp\vsc
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
```

18. When you are ready to add the **\\?\GLOBALROOT** value, right-click the mouse to paste the value you previously copied to the clipboard. Be sure to add the trailing slash at the end!

```
PS C:\Tools> cmd /c mklink /D C:\temp\vsc \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
symbolic link created for C:\temp\vsc <<===>> \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
PS C:\Tools>
```

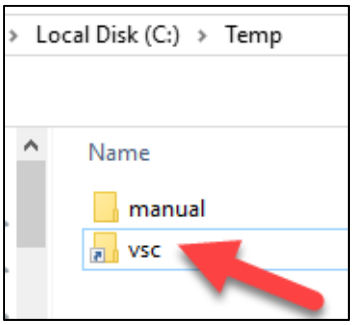
The message generated from the command indicates that **VSC1** is now mounted on the **C:\Temp\vsc** directory. Any directory naming convention can be used, such as **C:\vss\1**, which would indicate the **VSC** that the mount directory represents, etc.

Note: If you forgot the trailing backslash, delete the symlink you created with the following command:

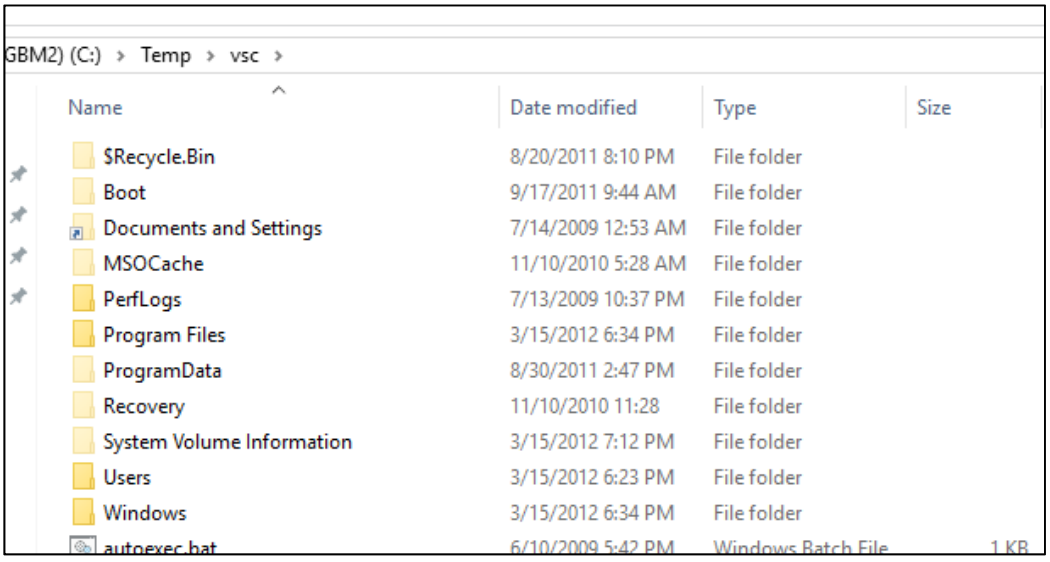
```
cd C:\Temp; cmd /c rmdir C:\Temp\vsc
```

Then try again, making sure to add the backslash.

19. To verify the **VSC** is accessible, open **File Explorer** and navigate to **C:\Temp**.



Double click on **vsc** and the contents of the **VSC** will be shown.

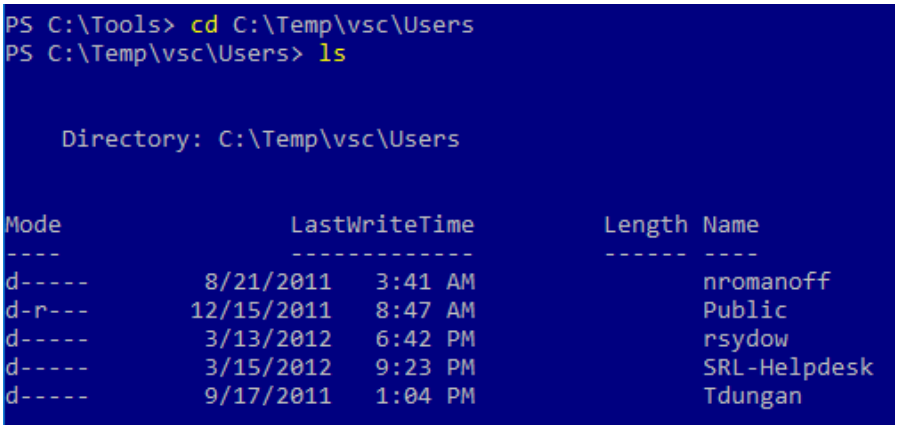


In some cases, **File Explorer** will not allow you to navigate into directories due to permission issues. We will get around this issue by using **PowerShell** to navigate around the **VSC** and extract files.

For example, try double-clicking the **Users** folder and then **NRomanoff**. You should get a permissions error.

20. In **PowerShell**, type the following commands, pressing **Enter** after each.

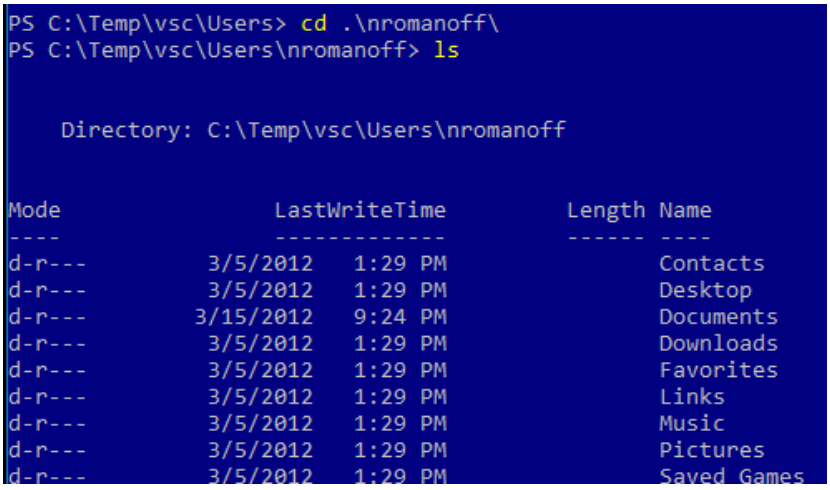
```
cd C:\Temp\vsc\Users
ls
```



This results in a listing of all the files and directories that exist in the current working directory.

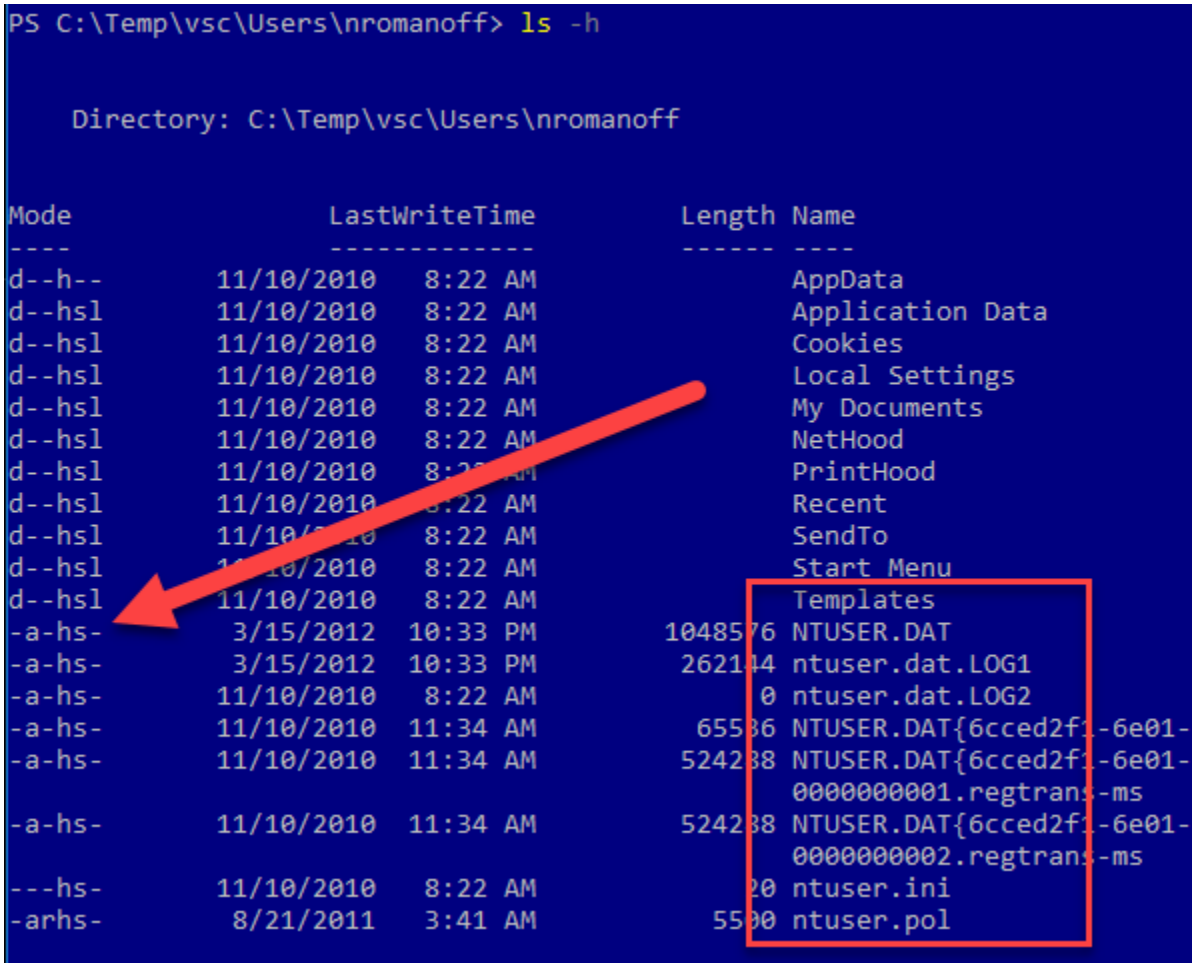
21. Type the following commands and press **Enter** after each to see what is in the **nromanoff** directory. Recall that you can use **TAB** completion to save some typing, so entering **`cd nr<TAB>`** would complete the path for you. Once you are in the directory, issue another **`dir`** or **`ls`** command.

```
cd nromanoff
ls
```



22. If you recall from the **Preparing the Analyst Machine** lab, we took steps to show system and hidden files, but in the last step, we only see directories listed. This is because, by default, **PowerShell** does not show files with the system or hidden attribute. To see hidden files, we need to use a switch that tells **PowerShell** to show hidden files, **`-h`**, which can be used with either the **`ls`** or **`dir`** command.

```
ls -h
```



Notice that we now see several additional files, and the **Mode** column reflects the fact that these files are hidden (h) and system (s) files.

23. The copy command can be used to recover a file from the current location. Making a copy of nromanoff's **ntuser.dat** Registry hive can be accomplished with the command:

```
copy ntuser.dat C:\Temp\nromanoff_ntuser.dat
```

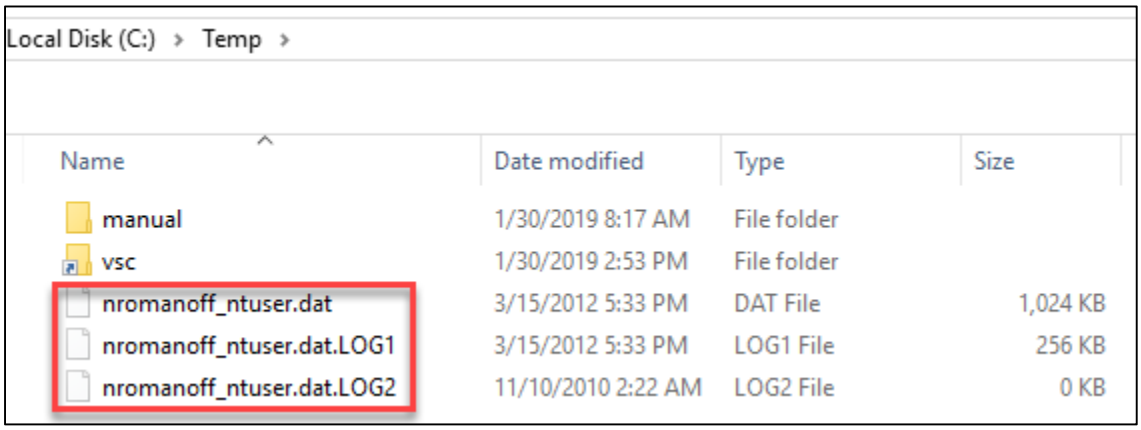
In this example, we are saving **ntuser.dat** to a different directory and filename so that we know who the **ntuser.dat** file belongs to.

Since we are copying a Registry hive, let's grab the **LOG** files too:

```
copy NTUSER.DAT.LOG1 C:\Temp\nromanoff_ntuser.dat.LOG1
copy NTUSER.DAT.LOG2 C:\Temp\nromanoff_ntuser.dat.LOG2
```

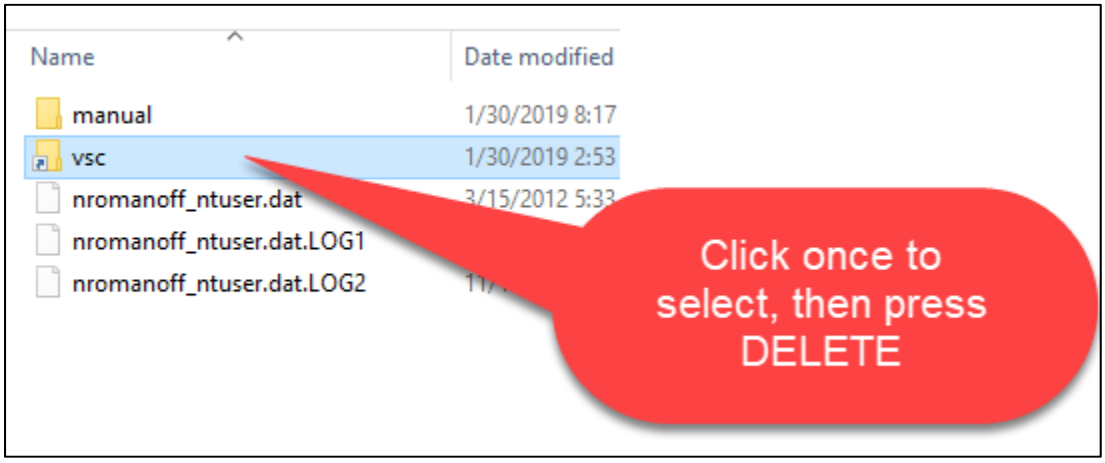
This same pattern can be repeated for any additional files you need to export from the **VSC**.

24. In **File Explorer**, navigate to **C:\Temp** and verify the new files are visible.



These files are now available and will continue to be available even after the **VSC** is unmounted.

25. To unmount the **VSC**, use **File Explorer** to select the **C:\Temp\vsc** directory and press the **Delete** key.



26. In the previous stage, we manually located and created a symbolic link to a single **VSC**, but in many cases, it is necessary to access more than one **VSC** for a drive. While the same manual method could be used to create symbolic links for each **VSC**, this becomes tedious when there are more than a few.

27. Start a new instance of **PowerShell** via the shortcut on the **Desktop**. Then run the **VSCMount.exe** command (note that we are in the **C:\Tools** directory).

```
.\VSCMount.exe
```

```
PS C:\tools> .\VSCMount.exe

VSCMount version [redacted]

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/VSCMount

        dl              Source drive to look for Volume Shadow Copies (C, D:, or F:\ for example)
        mp              The base directory where you want VSCs mapped to
        ud              Use VSC creation timestamps (yyyyMMddTHHmmss) in symbolic link names. Default is FALSE
        debug           Show debug information during processing
```

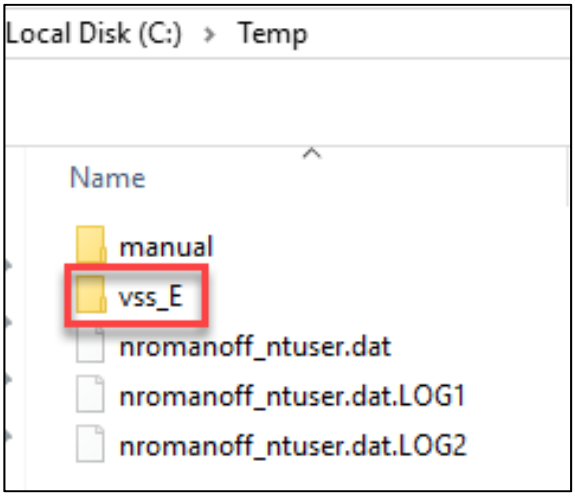
28. To make all the **VSCs** on the **E:** volume available, execute the following command:

```
.\VSCMount.exe --dl E: --mp C:\Temp\vss
```

```
PS C:\Tools> .\VSCMount.exe --dl E: --mp C:\Temp\vss
VSCMount version [redacted]

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/VSCMount

Command line: --dl E: --mp C:\Temp\vss

Creating directory 'C:\Temp\vss_E'
Mounting VSCs to 'C:\Temp\vss_E'

VSCs found on volume E: 4. Mounting...
        VSS 1    (Id {2aededbf-f100-45d7-bb85-3889035fb1ab}, Created on: 2012/03/15 23:12:03 UTC) mounted OK!
        VSS 2    (Id {aa09e73f-029a-4b1e-ba9b-94c52fab2cdd}, Created on: 2012/03/23 04:00:15 UTC) mounted OK!
        VSS 3    (Id {56b2ab57-269f-4cf6-bf25-422a6af5c2ed}, Created on: 2012/03/31 04:00:12 UTC) mounted OK!
        VSS 4    (Id {748b9466-321c-4df9-8b9b-a70f222f334a}, Created on: 2012/04/04 20:05:02 UTC) mounted OK!

Mounting complete. Navigate VSCs via symbolic links in 'C:\Temp\vss_E'

To remove VSC access, delete individual VSC directories or the main mountpoint directory
```

This command has done several things for us, including identifying all available **VSCs** on the **E:** drive, creating (if need be) the directory to mount all the **VSCs** to, and finally, mounting each **VSC** onto its own directory.

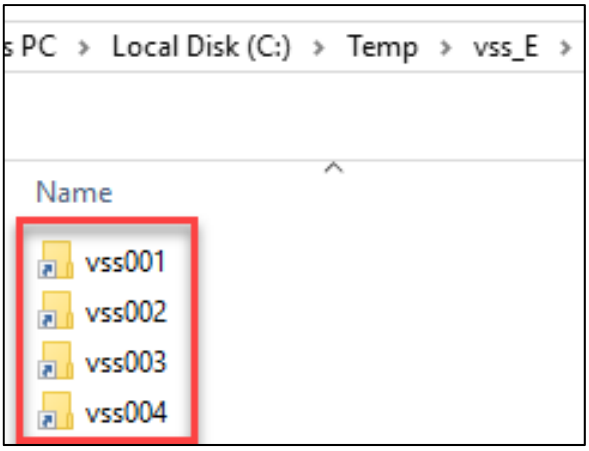
29. Open **File Explorer** and navigate to **C:\Temp**



VSCMount appends the source drive letter to the mount point provided to give context as to which drive letter the **VSCs** originated from.

30. In **File Explorer**, double click on the **vss_E** directory and you will see a symbolic link for each available **VSC**, along with the **VSC** that the link points to.

    NOTE: The **VSS** numbers will most likely be different on your machine compared to what is shown below!



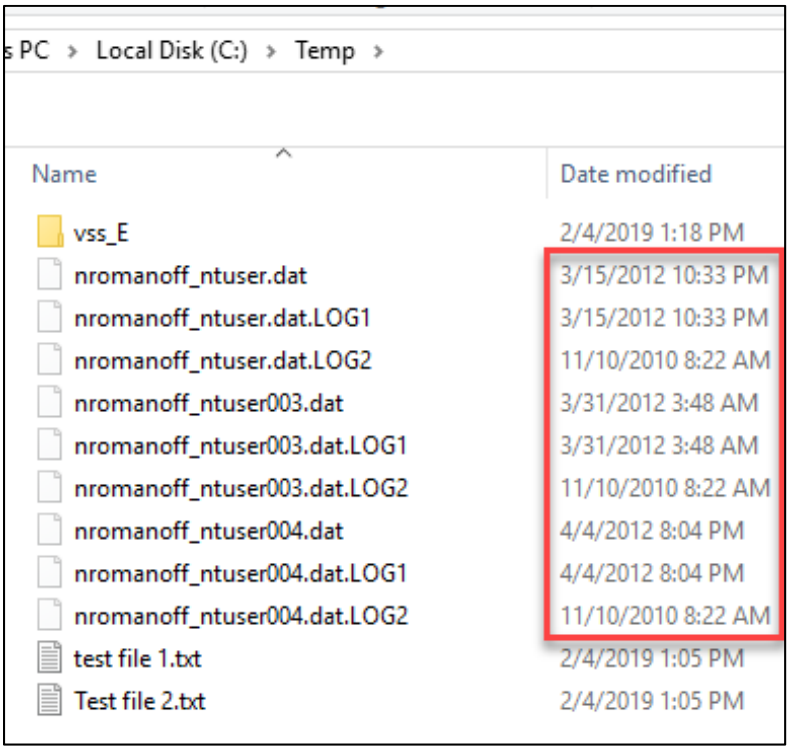
31. Each of these **VSC** mount points can be interacted with exactly as we did in the previous stage of the exercise.

    Following the example from the previous stage, copy nromanoff's **ntuser.dat** registry hives and logs from **vss003** and **vss004** and save the files to **C:\Temp**. it is often a good idea to indicate which **VSC** a file came from, so adding this information to the destination file is helpful.

    NOTE: Adjust these paths accordingly to match up with the **VSC** numbers that are shown on your computer. If you do not have **vss003** and **vss004**, use the last two that are shown and adjust the paths accordingly.

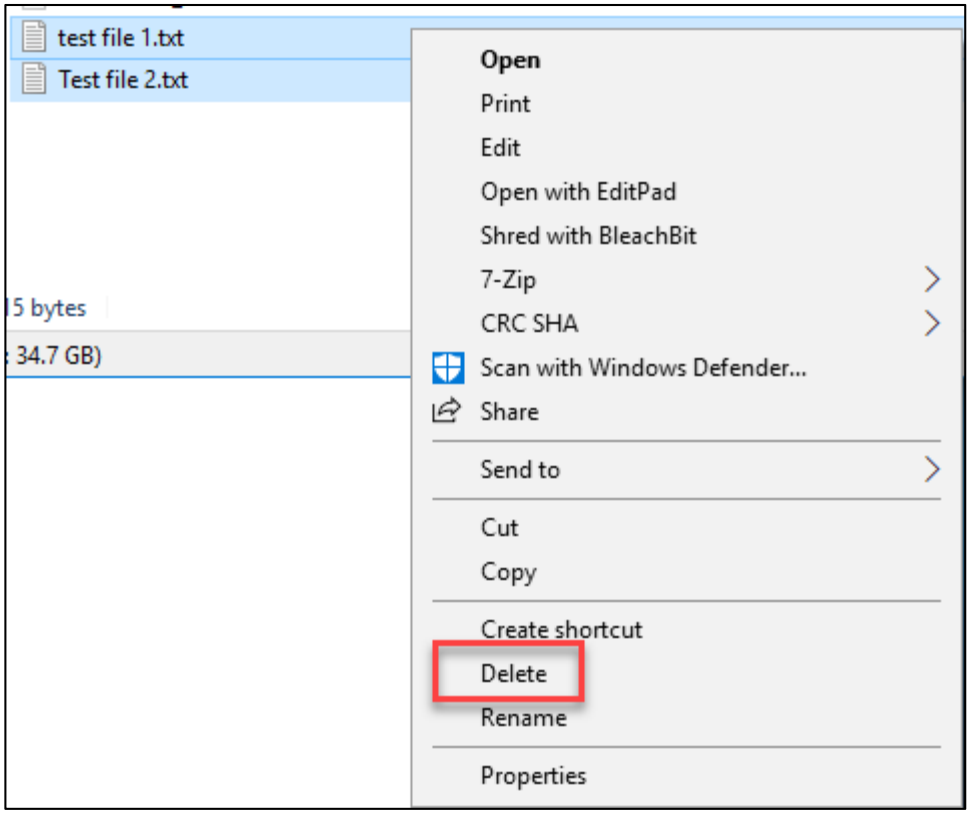
```
cd C:\Temp\vss_E\vss003\Users\nromanoff\
copy ntuser.dat C:\Temp\nromanoff_ntuser003.dat
copy NTUSER.DAT.LOG1 C:\Temp\nromanoff_ntuser003.dat.LOG1
copy NTUSER.DAT.LOG2 C:\Temp\nromanoff_ntuser003.dat.LOG2

cd C:\Temp\vss_E\vss004\Users\nromanoff\
copy ntuser.dat C:\Temp\nromanoff_ntuser004.dat
copy NTUSER.DAT.LOG1 C:\Temp\nromanoff_ntuser004.dat.LOG1
copy NTUSER.DAT.LOG2 C:\Temp\nromanoff_ntuser004.dat.LOG2
```

32. Open **File Explorer** and navigate to **C:\Temp**. Verify the copies were successful.

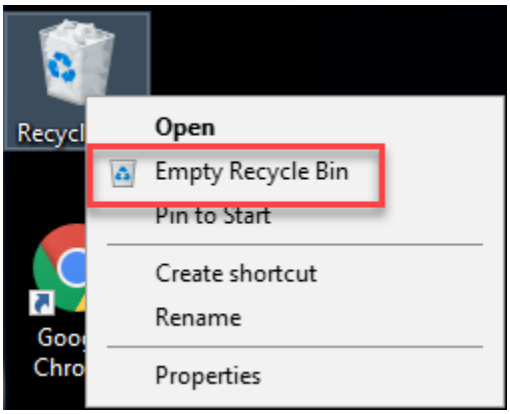


PRO TIP: Remember, a **VSC** points to a specific point in time for a given volume. Notice how this is reflected in the **Date modified** column in that each of the copies of nromanoff's **ntuser.dat** has a different last modified date.

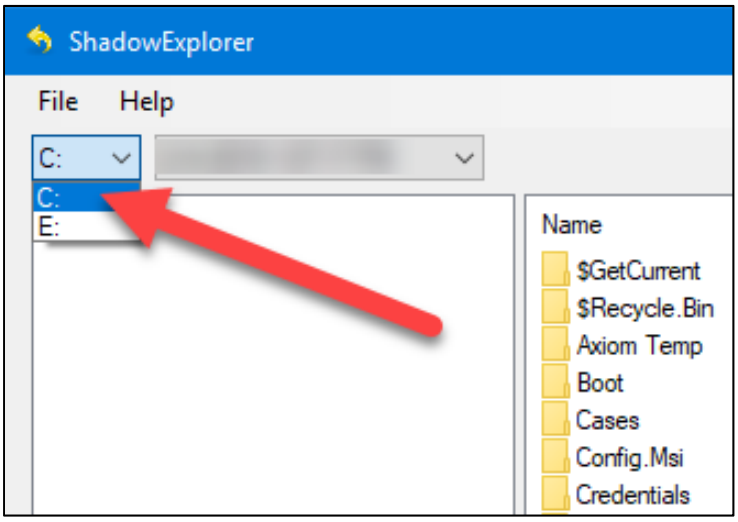
33. Return to **File Explorer**, and make sure you are in the **C:\Temp** directory. Select both text files we created earlier (**test file 1.txt** and **Test file 2.txt**) and delete them via the context menu, or by pressing the **Delete** key.



34. Empty your **Recycle Bin**.

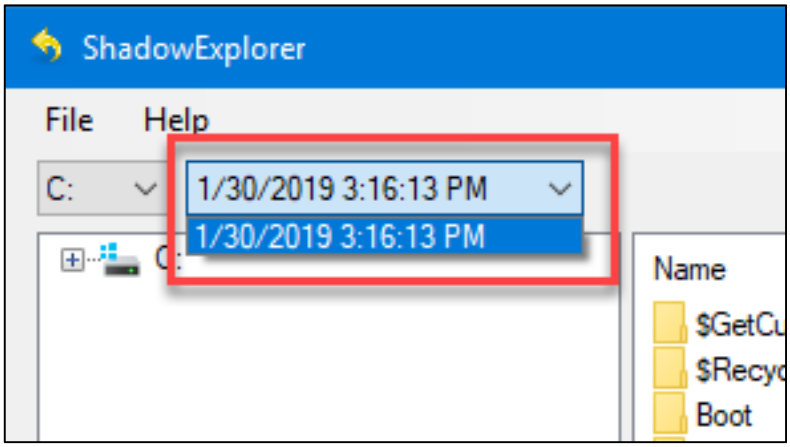


35. Start **Shadow Explorer** via the shortcut in the **Utilities** fence on the **Desktop**, click the drop-down in the upper left, and select **C:** from the list.

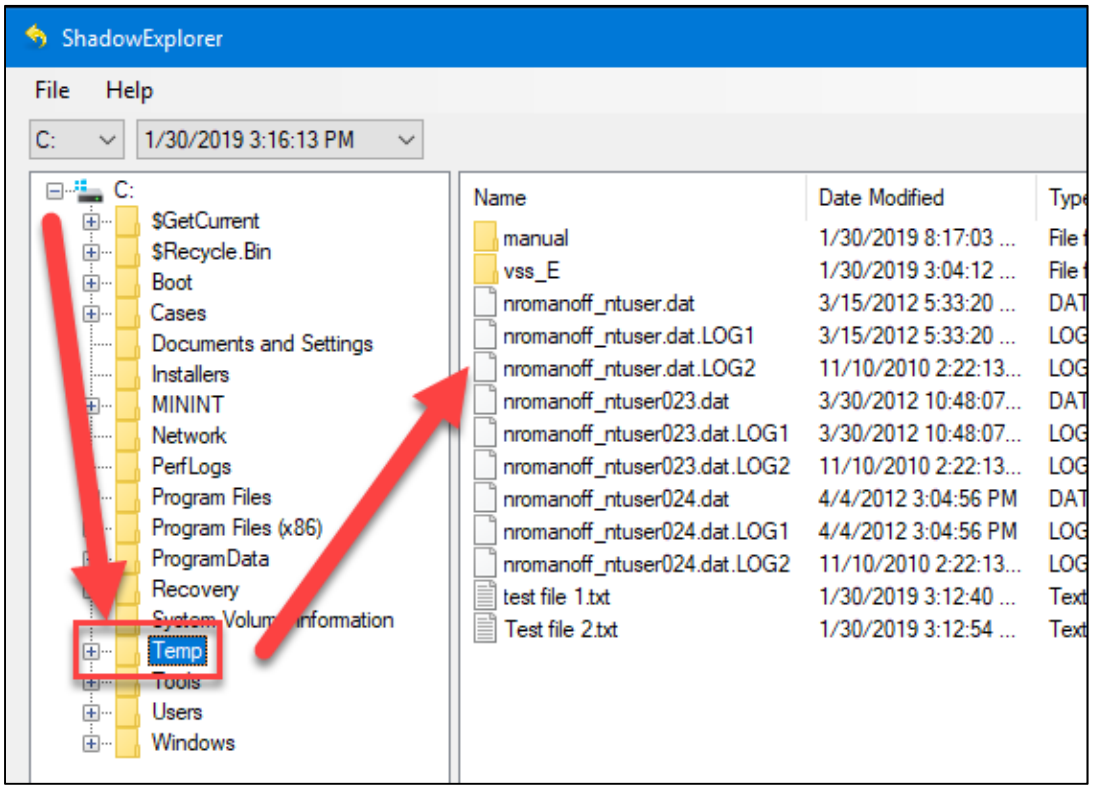


36. Once the **C:** drive is selected, the rightmost drop-down will contain a list of all available **VSCs** for the **C:** drive (your time stamp will be different than what is shown below). Select the newest **VSC** available from the list.

    NOTE: The timestamp should match very closely to the system time you recorded in the **Exercise preparation** step.

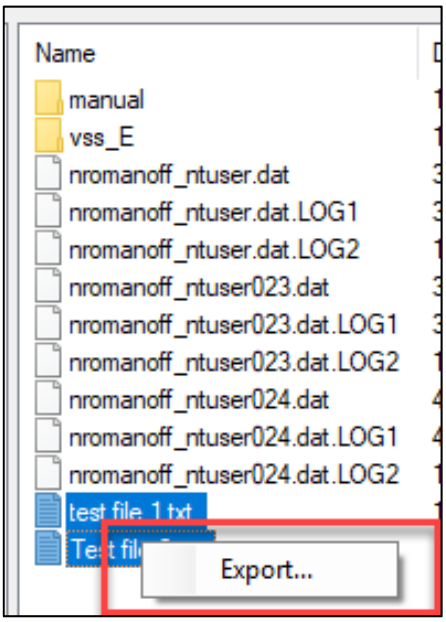


37. Expand the **C:** drive in the tree on the left, then select the **Temp** folder. Notice that the two text files we created, then deleted, are shown.

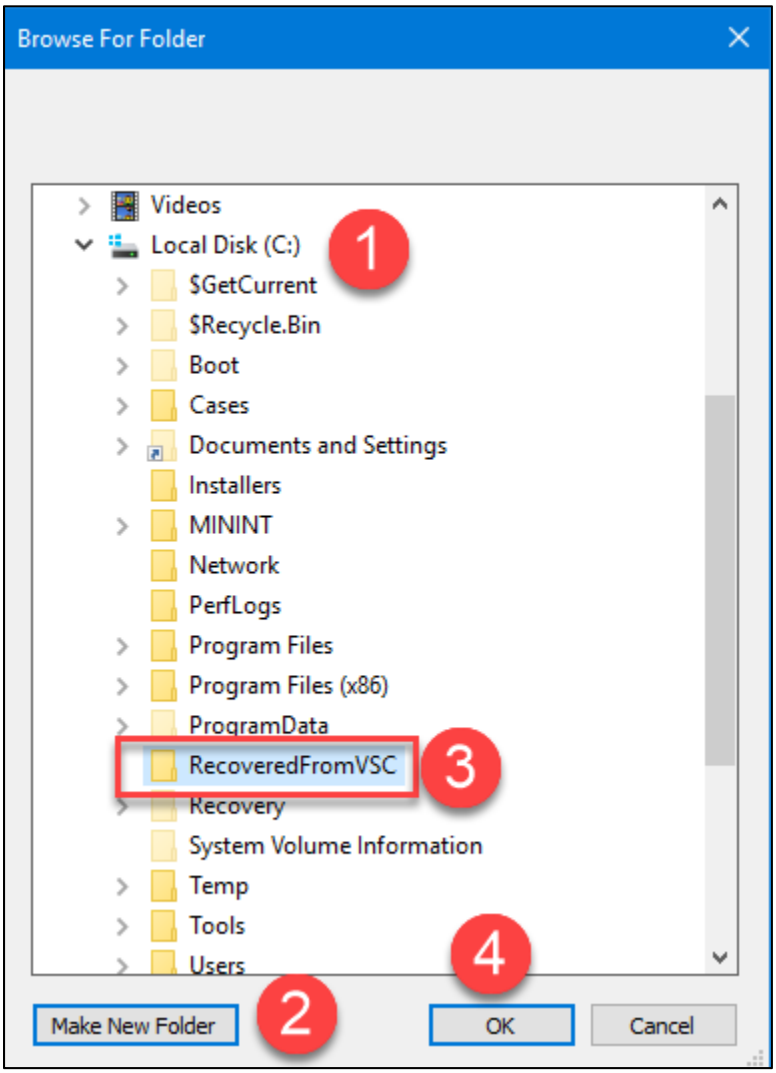


Recall from an earlier step that we deleted these two text files from **C:\temp**.

38. To recover the text files from a **VSC**, select both files, then **right click** to bring up a context menu. Select **Export…** from the context menu.



39. In the **Browse For Folder** dialog, select the **C:** drive, then click **Make New Folder**. Name the new folder **`RecoveredFromVSC`**, then click **OK**.

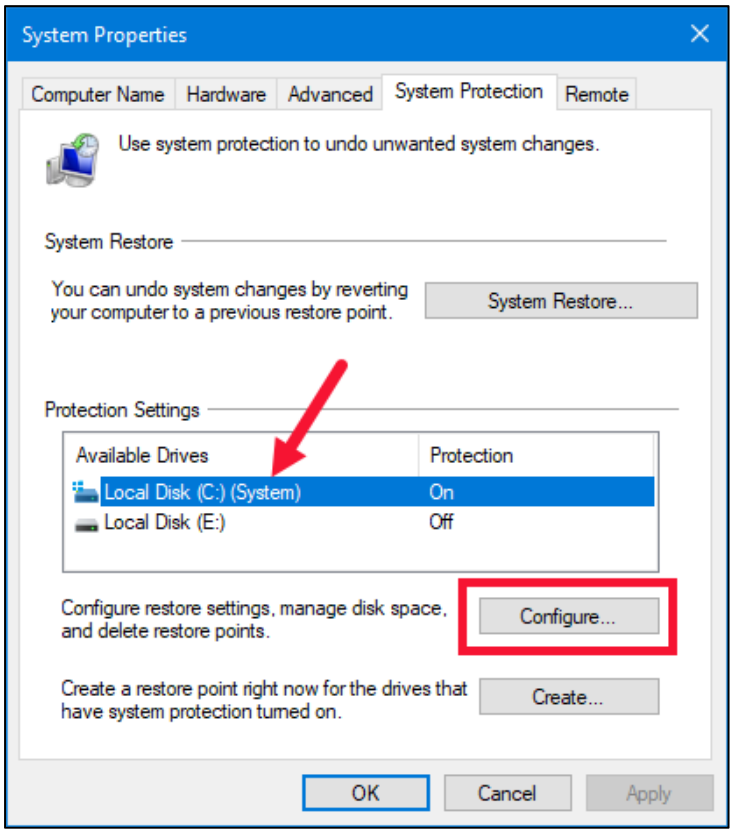


40. Using **File Explorer**, navigate to **C:\RecoveredFromVSC** and open each text file to confirm their contents. You can also **right-click** each file and choose **Properties** to verify the metadata associated with the files remains intact (created, modified, last access timestamps).

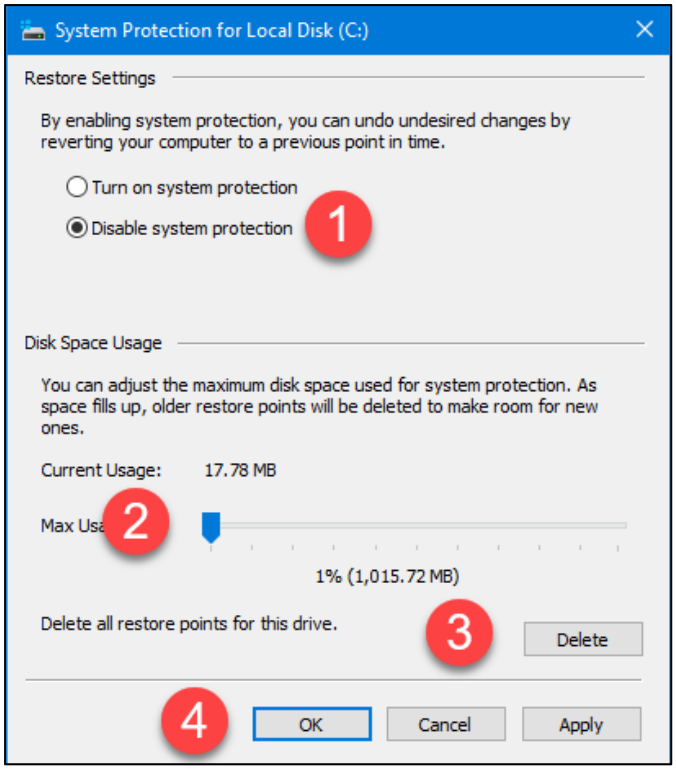
41. Using this same technique, select several other **VSCs** from your system and recover things such as Registry hives to the **C:\RecoveredFromVSC** folder.

42. Using **Shadow Explorer**, click on **File → Configure System Protection**.

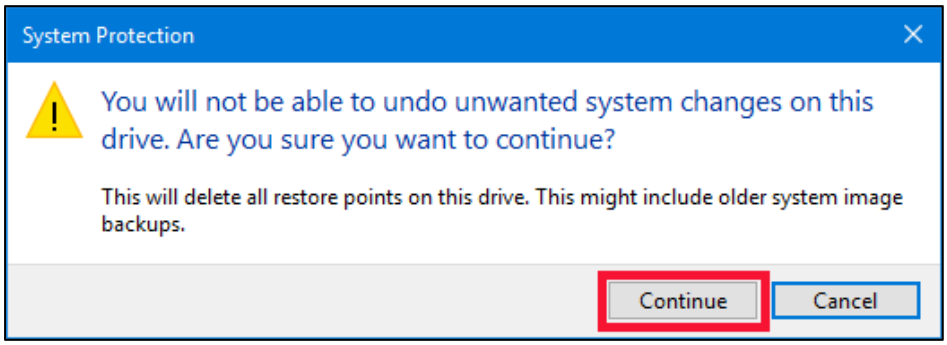
43. Select the **C:** drive from the list of drives, then click **Configure**.

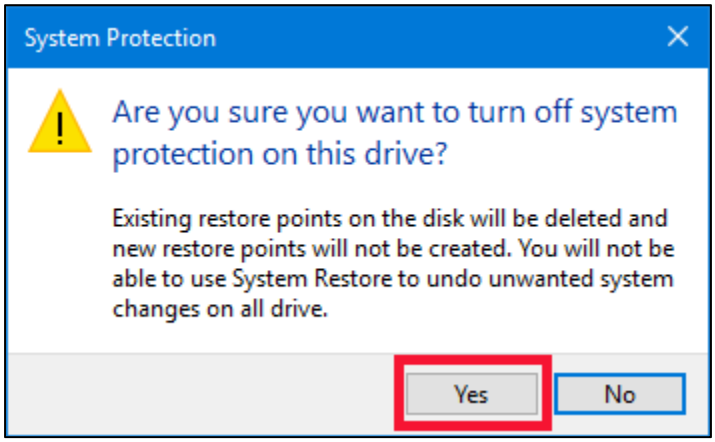


44. Select **Disable system protection**, slide **Max usage** all the way to the left, then click **Delete** to delete all restore points. Accept any confirmation dialogs, then click **OK** to close.



45. Click **Continue**, and then click **Yes** when asked to confirm turning off **System Protection**.





46. Close any remaining dialog boxes in **System Protection**.

47. **Volume Shadow Copies** are now disabled on the **VM**.

Exercise – Questions

1. Using **File Explorer**, navigate to **C:\RecoveredFromVSC**, then open each of the text files you restored using **Shadow Explorer**.

    a. Is the content changed in either of the two files? (Hint: Open both files and compare the contents to what was in each earlier when you created them)

    _____

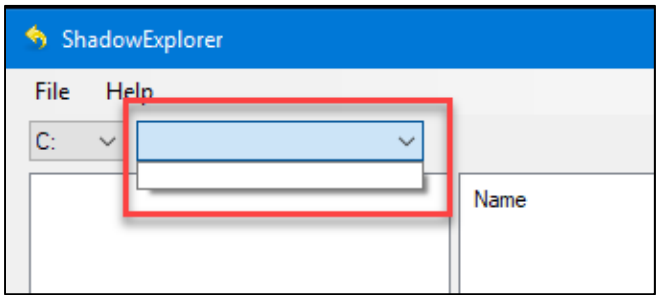
    b. Do either of these two files exist in **C:\Temp**?

    _____

2. If **Shadow Explorer** is running, close it.

3. Start **Shadow Explorer** via the shortcut in the **Utilities** fence on the **Desktop**.

4. Make sure the **C:** drive is selected in **Shadow Explorer**, then click the drop down to view all available **VSCs** on the **C:** drive.



   a. How many **VSCs** are available on the **C:** drive?

   _____

   b. Why are there no shadow copies available?

   _____

5. Start **Hasher** via the shortcut in the **Utilities** fence on the **Desktop**.

6. Using **File Explorer**, select the Registry hives and LOG files you recovered from **VSCs** and drag them into the **Hasher** window. You can also use **File → Select File** menu in **Hasher** to open these files.

7. Look at the hashes generated by **Hasher** and compare both the Registry hives (files ending in .dat) as well as the LOG files (files ending in .LOG1 or .LOG2) and answer the following questions.

   a. Do any of the hives have the same hash? If so, which ones?

   _____

   b. Do any of the LOG files have the same hash? If so, what are the filenames of the LOG files with matching hashes?

   _____

   _____

   _____

c. For the previous question, why do the hashes match? What does this mean?

_____

_____

_____

8. Using **File Explorer**, delete the **C:\Temp\vss_E** directory (adjust to match the drive letter where the **VSCs** were mounted if necessary), then delete the **C:\RecoveredFromVSC** directory.

Exercise – Questions Step-by-Step

1. Using **File Explorer**, navigate to **C:\RecovererdFromVSC**, then open each of the text files you restored using **Shadow Explorer**.

   a. Is the content changed in either of the two files? (Hint: Open both files and compare the contents to what was in each earlier when you created them)

      No.

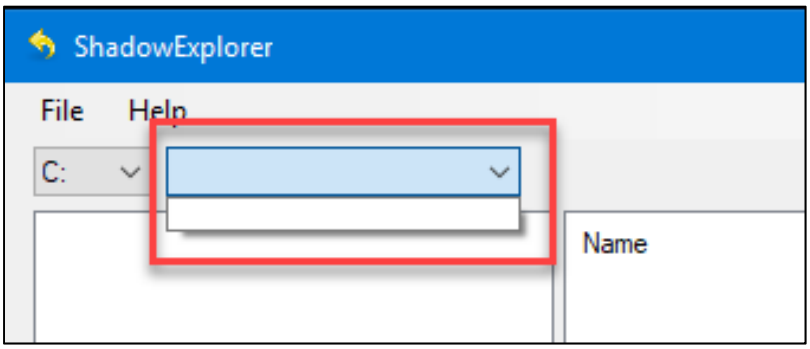
   b. Do either of these two files exist in **C:\Temp**?

      No, they do not. They were deleted and the Recycle Bin was emptied.

2. If **Shadow Explorer** is running, close it.

3. Start **Shadow Explorer** via the shortcut in the **Utilities** fence on the **Desktop**.

4. Make sure the **C:** drive is selected in **Shadow Explorer**, then click the drop down to view all available **VSCs** on the **C:** drive.



   a. How many **VSCs** are available on the **C:** drive?

      None.

   b. Why are there no shadow copies available?

      We disabled System protection on the C drive earlier and deleted any existing VSCs for the volume.

5. Start **Hasher** via the shortcut in the **Utilities** fence on the **Desktop**.

6. Using **File Explorer**, select the Registry hives and LOG files you recovered from **VSCs** and drag them into the **Hasher** window. You can also use **File → Select File** menu in **Hasher** to open these files.

7. Look at the hashes generated by **Hasher** and compare both the Registry hives (files ending in .dat) as well as the LOG files (files ending in .LOG1 or .LOG2) and answer the following questions.

   a. Do any of the hives have the same hash? If so, which ones?

      No, all, of the files ending in .dat have unique hashes.

   b. Do any of the LOG files have the same hash? If so, what are the filenames of the LOG files with matching hashes?

      Yes, three LOG files have the same hash:

      C:\Temp\nromanoff_ntuser.dat.LOG2
      C:\Temp\nromanoff_ntuser003.dat.LOG2
      C:\Temp\nromanoff_ntuser004.dat.LOG2

   c. For the previous question, why do the hashes match? What does this mean?

      The fact that they are all the same means the contents of the files have not changed as copies of the LOG2 file were preserved in various VSCs (as well as the active file on disk).

      Windows can use either LOG1 or LOG2 (or a combination of both) in regard to Registry transaction logs, and it seems, in this case, Windows was using LOG1 as its primary log. This is further confirmed by the fact that each hash for the LOG1 files that were recovered all have different hashes.

8. Using **File Explorer**, delete the **C:\Temp\vss_E** directory (adjust to match the drive letter where the **VSCs** were mounted if necessary), then delete the **C:\RecoveredFromVSC** directory.

Exercise—Key Takeaways

- Volume Shadow Copies are a fantastic resource of historical forensic data.
- There are many ways to access VSCs, ranging from command line to graphical user interfaces.
- Due to how much data is available in VSCs, the best approach is a targeted triage collection against VSCs instead of imaging a VSC in its entirety.

Exercise 4.2—Using KAPE for Battlefield Forensics

Background

Storage devices continue to grow and get cheaper. This presents problems for forensicators because as the amount of storage increases, so does the amount of time it takes to create a full disk image. It typically takes hours to image an average size device, and these hours are lost when it comes to generating leads and findings answers.

By changing our approach and performing battlefield forensics prior to, or even in place of, a full disk image, we can find answers, generate leads, and solve cases much faster.

KAPE is a tool capable of rapid collection and processing of key artifacts. It can target a live system or mounted drive letter to rapidly extract key files in a forensically sound manner. It can also process the collected files using various parsers to extract out the key data from the collected data. This data can then be analyzed, and the information used for a wide range of purposes. This process typically takes minutes to complete and can target a drive and any Volume Shadow Copies, all in a single operation.

Objectives

- Understand target files and how to make new ones
- Use KAPE's target capability to collect key artifacts
- Understand module files and how to make new ones
- Use KAPE's module capability to process collected artifacts
- Put it all together by using KAPE to collect from a drive letter and Volume Shadow Copies in a single operation, then process the results
- Analyze the data produced by KAPE using Timeline Explorer

Exercise Preparation: Creating KAPE Targets and Modules

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Verify **WindowsImage.E01** is mounted (refer to the **Mounting Evidence** exercise (**exercise 3.2**) for mounting instructions).

NOTE: KAPE comes with dozens of targets that cover the most common files that are of interest to investigators. It is also very easy to create new target files. In this exercise, we will create a new target file that looks for and collects files matching the pattern ***.zip**.

4. Ensure the script to update **KAPE** to the latest version has been run. To do this, open a new **PowerShell** window using the shortcut on the **Desktop** and execute the following commands (press **Enter** after each one to execute the command):

```
cd c:\Tools\KAPE
Get-KAPEUpdate.ps1
```

NOTE: This step requires Internet access, so skip it if you cannot connect to the Internet for some reason.

5. When the update is finished, run the following command to make sure the most current targets and modules are available:

```
.\kape.exe --sync
```

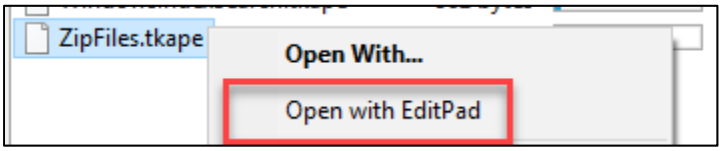
6. Open **File Explorer** and navigate to **C:\Tools\KAPE\Targets\Windows**

7. Locate the existing target file named **SRUM.tkape**. We will use this as our template for our new configuration.

8. Make a copy of **SRUM.tkape** by selecting the file, pressing **CTRL-C**, then **CTRL-V**. This will create a new file named **SRUM - Copy.tkape**.

9. Rename **SRUM - Copy.tkape** to **`ZipFiles.tkape`**.

10. Open **ZipFiles.tkape** by right clicking on it and choosing **Open with EditPad**



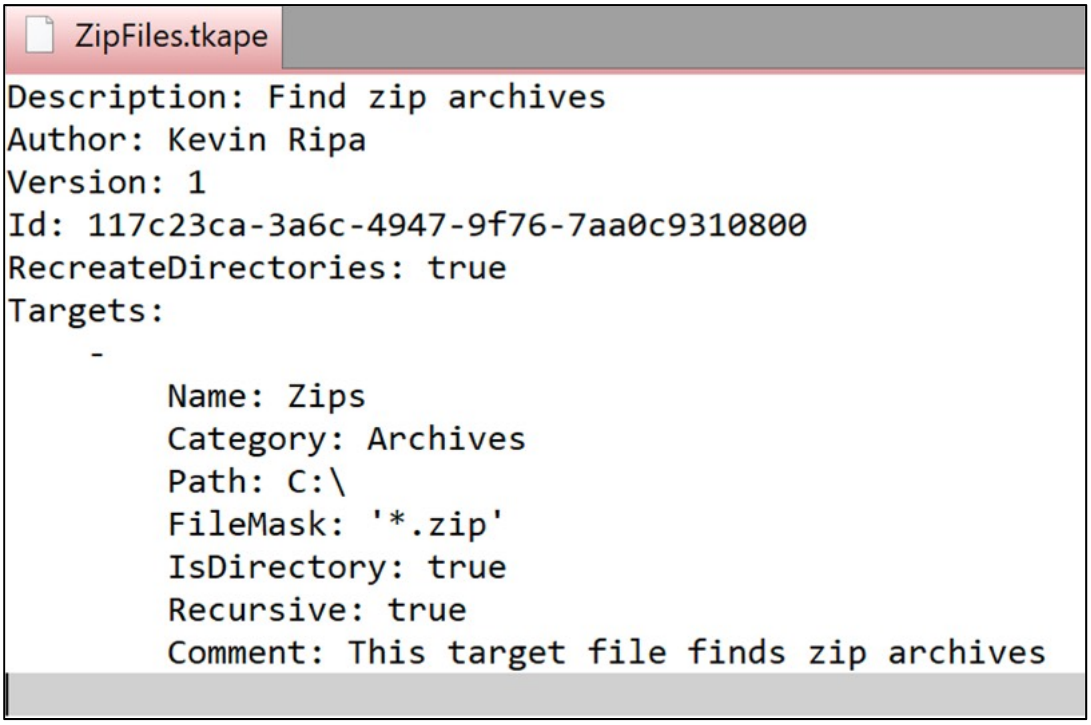
11. Edit the file to reflect the following:

    Note: When updating the GUID below, changing a few characters around is sufficient, but other ways are shown below.

    a. **UPDATE** Description: Find zip archives
    b. **UPDATE** Author: <Your name here>
    c. **UPDATE** Id: A random GUID. **KAPE** can generate a random GUID for you via the **`KAPE.exe --guids`** switch. You can also generate one at **https://www.guidgenerator.com/**
    d. Under the Targets section:
        i. **UPDATE** Name: Zips
        ii. **UPDATE** Category: Archives
        iii. **UPDATE** Path: C:\
        iv. ***ADD*** FileMask: '*.zip'
        v. **UPDATE** Comment: This target file finds zip archives

    NOTE: YAML (YAML Ain't Markup Language) does not allow **TAB** characters to be used, so be sure to use spaces when adding **FileMask**.

12. When you are done, your file should look like this (except for the GUID, as yours will be different).



13. Be sure to save the file you just created, and then close it.

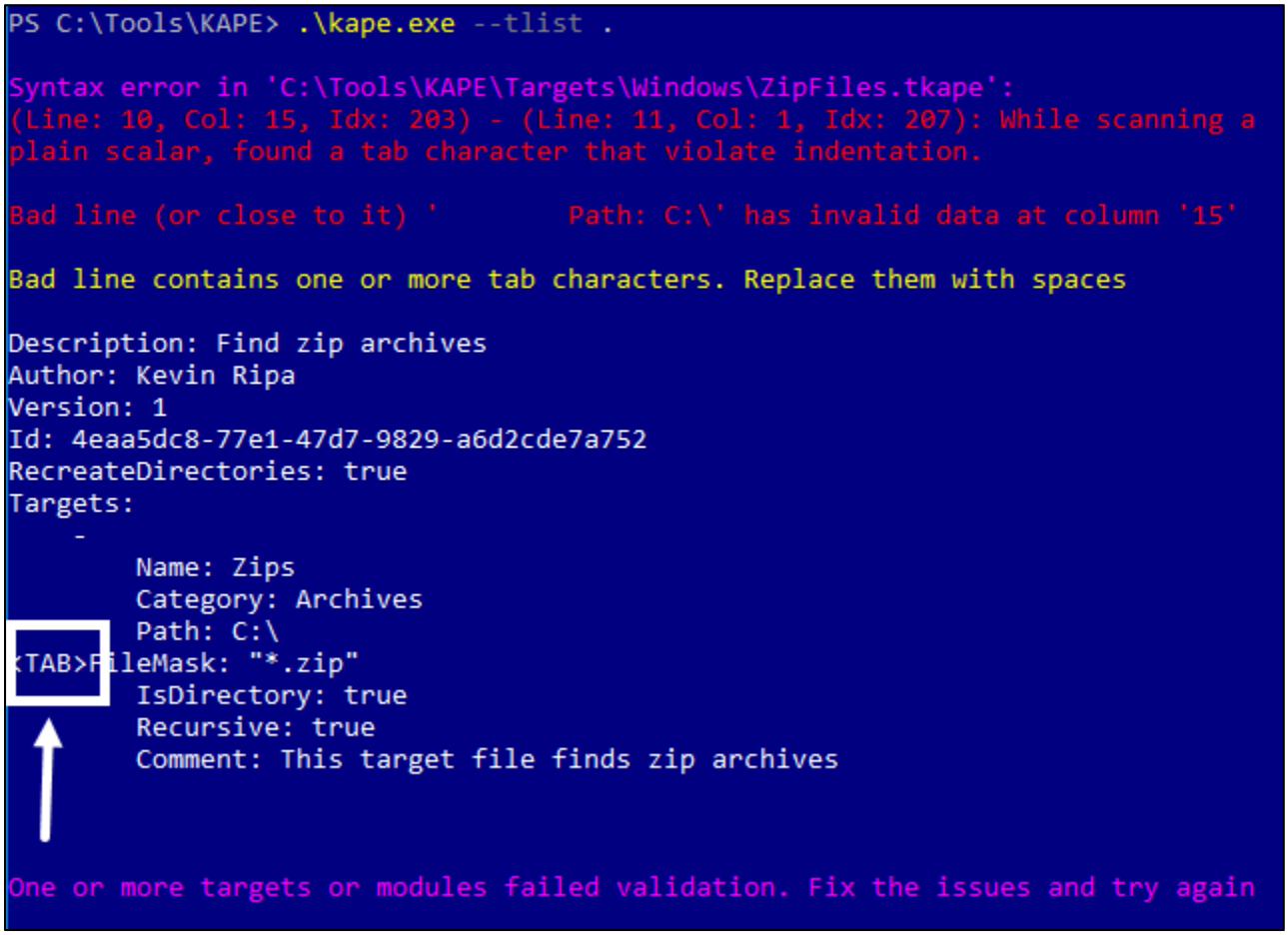
14. Since target files are defined using **YAML**, it is good practice to validate your file. **KAPE** will do this for you when the target is used, but the **`--tlist`** and **`--tdetail`** switches will validate things for you as well.

    Open a new **PowerShell** window using the shortcut on your **Desktop** and run the following command to validate your newly saved target file. Be sure to include the **<SPACE>period** at the end of the command as this tells **KAPE** to look in the current directory for targets.

```
cd c:\Tools\KAPE
.\kape.exe --tlist .
```

```
PS C:\Tools\KAPE> .\kape.exe --tlist .

Targets found: 2

Target: !BasicCollection
        Description: Basic Collection

Target: !SANS_Triage
        Description: SANS Triage Collection.


Sub directories
!Disabled
!Local
Antivirus
Apps
Browsers
Logs
Misc
P2P
Windows
```

KAPE will also perform validation when it runs. If a target is not well defined, **KAPE** will inform you and exit, like this:



Here **KAPE** is telling us about the error we mentioned related to tab characters but notice that it tells you where the issue is and shows the **<TAB>** in the config file.

If you get any errors, correct them, resave the file, and repeat the above command.

15. The target configuration is now ready to be used in **KAPE**.

16. Next, let's create a module to extract some details about the contents of the files contained in the **zip** files our target finds. Open **File Explorer** and navigate to **C:\Tools\KAPE\Modules\ProgramExecution**

17. Locate the existing module named **AmcacheParser.mkape**. We will use this as our template for our new module.

18. Make a copy of **AmcacheParser.mkape** by selecting the file, pressing **CTRL-C**, then **CTRL-V**. This will create a new file named **AmcacheParser - Copy.mkape**.

19. Rename **AmcacheParser - Copy.mkape** to **`ZipFileContents.mkape`**.

20. Open **ZipFileContents.mkape** by right clicking on it and choosing **Open with EditPad**.

21. Edit the file to reflect the following:

    NOTE: When updating the GUID below, changing a few characters around is sufficient, but other ways are shown below.

    NOTE: In the update below, the **CommandLine** is a lower-case L (ell), not a one or a pipe. Additionally, notice a few lines are enclosed by single quotes. This is to "turn off" special meaning in **YAML** for values starting with a *****, or values that contain other special characters, like a colon. When in doubt, you can always enclose strings in single quotes.

    a. **UPDATE** Description: '7z.exe: List contents of zip archives'
    b. **UPDATE** Category: Archives
    c. **UPDATE** Author: <Your name here>
    d. **UPDATE** Id: A random GUID. **KAPE** can generate a random GUID for you via the **`--guids`** switch. You can also generate one at **https://www.guidgenerator.com/**
    e. **UPDATE** BinaryUrl**:** https://www.7-zip.org/download.html
    f. **UPDATE** ExportFormat: txt
    g. **UPDATE** FileMask: '*.zip'
    h. Under the Processors section:
        i. **UPDATE** Executable: 7z.exe
        ii. **UPDATE** CommandLine: l %SourceFile%
        iii. **UPDATE** ExportFormat: txt
        iv. **ADD** ExportFile: 7zFileContents.txt

    While the **FileMask** can be anything that **7z.exe** supports, we are only interested in **zip** files (***.zip**) and have set the **FileMask** to look for these types of files.

    Again, be sure to not use **TAB** characters.

22. When you are done, your file should look like this (except for the GUID as yours will be different).

ZipFileContents.mkape

```
Description: '7z.exe: List contents of zip archives'
Category: Archives
Author: Kevin Ripa
Version: 1
Id: b05fbf5b-0adf-465c-89d0-0ab4b63148cb
BinaryUrl: https://www.7-zip.org/download.html
ExportFormat: txt
FileMask: '*.zip'
Processors:
    -
        Executable: 7z.exe
        CommandLine: l %SourceFile%
        ExportFormat: txt
        ExportFile: 7zFileContents.txt
```

23. Be sure to save the file you just created, and then close it.

Like we saw with targets, **KAPE** can check for syntax errors in modules too. Run the following command to verify your newly saved module file. Be sure to include the **<SPACE>period** at the end of the command, as this tells **KAPE** to look in the current directory for modules.

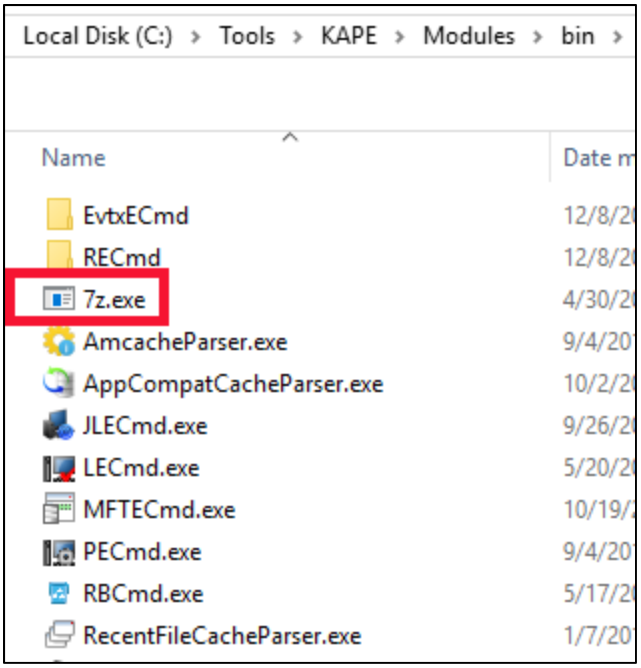
```
cd C:\Tools\KAPE
.\kape.exe --mlist .
```

If you get any errors, correct them, resave the file, and repeat the above command.

24. Since this is a new module, **7z.exe** does not currently exist where it needs to for **KAPE** to use it. We need to make a copy of **7z.exe** and place it in the **C:\Tools\KAPE\Modules\bin** directory. In **PowerShell**, type the following and press **Enter**.

```
cp C:\Tools\7z.exe C:\Tools\KAPE\Modules\bin\
```

NOTE: There is also a **C:\Tools\7z** directory, but we do *not* want the executable in that folder. Be sure to copy the **7z.exe** file from **C:\Tools**.



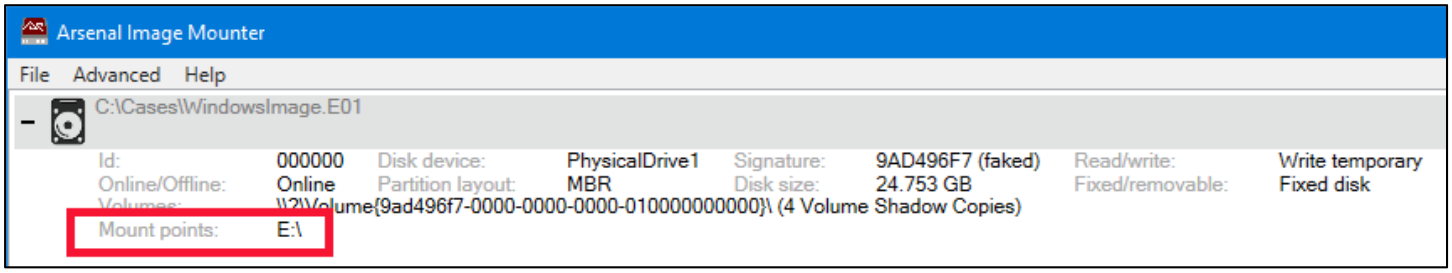
25. **KAPE** will look for executables in modules in the **bin** directory. If we did not copy the executable so **KAPE** could find it and we tried to run our module, we would end up with something like this:

```
PS C:\Tools\KAPE> .\kape.exe --msource C:\Temp\out --mdest C:\Temp\mout --module ZipFileContents --mflush
KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)

KAPE directory: C:\Tools\KAPE
Command line: --msource C:\Temp\out --mdest C:\Temp\mout --module ZipFileContents --mflush

Using Module operations
        Flushing module destination directory 'C:\Temp\mout'
        Creating module destination directory 'C:\Temp\mout'
                Found processor 'Executable: 7z.exe, Cmd line: l %SourceFile%, Export: txt, Append: False'!

Discovered 1 processor to run.
Executing modules with file masks
Cannot find executable '7z.exe' in directory 'C:\Tools\KAPE\Modules\ZipFileContents' or 'C:\Tools\KAPE\Modu
les\bin'. Aborting execution and skipping any further modules using this executable
Executing remaining modules...
Executed 1 processor in 0.0194 seconds

Total execution time: 0.0559 seconds
```

26. The module configuration is now ready to be used in **KAPE**.

Exercise Questions: Collection and Analysis with KAPE

1. Let's take advantage of the target file we just made to find and extract **zip** archives from a drive. Verify the **WindowsImage.E01** file is mounted in **Arsenal Image Mounter (AIM)** and verify the drive letter in the **AIM** interface. In the image below, the E01 has been mounted to **E:\.**



2. To use our new target, open a **PowerShell** window with Administrator privilege and navigate to **C:\Tools\KAPE**. Once there, run .**kape.exe** without any arguments. It will look something like this (partial output shown):

```
PS C:\Tools\KAPE> .\kape.exe

KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)

        tsource         Target source drive to copy files from (C, D:, or F:\ f
        target          Target configuration to use
        tdest           Destination directory to copy files to. If --vhdx, --vh
s set, files will end up in VHD(X) container or zip file
        tlist           List available targets. Use . for Targets directory or
irectory under Targets.
        tdetail         Dump target file details
        tflush          Delete all files in 'tdest' prior to collection
```

3. Since we just made a target file, we will use the target related options. In **PowerShell**, enter the following command, then press **Enter.**

   NOTE: Be sure to update the drive letter used with **--tsource** to what **AIM** shows!

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --
tflush
```

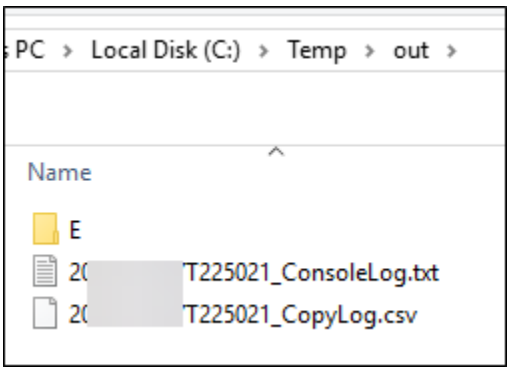
4. **KAPE** will now run the **ZipFiles** target against the mounted drive, and copy any files found, to the **C:\Temp\out** directory.

```
PS C:\Tools\KAPE> .\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --tflush
KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)

KAPE directory: C:\Tools\KAPE
Command line: --tdest C:\Temp\out --tsource E: --target ZipFiles --tflush

Using Target operations
        Flushing target destination directory 'C:\Temp\out'
        Creating target destination directory 'C:\Temp\out'
Found 1 targets. Expanding targets to file list...
Found 4 files in 13.390 seconds. Beginning copy...

Copied 4 out of 4 files in 14.5556 seconds. See '*_CopyLog.csv' in 'C:\Temp\out' for copy
details

Total execution time: 14.5853 seconds
```

   a. How many total files did **KAPE** find?

      _____

5. Open **File Explorer** and navigate to **C:\Temp\out**.



Notice there is a directory and several files. The directory contains all the files (and their full directory paths) that were located under the **E:** drive. The **CopyLog.csv** file contains details of files copied including timestamps, SHA-1, source and destination path, and so on which makes it easier to filter, search, etc. Finally, the **ConsoleLog.txt** file contains a copy of everything that was displayed on the screen while **KAPE** ran. This is useful for reviewing any messages related to errors, etc., should any occur.

6. Open the **<*date/time*>_CopyLog.csv** and answer the following questions:

   a. What are the first 5 characters of the **SHA-1** value for **Undercover Agent-List-Classified\Agents-List-CLASSIFIED-TOP-SECRET.zip**?

   _____

   b. What is the **CreatedOn** date for file **$ISEBKEC.zip**?

   _____

7. Open the **<*date/time*>_ConsoleLog.txt** file and review its contents, then close both files that are open.

8. Rerun the previous command by pressing the Up arrow. Add the **`--debug`** switch to the end of the command, then press **Enter.**

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --
tflush --debug
```

9. When **KAPE** is finished running, open the **<*date/time*>_ConsoleLog.txt** file and review its contents.

    a. What kinds of information is available in the **<*date/time*>_ConsoleLog.txt** file now that was not in the previous one?

    _____

    _____

    _____

    _____

    b. Why would you want to use the **`--debug`** option?

    _____

    _____

10. Repeat the last command, but this time, add the **--trace** option, then press **Enter.**

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --
tflush --debug --trace
```

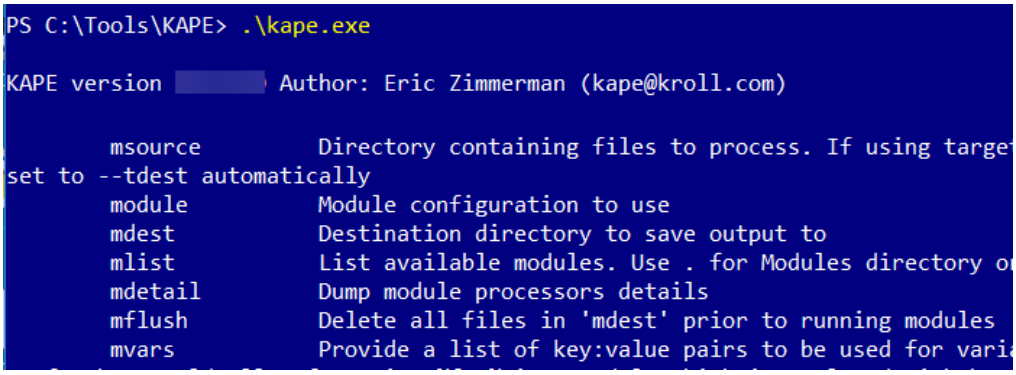
11. Review the **<*date/time*>_ConsoleLog.txt** file again (or simply scroll up in the **PowerShell** window). Trace adds a significant amount of information to the output when compared to the **--debug** option.

12. Finally, spend a few moments exploring the contents of **C:\Temp\out\E** and its various subdirectories to see how **KAPE** found and saved the target files. Notice that all the timestamps have been preserved for each directory and file that was recovered.

    The debug and trace options are useful for troubleshooting or viewing file progress for slower connections.

13. Next, let's process the files we just collected with the module file we created earlier. Run **kape.exe** without any options and look for the switches related to modules.

```
.\kape.exe
```



14. In **PowerShell**, enter the following command, then press **Enter.**

```
.\kape.exe --msource C:\Temp\out --mdest C:\Temp\mout --module
ZipFileContents --mflush
```

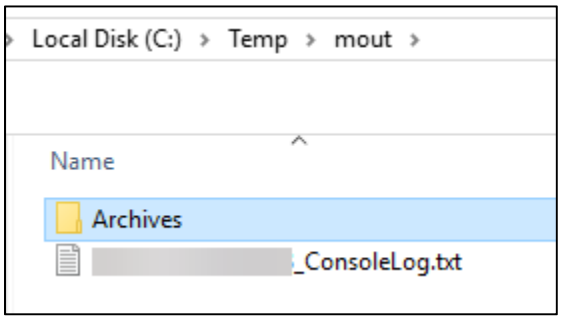
15. **KAPE** will now run the **ZipFileContents** module against the files in **C:\Temp\out** (which is where we told **KAPE** to save files collected via the target from earlier in the exercise) and save the results to **C:\Temp\mout**.

```
PS C:\Tools\KAPE> .\kape.exe --msource C:\Temp\out --mdest C:\Temp\mout --module ZipFileContents --mflush
KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)

KAPE directory: C:\Tools\KAPE
Command line: --msource C:\Temp\out --mdest C:\Temp\mout --module ZipFileContents --mflush

Using Module operations
        Flushing module destination directory 'C:\Temp\mout'
        Creating module destination directory 'C:\Temp\mout'
                Found processor 'Executable: 7z.exe, Cmd line: l %SourceFile%, Export: txt, Append: False'!
Discovered 1 processor to run.
Executing modules with file masks...
        Running '7z.exe': l "C:\Temp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$ISEBKEC.zip"
                ERROR: C:\Temp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$ISEBKEC.zip : C:\Te
mp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$ISEBKEC.zip
                Open ERROR: Can not open the file as [zip] archive
                ERRORS:
                Is not archive
        Running '7z.exe': l "C:\Temp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$RSEBKEC.zip"
                Output file updated to 'C:\Temp\mout\Archives\7zFileContents_1.txt'
        Running '7z.exe': l "C:\Temp\out\E\Users\nromanoff\Documents\Undercover Agent-List-Classified\Agents-List-CLA
SSIFIED-TOP-SECRET.zip"
                Output file updated to 'C:\Temp\mout\Archives\7zFileContents_2.txt'
        Running '7z.exe': l "C:\Temp\out\E\Users\nromanoff\AppData\LocalLow\Adobe\Acrobat\10.0\rdrmessage.zip"
                Output file updated to 'C:\Temp\mout\Archives\7zFileContents_3.txt'
Executing remaining modules...
Executed 1 processor in 0.2502 seconds

Total execution time: 0.2937 seconds
```

a. How many total files did **KAPE** process?

_____

16. Open **File Explorer** and navigate to **C:\Temp\mout**.



Again, we see a directory and a ***<date/time>*_ConsoleLog.txt** file. The **ConsoleLog** file serves the same purpose as we saw earlier. Recall from the module configuration that we specified a **Category** of "**Archives**" and here we see a directory with the same name. This allows you to tie specific processors to a common group so related files end up in the same directory.

17. Open the **Archives** directory in **File Explorer**.

18. Open the text files in **EditPad Lite**.

a. What do the files contain?

_____

_____

_____

b. Locate the **7zFileContents** text file that relates to the **Agents** list. What is the name of the archive file?

_____

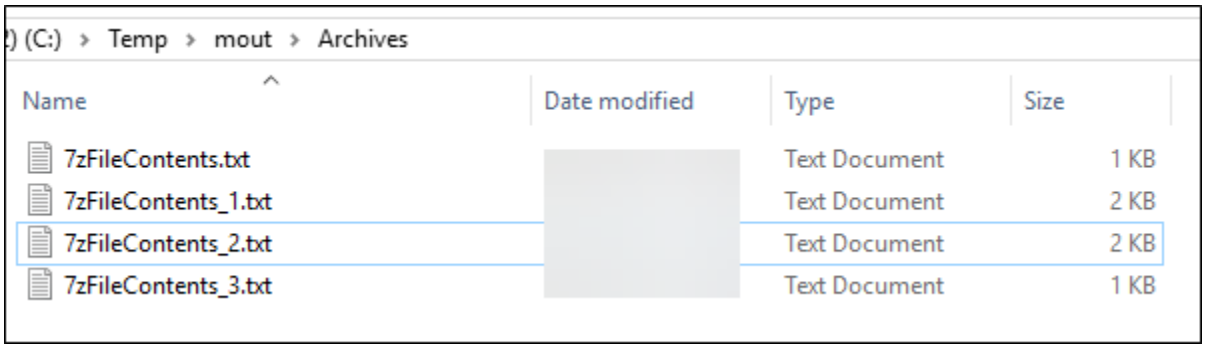
c. How many files does this **zip** archive contain?

_____

d. What is the total size for the files (uncompressed)?

_____

19. Recall from earlier that **KAPE** found four **zip** files. Here we see the same number of text files. **KAPE** is smart enough to not overwrite files as each target is found, and when this happens, **KAPE** appends a unique counter to the end of the file name, resulting in the file names shown below.



20. Spend a few minutes exploring the remaining **ZipFileContents** text files.

    a. Did any of them run into errors? If so, what was the name of the **zip** file?

    _____

    b. Locate the **zip** file from above and open it by double clicking it. Does it open? (**Hint**: You will have to go to the target's output folder, which is shown in the text file where you got the information from the previous question)

    _____

    BONUS: Why does the **zip** file not open? Can you find the contents of the **zip** file?

    _____

    _____

    _____

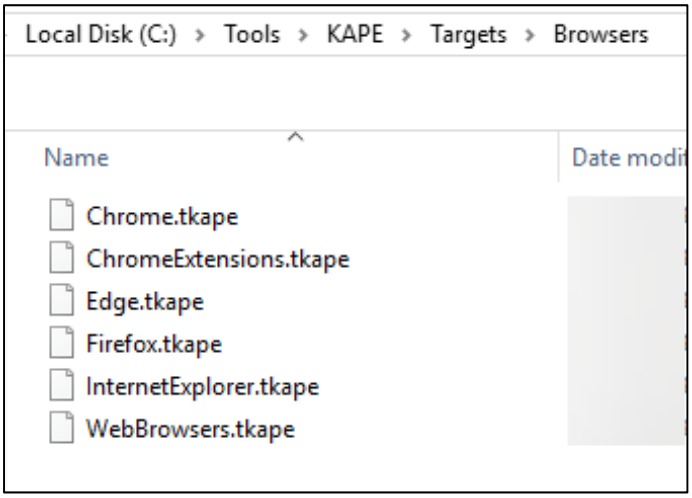
21. In the previous sections, we saw how **KAPE** can be used to both collect files via targets, and process files via modules. In this section, we will combine the two operations into a single step.

22. For both targets and modules, **KAPE** has command line switches that list all available targets and modules, **`--tlist`** and **`--mlist`**.

23. Open a **PowerShell** window with administrator rights and navigate to **`C:\Tools\KAPE`**

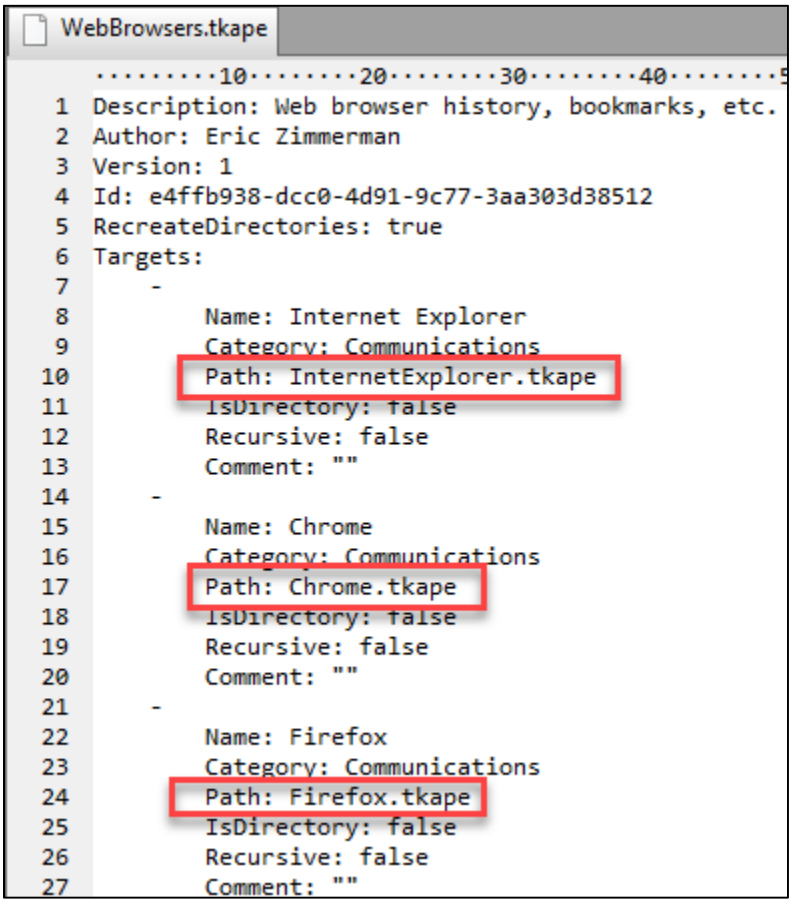
24. While we could run any of the targets or modules independently and execute **KAPE** multiple times (one per target or module), **KAPE** can define targets and modules that refer to other targets and modules.

25. To create a compound target file, open **File Explorer** and navigate to **C:\Tools\KAPE\Targets\Browsers** to see a list of available targets.



26. Right click on **WebBrowsers.tkape** and open it with **EditPad**.

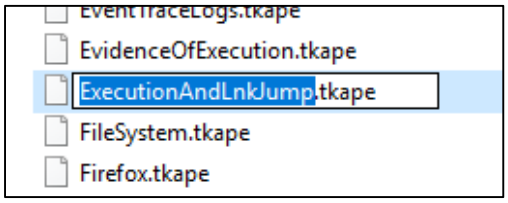
Notice how the **Path** property references another target configuration. Because there are many different browsers, target files were created that specifically look for a single browser, but in most cases, we are interested in knowing about all web browsers, so a compound target was made that pulls in four web browsers (although the image below is only showing the first three).



27. For our purposes, we want to build a compound target that will collect **evidence of execution** artifacts as well as **lnk** files and **jumplists**. Since targets already exist for those things, we will create a new compound target and reference the existing target configurations.

28. Close any currently opened files in **EditPad**. In **File Explorer**, select **WebBrowsers.tkape** by clicking it once, then press **CTRL-C**, **CTRL-V** to make a copy of the file.

29. Select **WebBrowsers - Copy.tkape** by clicking it once, then press **F2** to rename the file. Rename the file **`ExecutionAndLnkJump.tkape`**



30. Right click on **ExecutionAndLnkJump.tkape** and open it with **EditPad**. Edit the file to reflect the following:

    a. **UPDATE** Description: Collect evidence of execution, lnk files, and jumplists
    b. **UPDATE** Author: <Your name here>
    c. **UPDATE** Id: A random GUID. Use **KAPE** to generate a GUID.
    d. Under the **FIRST** Targets section:
        i. **UPDATE** Name: Evidence Of Execution
        ii. **UPDATE** Category: Execution
        iii. **UPDATE** Path: EvidenceOfExecution.tkape
    e. Under the **SECOND** Targets section:
        i. **UPDATE** Name: Lnk and JumpLists
        ii. **UPDATE** Category: FileFolderOpening
        iii. **UPDATE** Path: LnkFilesAndJumpLists.tkape
    f. **DELETE** the remaining targets (Firefox, Edge, etc.).

31. When you are done, your file should look like this (except for the GUID as yours will be different).

ExecutionAndLnkJump.tkape

```
Description: Collect evidence of execution, lnk files, and jumplists
Author: Kevin Ripa
Version: 1
Id: 9cf813a0-281c-4208-89c6-c93d3695e4df
RecreateDirectories: true
Targets:
    -
        Name: Evidence of Execution
        Category: Execution
        Path: EvidenceOfExecution.tkape
        IsDirectory: false
        Recursive: false
        Comment: ""
    -
        Name: Lnk and JumpLists
        Category: FileFolderOpening
        Path: LnkFilesAndJumpLists.tkape
        IsDirectory: false
        Recursive: false
        Comment: ""
```

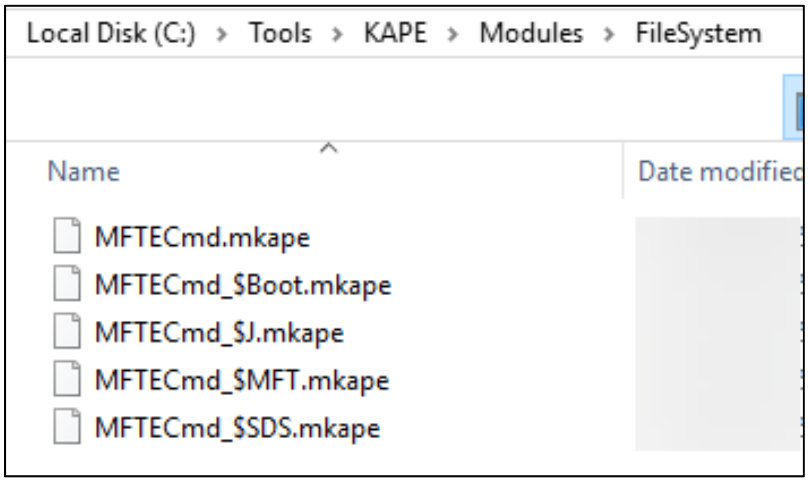
32. Save the file and validate it.

```
.\kape.exe --tlist .
```

Correct any errors, resave, and revalidate with the above command.

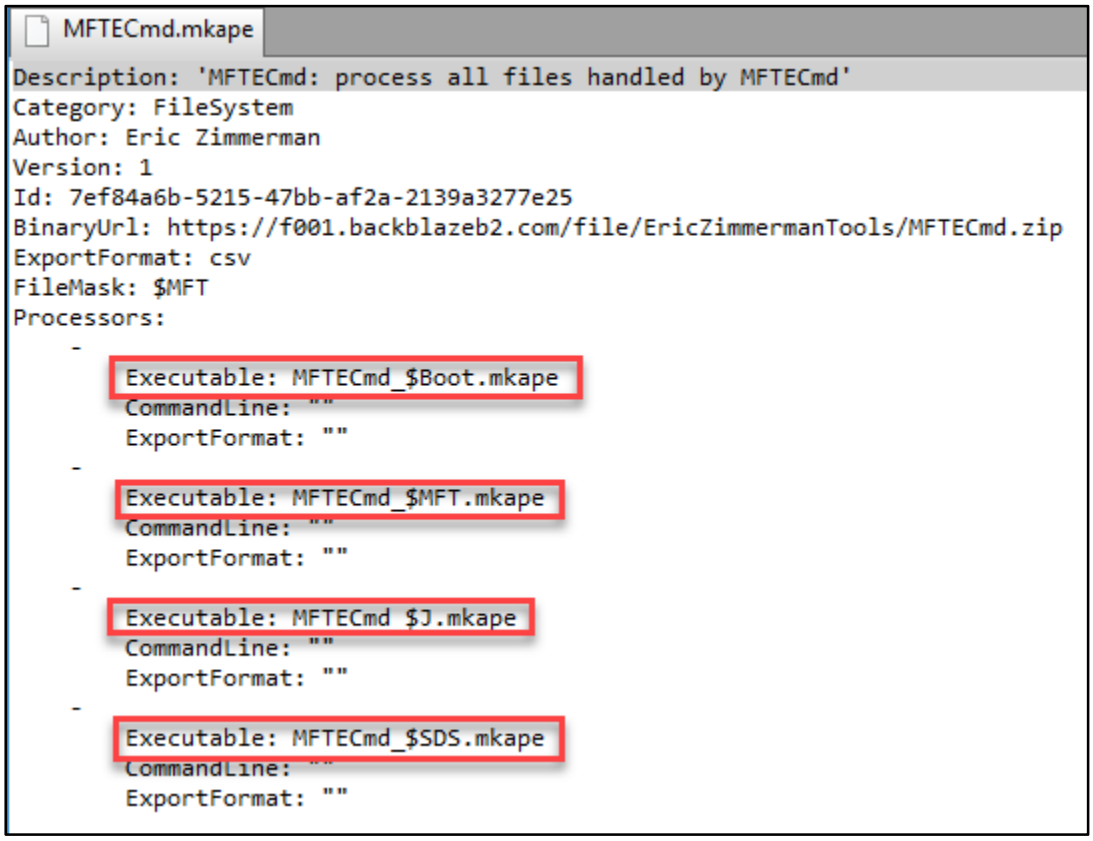
33. The compound target is now finished and can be used to collect all the file types as defined in the two target files referenced. Before we collect files using this target, let's follow the same process, but this time with a module file.

34. Using **File Explorer**, navigate to **C:\Tools\KAPE\Modules\FileSystem**



35. Right click on **MFTECmd.mkape**, then open the file with **EditPad**.

    Notice how the **Executable** property references another module file. Just as we saw with browsers, there are many **evidence of execution** related artifacts, so we will create a module that targets these, as well as **lnk** files and **jumplists**.



36. Close any currently opened files in **EditPad**. In **File Explorer**, select **MFTECmd.mkape** by clicking it once, then press **CTRL-C**, **CTRL-V** to make a copy of the file.

37. Select **MFTECmd - Copy.mkape** by clicking it once, then press **F2** to rename the file. Rename the file **`ExecutionAndLnkJump.mkape`**

38. Right click on **ExecutionAndLnkJump.mkape** and open it with **EditPad**. Edit the file to reflect the following:
    a. **UPDATE** Description: Collect evidence of execution, lnk files, and jump lists
    b. **UPDATE** Category: EvidenceOfExecution
    c. **UPDATE** Author: <Your name here>
    d. **UPDATE** Id: A random GUID. Use **KAPE** to generate one.
    e. **UPDATE** BinaryUrl: https://ericzimmerman.github.io/#!index.md
    f. **UPDATE** FileMask: “”
    g. Under the **FIRST** Processors section:
        i. **UPDATE** Executable: AmcacheParser.mkape
    h. Under the **SECOND** Processors section:
        i. **UPDATE** Executable: AppCompatCacheParser.mkape
    i. Under the **THIRD** Processors section:
        i. **UPDATE** Executable: PECmd.mkape
    j. Under the **FOURTH** Processors section:
        i. **UPDATE** Executable: LECmd.mkape
    k. **ADD** a **FIFTH** Processors section:
        i. **MAKE** Executable: JLECmd.mkape

39. When you are done, your file should look like this (except for the GUID as yours will be different).

ExecutionAndLnkJump.mkape

```
Description: 'Collect evidence of execution, lnk files, and jumplists'
Category: EvidenceOfExecution
Author: Kevin Ripa
Version: 1
Id: 151e5a9b-a684-4369-ada7-2659c5367db8
BinaryUrl: https://ericzimmerman.github.io/#!index.md
ExportFormat: csv
FileMask: ""
Processors:
    -
        Executable: AmcacheParser.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: AppCompatCacheParser.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: PECmd.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: LECmd.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: JLECmd.mkape
        CommandLine: ""
        ExportFormat: ""
```

40. Save the file and validate it.

```
.\kape.exe --mlist .
```

Correct any errors, resave, and revalidate with the above command.

41. The compound module is now finished and can be used to process all the file types as defined in the processors referenced.

42. We are now ready to collect and process in one step. Open a **PowerShell** prompt with administrator privileges and navigate to **C:\Tools\KAPE**, then run the following command:

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target
ExecutionAndLnkJump --tflush --msource C:\Temp\out --mdest
C:\Temp\mout --module ExecutionAndLnkJump --mflush
```

43. **KAPE** will now collect files to **C:\Temp\out**, process the files found in **C:\Temp\out**, and place the parsing results into C**:\Temp\mout**.

```
KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)
Command line: --tdest C:\Temp\out --tsource E: --target ExecutionAndLnkJump --tflush --msource C:\Temp\out --mdest C:\Te

Using Target operations
        Flushing target destination directory 'C:\Temp\out'
        Creating target destination directory 'C:\Temp\out'
Found 2 targets. Expanding targets to file list...
Found 252 files. Beginning copy...

Copied 237 (Deduplicated: 15) out of 252 files in 2.3686 seconds. See '*_copylog.txt' in 'C:\Temp\out' for copy details


Using Module operations
        Flushing module destination directory 'C:\Temp\mout'
        Creating module destination directory 'C:\Temp\mout'
Module 'AmcacheParser': Found 1 processor
        Found processor 'Executable: AmcacheParser.exe, Cmd line: -f %sourceFile% --csv %destinationDirectory% -i, Expor
Module 'AppCompatCacheParser': Found 1 processor
        Found processor 'Executable: AppCompatCacheParser.exe, Cmd line: -f %sourceFile% --csv %destinationDirectory%, E
Module 'PECmd': Found 3 processors
        Found processor 'Executable: PECmd.exe, Cmd line: -d %sourceDirectory% --csv %destinationDirectory% -q, Export:
Module 'LECmd': Found 3 processors
        Found processor 'Executable: LECmd.exe, Cmd line: -d %sourceDirectory% --csv %destinationDirectory% -q, Export:
Module 'JLECmd': Found 3 processors
        Found processor 'Executable: JLECmd.exe, Cmd line: -d %sourceDirectory% --csv %destinationDirectory% -q, Export
Discovered 5 processors to run.
Executing modules with file masks...
        Skipping 'AmcacheParser.exe': No matching files found for 'Amcache.hve'!
        Skipping 'AppCompatCacheParser.exe': No matching files found for 'SYSTEM'!
Executing remaining modules...
        Running 'PECmd.exe': -d C:\Temp\out --csv C:\Temp\mout\ProgramExecution -q
        Running 'LECmd.exe': -d C:\Temp\out --csv C:\Temp\mout\FileFolderAccess -q
        Running 'JLECmd.exe': -d C:\Temp\out --csv C:\Temp\mout\FileFolderAccess -q
Executed 5 processors in 1.9213 seconds

Total execution time: 4.3198 seconds
```

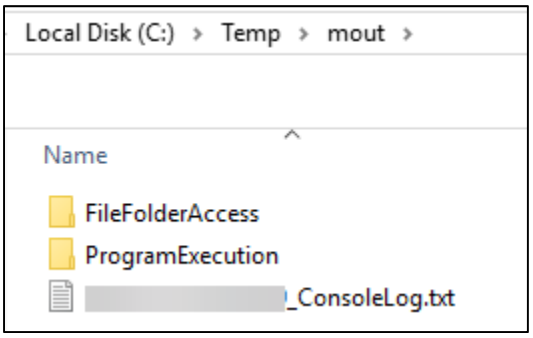
Notice that processing for **AmcacheParser.exe** and **AppCompatCacheParser.exe** were skipped because **KAPE** didn't find any files that were collected that matched the specifications for those modules. Both modules look for Registry hives, and since we did not collect registry hives with the target options, those two processors did not have anything to do.

ExecutionAndLnkJump.tkape could be modified however, to pull Registry hives (by adding a Target referencing **RegistryHives.tkape**), and this would then give us different results in the modules phase.

44. With collection and processing complete, we can now analyze our results.

45. Because modules typically generate CSV data, we will use **Timeline Explorer** to analyze the output from the previous commands. Open **File Explorer** and navigate to **C:\Temp\mout**.

46. This folder contains everything generated by **KAPE** when it ran our module configuration.



The **<date/time>_ConsoleLog.txt** file contains all the text that **KAPE** generated as it ran. The two directories will contain the details from each of the processors that **KAPE** ran, organized by category. This is useful because it allows for multiple artifacts to be related to each other, thereby making it easier to review artifacts that can be used for correlation and corroboration.

Under each directory is one or more files related to the directory name that contains them. For example, in the **FileFolderAccess** directory, we see results from **lnk** and **jumplist** parsing.

47. Start **Timeline Explorer** using the shortcut in the **Utilities** fence on the **Desktop**.

48. Once **Timeline Explorer** starts, load the **LECmd_Output.csv** file from the **FileFolderAccess** directory by either dragging and dropping, or using **File | Open** menu option. Once the file is loaded, answer the following questions:

    a. How many files with a **Relative Path** that contains a **.jpg** extension were opened according to **lnk** file analysis?

    _____

    b. What user profile are the .**jpg** files found under? (**Hint**: Check the **Working Directory** column)

    _____

    c. What is the name of the .**jpg** that was **CREATED** in 2011?

    _____

    d. How many **Word** documents were opened according to **lnk** file analysis? (**Hint**: Use the **Local Path** column)

    _____

    e. What do all the **Word** files seem to have in common?

    _____

f. What is the **volume serial number** that each of the **Word** documents are found on?

_____

49. Load the **AutomaticDestinations.csv** file located in the **FileFolderAccess** directory into **Timeline Explorer** and answer the following questions:

    a. How many unique **App IDs** are represented?

    _____

    b. List the directories accessed from the **controller** host (**Hint**: Use the **Hostname** column or the Power filter)

    _____

    c. How many entries exist related to the **Control Panel**? (**Hint**: Use the **App ID Description** column)

    _____

50. Load the **PECmd_Output.csv** file located in the **ProgramExecution** directory into **Timeline Explorer** and answer the following questions:

    a. How many times was **A.EXE** executed in total?

    _____

    b. For **A.EXE** that was executed out of the **Windows\temp** directory, what was the last date and time it was executed? (**Hint**: use the Power filter criteria ***windows\temp***)

    _____

    c. When was the last date and time **EXCEL.EXE** was run? **BONUS**: When was the first date and time **EXCEL.EXE** was executed?

    _____

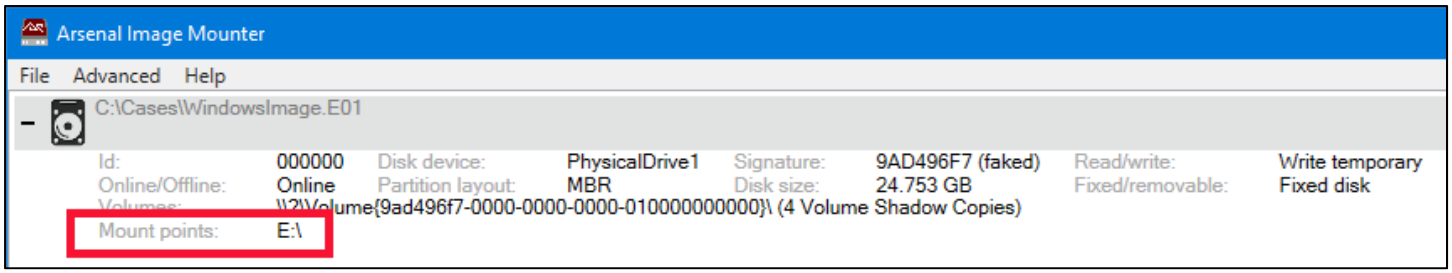
    _____

    d. What is the name of the executable that was run the most?

    _____

Exercise Questions with Step-by-Step

1. Let's take advantage of the target module we just made to find and extract **zip** archives from a drive. Verify the **WindowsImage.E01** file is mounted in **Arsenal Image Mounter (AIM)** and verify the drive letter in the **AIM** interface. In the image below, the E01 has been mounted to **E:\.**



2. To use our new target, open a **PowerShell** window with Administrator privilege and navigate to **C:\Tools\KAPE**. Once there, run .**kape.exe** without any arguments. It will look something like this (partial output shown):

```
PS C:\Tools\KAPE> .\kape.exe

KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)

        tsource         Target source drive to copy files from (C, D:, or F:\ f
        target          Target configuration to use
        tdest           Destination directory to copy files to. If --vhdx, --vh
s set, files will end up in VHD(X) container or zip file
        tlist           List available targets. Use . for Targets directory or
irectory under Targets.
        tdetail         Dump target file details
        tflush          Delete all files in 'tdest' prior to collection
```

3. Since we just made a target file, we will use the target related options. In **PowerShell**, enter the following command, then press **Enter.**

   NOTE: Be sure to update the drive letter used with **--tsource** to what **AIM** shows!

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --
tflush
```

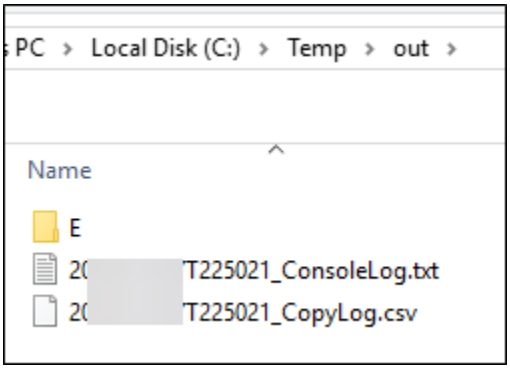
4. **KAPE** will now run the **ZipFiles** target against the mounted drive, and copy any files found, to the **C:\Temp\out** directory.

```
PS C:\Tools\KAPE> .\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --tflush
KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)

KAPE directory: C:\Tools\KAPE
Command line: --tdest C:\Temp\out --tsource E: --target ZipFiles --tflush

Using Target operations
        Flushing target destination directory 'C:\Temp\out'
        Creating target destination directory 'C:\Temp\out'
Found 1 targets. Expanding targets to file list...
Found 4 files in 13.390 seconds. Beginning copy...

Copied 4 out of 4 files in 14.5556 seconds. See '*_CopyLog.csv' in 'C:\Temp\out' for copy
details

Total execution time: 14.5853 seconds
```

   a. How many total files did **KAPE** find?

      4 files

5. Open **File Explorer** and navigate to **C:\Temp\out**.



   Notice there is a directory and several files. The directory contains all the files (and their full directory paths) that were located under the **E:** drive. The **CopyLog.csv** file contains details of files copied including timestamps, SHA-1, source and destination path, and so on which makes it easier to filter, search, etc. Finally, the **ConsoleLog.txt** file contains a copy of everything that was displayed on the screen while **KAPE** ran. This is useful for reviewing any messages related to errors, etc., should any occur.

6. Open the **<*date/time*>_CopyLog.csv** and answer the following questions:

   a. What are the first 5 characters of the **SHA-1** value for **Undercover Agent-List-Classified\Agents-List-CLASSIFIED-TOP-SECRET.zip**?

      F65FE

b. What is the **CreatedOn** date for file **$ISEBKEC.zip**?

2012-03-16 20:27:20.5009230

7. Open the **<*date/time*>_ConsoleLog.txt** file and review its contents, then close both files that are open.

8. Rerun the previous command by pressing the Up arrow. Add the **`--debug`** switch to the end of the command, then press **Enter.**

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --
tflush --debug
```

9. When **KAPE** is finished running, open the **<*date/time*>_ConsoleLog.txt** file and review its contents.

   a. What kinds of information is available in the **<*date/time*>_ConsoleLog.txt** file now that was not in the previous one?

   Several additional things are shown, including every match KAPE found, directory creation, and copy confirmation

   b. Why would you want to use the **`--debug`** option?

   The --debug option is useful for testing new targets as well as when collecting over a slow link using something like F-Response. In this situation, the fact that you can see the file copy operations lets you know what KAPE is doing as it works.

10. Repeat the last command, but this time, add the **--trace** option, then press **Enter .**

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target ZipFiles --
tflush --debug --trace
```

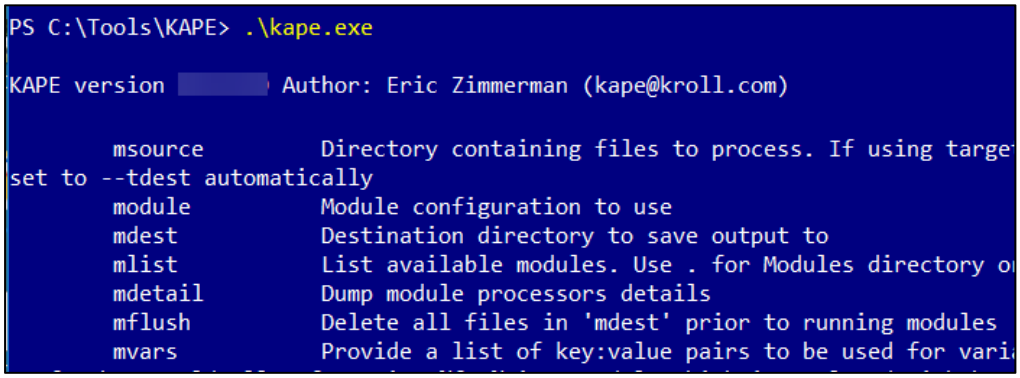
11. Review the **<*date/time*>_ConsoleLog.txt** file again (or simply scroll up in the **PowerShell** window). Trace adds a significant amount of information to the output when compared to the **--debug** option.

12. Finally, spend a few moments exploring the contents of **C:\Temp\out\E** and its various subdirectories to see how **KAPE** found and saved the target files. Notice that all the timestamps have been preserved for each directory and file that was recovered.

    The debug and trace options are useful for troubleshooting or viewing file progress for slower connections.

13. Next, let's process the files we just collected with the module file we created earlier. Run **kape.exe** without any options and look for the switches related to modules.

```
.\kape.exe
```



14. In **PowerShell**, enter the following command, then press **Enter.**

```
.\kape.exe --msource C:\Temp\out --mdest C:\Temp\mout --module
ZipFileContents --mflush
```

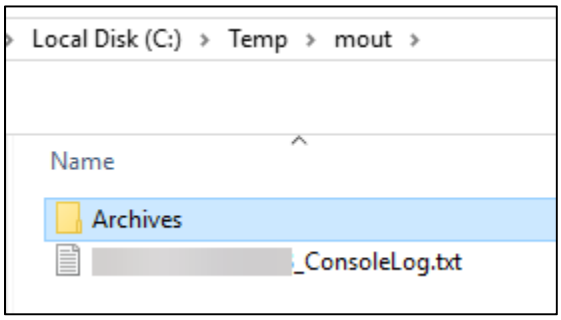
15. **KAPE** will now run the **ZipFileContents** module against the files in **C:\Temp\out** (which is where we told **KAPE** to save files collected via the target from earlier in the exercise) and save the results to **C:\Temp\mout**.

```
PS C:\Tools\KAPE> .\kape.exe --msource C:\Temp\out --mdest C:\Temp\mout --module ZipFileContents --mflush
KAPE version Author: Eric Zimmerman (kape@kroll.com)

KAPE directory: C:\Tools\KAPE
Command line: --msource C:\Temp\out --mdest C:\Temp\mout --module ZipFileContents --mflush

Using Module operations
        Flushing module destination directory 'C:\Temp\mout'
        Creating module destination directory 'C:\Temp\mout'
                Found processor 'Executable: 7z.exe, Cmd line: l %SourceFile%, Export: txt, Append: False'!
Discovered 1 processor to run.
Executing modules with file masks...
        Running '7z.exe': l "C:\Temp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$ISEBKEC.zip"
                ERROR: C:\Temp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$ISEBKEC.zip : C:\Te
mp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$ISEBKEC.zip
                Open ERROR: Can not open the file as [zip] archive
                ERRORS:
                Is not archive
        Running '7z.exe': l "C:\Temp\out\E\$Recycle.Bin\S-1-5-21-2036804247-3058324640-2116585241-1109\$RSEBKEC.zip"
                Output file updated to 'C:\Temp\mout\Archives\7zFileContents_1.txt'
        Running '7z.exe': l "C:\Temp\out\E\Users\nromanoff\Documents\Undercover Agent-List-Classified\Agents-List-CLA
SSIFIED-TOP-SECRET.zip"
                Output file updated to 'C:\Temp\mout\Archives\7zFileContents_2.txt'
        Running '7z.exe': l "C:\Temp\out\E\Users\nromanoff\AppData\LocalLow\Adobe\Acrobat\10.0\rdrmessage.zip"
                Output file updated to 'C:\Temp\mout\Archives\7zFileContents_3.txt'
Executing remaining modules...
Executed 1 processor in 0.2502 seconds

Total execution time: 0.2937 seconds
```

a. How many total files did **KAPE** process?

4 files

16. Open **File Explorer** and navigate to **C:\Temp\mout**.



Again, we see a directory and a ***<date/time>*_ConsoleLog.txt** file. The **ConsoleLog** file serves the same purpose as we saw earlier. Recall from the module configuration that we specified a **Category** of "**Archives**" and here we see a directory with the same name. This allows you to tie specific processors to a common group so related files end up in the same directory.

17. Open the **Archives** directory in **File Explorer**.

18. Open the text files in **EditPad Lite**.

a. What do the files contain?

The console output showing information about the zip file, including information about all the files contained in the zip file. The full path to the zip archive is also shown.

b. Locate the **7zFileContents** file that relates to the **Agents** list. What is the name of the archive file?

Opening and viewing each text file in the Archives directory shows that one of them contains a list of Excel spreadsheets related to undercover agents.

These Excel files are in a file named *Agents-List-CLASSIFIED-TOP-SECRET.zip*

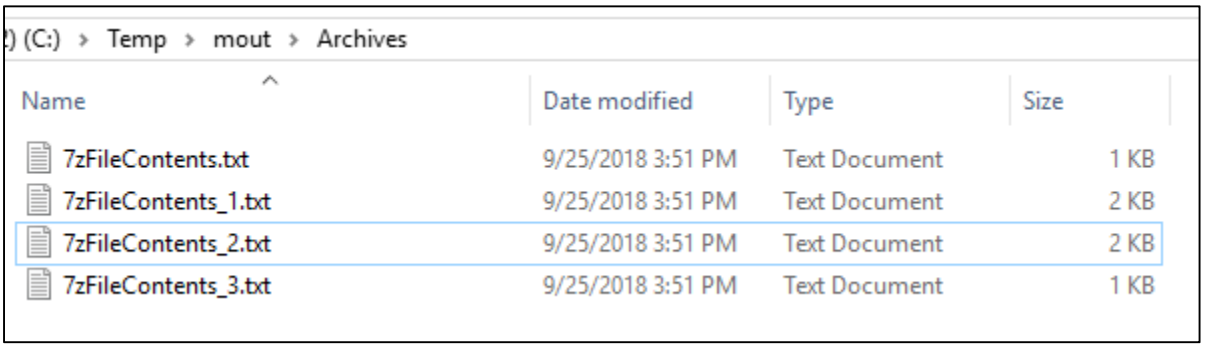
c. How many files does this **zip** archive contain?

4 files

d. What is the total size for the files (uncompressed)?

1690486 bytes, which can be seen under the Size column in the text file where we found the filename above.

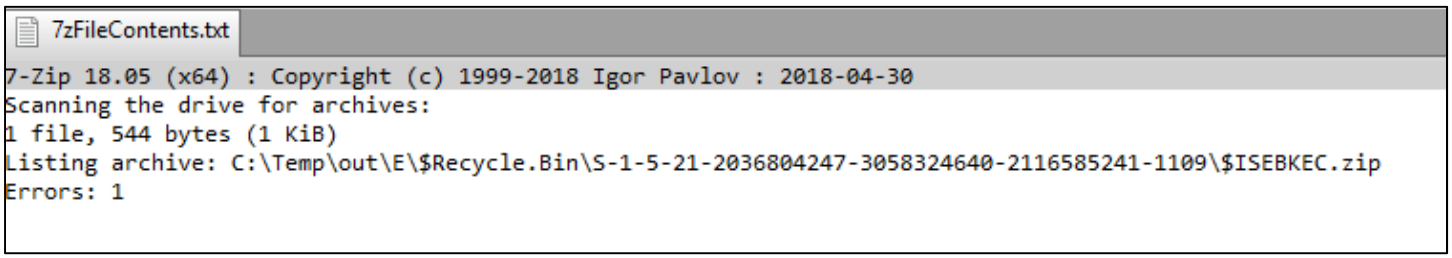
19. Recall from earlier that **KAPE** found four **zip** files. Here we see the same number of text files. **KAPE** is smart enough to not overwrite files as each target is found and when this happens, **KAPE** appends a unique counter to the end of the file name, resulting in the file names shown below.



20. Spend a few minutes exploring the remaining **ZipFileContents** text files.

    a. Did any of them run into errors? If so, what was the name of the **zip** file?

    Yes, one error. The name of the zip file that errored is $ISEBKEC.zip and this can be found in 7zFileContents.txt



    b. Locate the **zip** file from above and open it by double clicking it. Does it open? (**Hint**: You will have to go to the target's output folder, which is shown in the text file where you got the information from the previous question)

    The zip file above will not open.

    BONUS: Why does the **zip** file not open? Can you find the contents of the **zip** file?

    The zip file above will not open because it is not really a zip file, even though it ends in .zip. This is how Windows keeps track of files and directories that are deleted and end up in the Recycle Bin. If you right click on the zip file you found above and open it with EditPad Lite, you will see it contains the path to the original file that was deleted. Each file has a $I (contains information about the original file) and a corresponding $R file (contains the original file data; i.e. it's the original file).

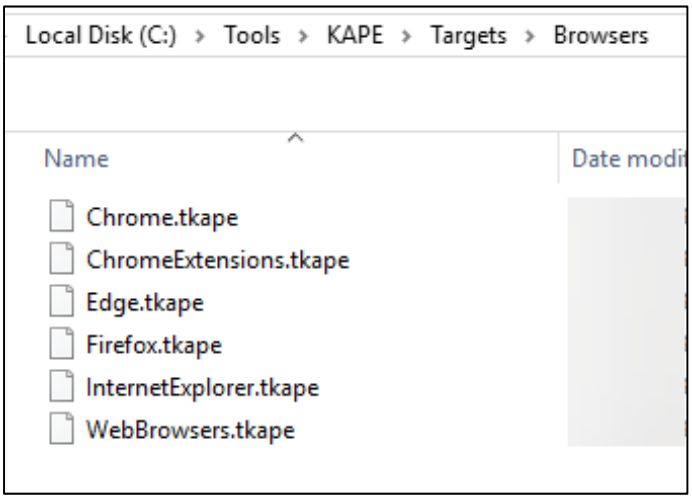
21. In the previous sections, we saw how **KAPE** can be used to both collect files via targets, and process files via modules. In this section, we will combine the two operations into a single step.

22. For both targets and modules, **KAPE** has command line switches that list all available targets and modules, **`--tlist`** and **`--mlist`**.

23. Open a **PowerShell** window with administrator rights and navigate to `C:\Tools\KAPE`

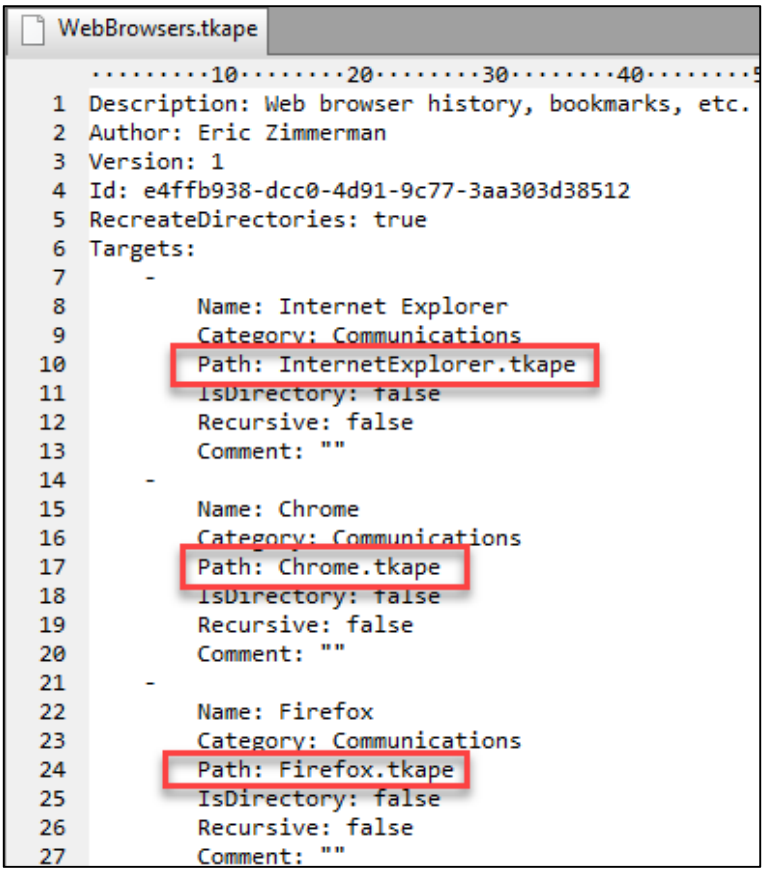
24. While we could run any of the targets or modules independently and execute **KAPE** multiple times (one per target or module), **KAPE** can define targets and modules that refer to other targets and modules.

25. To create a compound target file, open **File Explorer** and navigate to **C:\Tools\KAPE\Targets\Browsers**.



26. Right click on **WebBrowsers.tkape** and open it with **EditPad**.

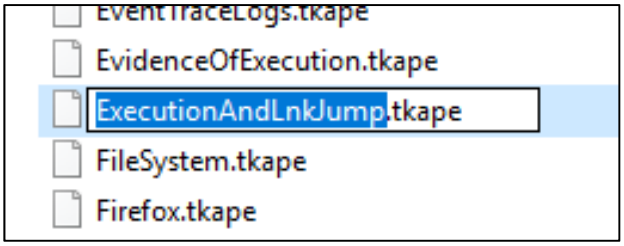
    Notice how the **Path** property references another target configuration. Because there are many different browsers, target files were created that specifically look for a single browser, but in most cases, we are interested in knowing about all web browsers, so a compound target was made that pulls in four web browsers (although the image below is only showing the first three).



27. For our purposes, we want to build a compound target that will collect **evidence of execution** artifacts as well as **lnk** files and **jumplists**. Since targets already exist for those things, we will create a new compound target and reference the existing target configurations.

28. Close any currently opened files in **EditPad**. In **File Explorer**, select **WebBrowsers.tkape** by clicking it once, then press **CTRL-C**, **CTRL-V** to make a copy of the file.

29. Select **WebBrowsers - Copy.tkape** by clicking it once, then press **F2** to rename the file. Rename the file **`ExecutionAndLnkJump.tkape`**



30. Right click on **ExecutionAndLnkJump.tkape** and open it with **EditPad**. Edit the file to reflect the following:

    a. **UPDATE** Description: Collect evidence of execution, lnk files, and jumplists
    b. **UPDATE** Author: <Your name here>
    c. **UPDATE** Id: A random GUID. Use **KAPE** to generate a GUID.
    d. Under the **FIRST** Targets section:
        i. **UPDATE** Name: Evidence Of Execution
        ii. **UPDATE** Category: Execution
        iii. **UPDATE** Path: EvidenceOfExecution.tkape
    e. Under the **SECOND** Targets section:
        i. **UPDATE** Name: Lnk and JumpLists
        ii. **UPDATE** Category: FileFolderOpening
        iii. **UPDATE** Path: LnkFilesAndJumpLists.tkape
    f. **DELETE** the remaining targets (Firefox, Edge, etc.).

31. When you are done, your file should look like this (except for the GUID as yours will be different).

ExecutionAndLnkJump.tkape

```
Description: Collect evidence of execution, lnk files, and jumplists
Author: Kevin Ripa
Version: 1
Id: 9cf813a0-281c-4208-89c6-c93d3695e4df
RecreateDirectories: true
Targets:
    -
        Name: Evidence of Execution
        Category: Execution
        Path: EvidenceOfExecution.tkape
        IsDirectory: false
        Recursive: false
        Comment: ""
    -
        Name: Lnk and JumpLists
        Category: FileFolderOpening
        Path: LnkFilesAndJumpLists.tkape
        IsDirectory: false
        Recursive: false
        Comment: ""
```

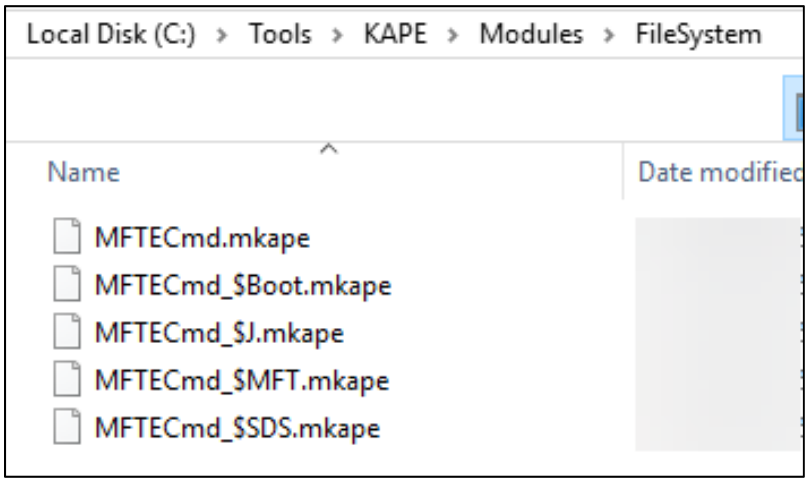
32. Save the file and validate it.

```
.\kape.exe --tlist .
```

    Correct any errors, resave, and revalidate with the above command.

33. The compound target is now finished and can be used to collect all the file types as defined in the two target files referenced. Before we collect files using this target, let's follow the same process, but this time with a module file.

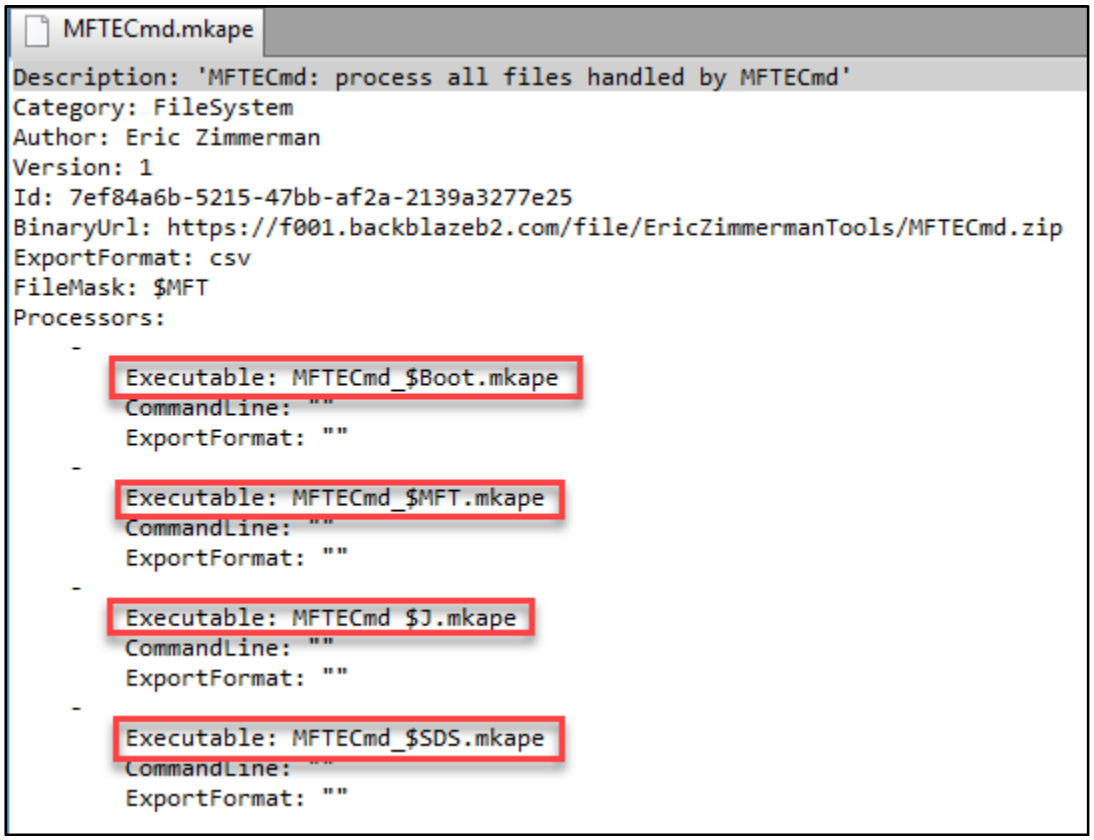
34. Using **File Explorer**, navigate to **C:\Tools\KAPE\Modules\FileSystem**



Here we see a list of all the available modules.

35. Right click on **MFTECmd.mkape**, then open the file with **EditPad**.

Notice how the **Executable** property references another module file. Just as we saw with browsers, there are many **evidence of execution** related artifacts, so we will create a module that targets these, as well as **lnk** files and **jumplists**.



36. Close any currently opened files in **EditPad**. In **File Explorer**, select **MFTECmd.mkape** by clicking it once, then press **CTRL-C**, **CTRL-V** to make a copy of the file.

37. Select **MFTECmd - Copy.mkape** by clicking it once, then press **F2** to rename the file. Rename the file **`ExecutionAndLnkJump.mkape`**

38. Right click on **ExecutionAndLnkJump.mkape** and open it with **EditPad**. Edit the file to reflect the following:
    l. **UPDATE** Description: Collect evidence of execution, lnk files, and jump lists
    m. **UPDATE** Category: EvidenceOfExecution
    n. **UPDATE** Author: <Your name here>
    o. **UPDATE** Id: A random GUID. Use **KAPE** to generate one.
    p. **UPDATE** BinaryUrl: https://ericzimmerman.github.io/#!index.md
    q. **UPDATE** FileMask: “”
    r. Under the **FIRST** Processors section:
        i. **UPDATE** Executable: AmcacheParser.mkape
    s. Under the **SECOND** Processors section:
        i. **UPDATE** Executable: AppCompatCacheParser.mkape
    t. Under the **THIRD** Processors section:
        i. **UPDATE** Executable: PECmd.mkape
    u. Under the **FOURTH** Processors section:
        i. **UPDATE** Executable: LECmd.mkape
    v. **ADD** a **FIFTH** Processors section:
        i. **MAKE** Executable: JLECmd.mkape

39. When you are done, your file should look like this (except for the GUID as yours will be different).

ExecutionAndLnkJump.mkape

```
Description: 'Collect evidence of execution, lnk files, and jumplists'
Category: EvidenceOfExecution
Author: Kevin Ripa
Version: 1
Id: 151e5a9b-a684-4369-ada7-2659c5367db8
BinaryUrl: https://ericzimmerman.github.io/#!index.md
ExportFormat: csv
FileMask: ""
Processors:
    -
        Executable: AmcacheParser.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: AppCompatCacheParser.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: PECmd.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: LECmd.mkape
        CommandLine: ""
        ExportFormat: ""
    -
        Executable: JLECmd.mkape
        CommandLine: ""
        ExportFormat: ""
```

40. Save the file and validate it.

```
.\kape.exe --mlist .
```

Correct any errors, resave, and revalidate with the above command.

41. The compound module is now finished and can be used to process all the file types as defined in the processors referenced.

42. We are now ready to collect and process in one step. Open a **PowerShell** prompt with administrator privileges and navigate to **C:\Tools\KAPE**, then run the following command:

```
.\kape.exe --tdest C:\Temp\out --tsource E: --target
ExecutionAndLnkJump --tflush --msource C:\Temp\out --mdest
C:\Temp\mout --module ExecutionAndLnkJump --mflush
```

43. **KAPE** will now collect files to **C:\Temp\out**, process the files found in **C:\Temp\out**, and place the parsing results into C**:\Temp\mout**.

```
KAPE version [redacted] Author: Eric Zimmerman (kape@kroll.com)
Command line: --tdest C:\Temp\out --tsource E: --target ExecutionAndLnkJump --tflush --msource C:\Temp\out --mdest C:\Te

Using Target operations
        Flushing target destination directory 'C:\Temp\out'
        Creating target destination directory 'C:\Temp\out'
Found 2 targets. Expanding targets to file list...
Found 252 files. Beginning copy...

Copied 237 (Deduplicated: 15) out of 252 files in 2.3686 seconds. See '*_copylog.txt' in 'C:\Temp\out' for copy details


Using Module operations
        Flushing module destination directory 'C:\Temp\mout'
        Creating module destination directory 'C:\Temp\mout'
Module 'AmcacheParser': Found 1 processor
        Found processor 'Executable: AmcacheParser.exe, Cmd line: -f %sourceFile% --csv %destinationDirectory% -i, Expor
Module 'AppCompatCacheParser': Found 1 processor
        Found processor 'Executable: AppCompatCacheParser.exe, Cmd line: -f %sourceFile% --csv %destinationDirectory%, E
Module 'PECmd': Found 3 processors
        Found processor 'Executable: PECmd.exe, Cmd line: -d %sourceDirectory% --csv %destinationDirectory% -q, Export:
Module 'LECmd': Found 3 processors
        Found processor 'Executable: LECmd.exe, Cmd line: -d %sourceDirectory% --csv %destinationDirectory% -q, Export:
Module 'JLECmd': Found 3 processors
        Found processor 'Executable: JLECmd.exe, Cmd line: -d %sourceDirectory% --csv %destinationDirectory% -q, Export
Discovered 5 processors to run.
Executing modules with file masks...
        Skipping 'AmcacheParser.exe': No matching files found for 'Amcache.hve'!
        Skipping 'AppCompatCacheParser.exe': No matching files found for 'SYSTEM'!
Executing remaining modules...
        Running 'PECmd.exe': -d C:\Temp\out --csv C:\Temp\mout\ProgramExecution -q
        Running 'LECmd.exe': -d C:\Temp\out --csv C:\Temp\mout\FileFolderAccess -q
        Running 'JLECmd.exe': -d C:\Temp\out --csv C:\Temp\mout\FileFolderAccess -q
Executed 5 processors in 1.9213 seconds

Total execution time: 4.3198 seconds
```

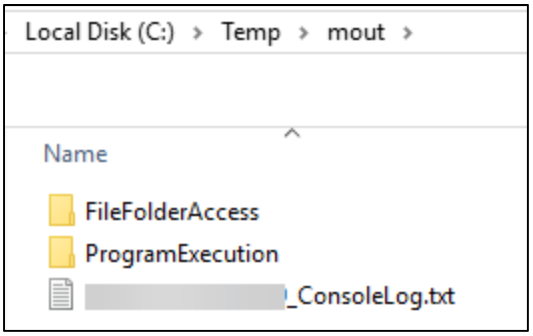
Notice that processing for **AmcacheParser.exe** and **AppCompatCacheParser.exe** were skipped because **KAPE** didn't find any files that were collected that matched the specifications for those modules. Both modules look for Registry hives, and since we did not collect registry hives with the target options, those two processors did not have anything to do.

ExecutionAndLnkJump.tkape could be modified however, to pull Registry hives (by adding a Target referencing **RegistryHives.tkape**), and this would then give us different results in the modules phase.

44. With collection and processing complete, we can now analyze our results.

45. Because modules typically generate CSV data, we will use **Timeline Explorer** to analyze the output from the previous commands. Open **File Explorer** and navigate to **C:\Temp\mout**.

46. This folder contains everything generated by **KAPE** when it ran our module configuration.



The **<date/time>_ConsoleLog.txt** file contains all the text that **KAPE** generated as it ran. The two directories will contain the details from each of the processors that **KAPE** ran, organized by category. This is useful because it allows for multiple artifacts to be related to each other, thereby making it easier to review artifacts that can be used for correlation and corroboration.

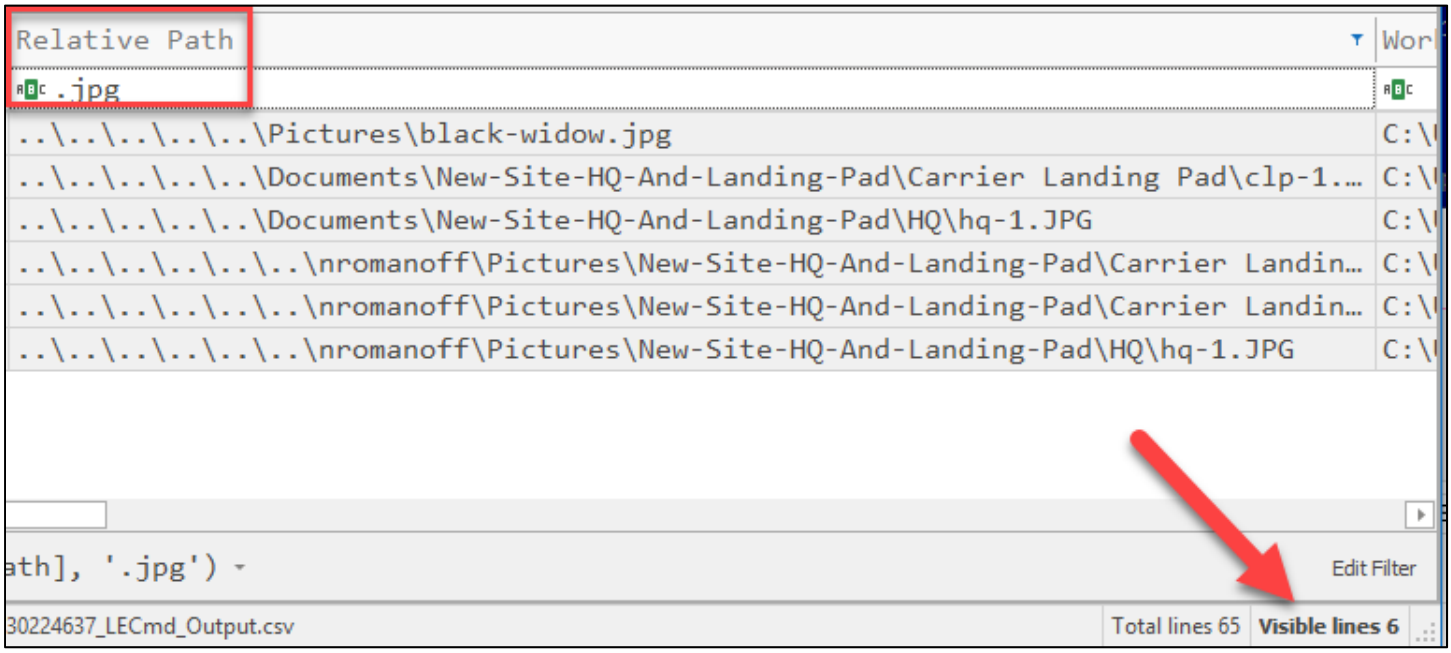
Under each directory is one or more files related to the directory name that contains them. For example, in the **FileFolderAccess** directory, we see results from **lnk** and **jumplist** parsing.

47. Start **Timeline Explorer** using the shortcut in the **Utilities** fence on the **Desktop**.

48. Once **Timeline Explorer** starts, load the **LECmd_Output.csv** file from the **FileFolderAccess** directory by either dragging and dropping, or using **File | Open** menu option. Once the file is loaded, answer the following questions:

    a. How many files with a **Relative Path** that contains a **.jpg** extension were opened according to **lnk** file analysis?

    Filtering on the Relative Path column for .jpg shows 6 files.



    b. What user profile are the. **jpg** files found under? (**Hint**: Check the **Working Directory** column)

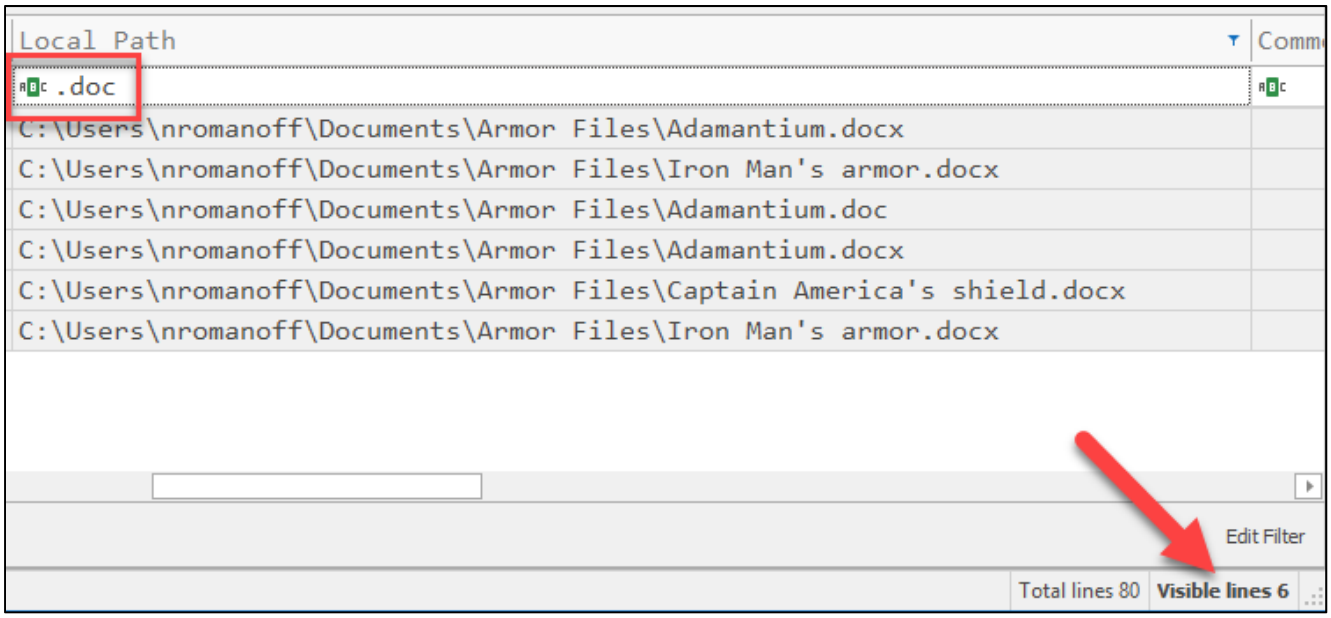
    nromanoff

    c. What is the name of the .**jpg** that was **CREATED** in 2011?

    Looking at the Target Created column shows one file created in 2011, *black-widow.jpg*

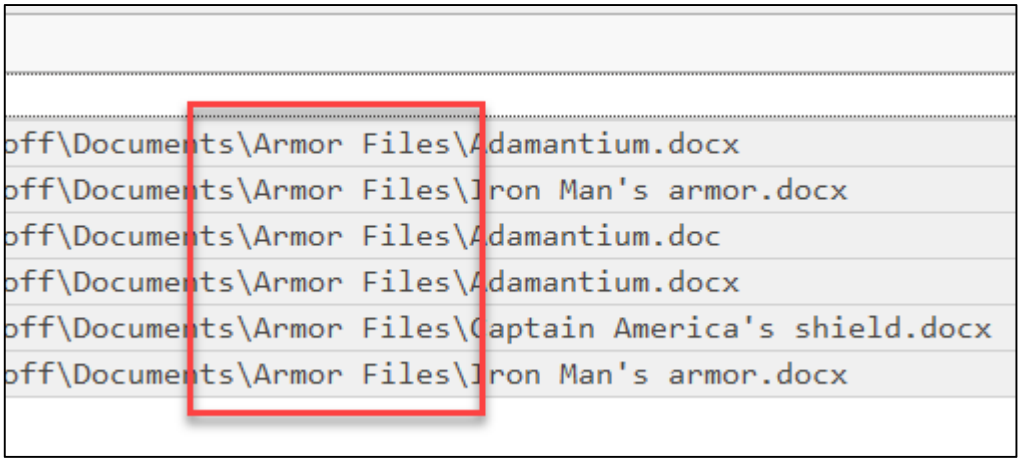
d. How many **Word** documents were opened according to **lnk** file analysis? (**Hint**: Use the **Local Path** column)

Filtering for .doc (which also gets us .docx files) shows there are six Word documents opened.



e. What do all the **Word** files seem to have in common?

They are all related to armor.



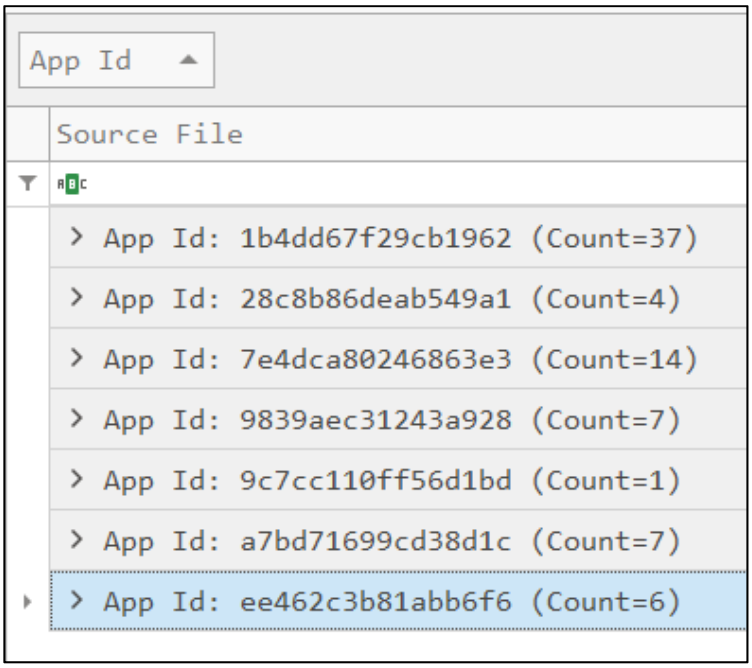
f. What is the **volume serial number** that each of the **Word** documents are found on?

AC036525

49. Load the **AutomaticDestinations.csv** file located in the **FileFolderAccess** directory into **Timeline Explorer** and answer the following questions:

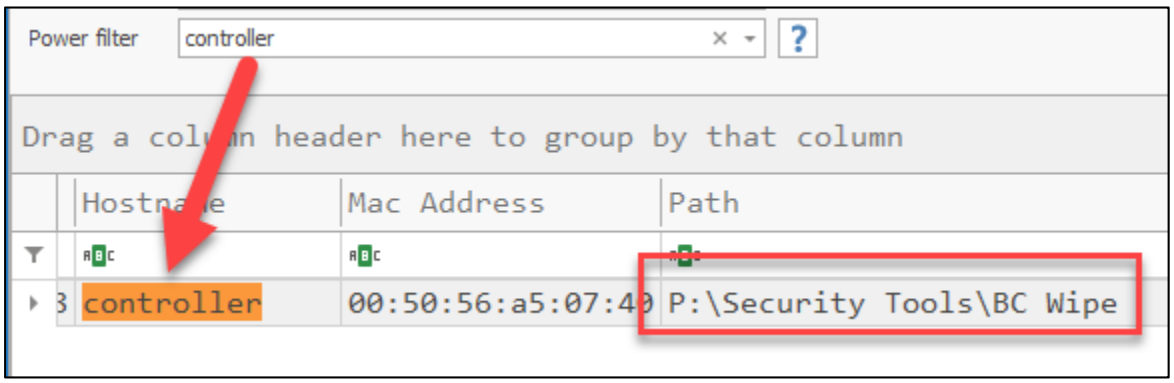
    a. How many unique **App IDs** are represented?

    Grouping by App Id column shows there are seven unique App IDs.



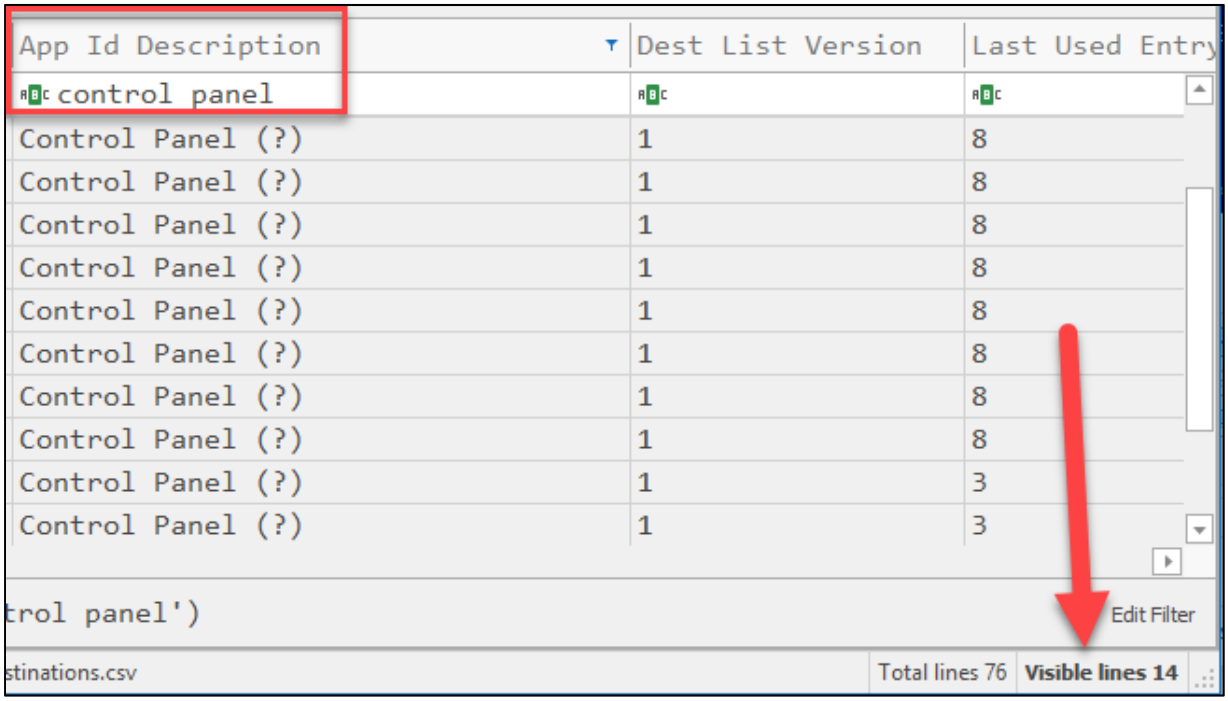
    b. List the directories accessed from the **controller** host (**Hint**: Use the **Hostname** column or the Power filter)

    There is one row that contains controller, and the directory accessed is *Security Tools\BC Wipe*



c. How many entries exist related to the **Control Panel**? (**Hint**: Use the **App ID Description** column)

14 entries.



The Power filter could have also been used here with a value of **`control +panel`**

50. Load the **PECmd_Output.csv** file located in the **ProgramExecution** directory into **Timeline Explorer** and answer the following questions:

a. How many times was **A.EXE** executed in total?

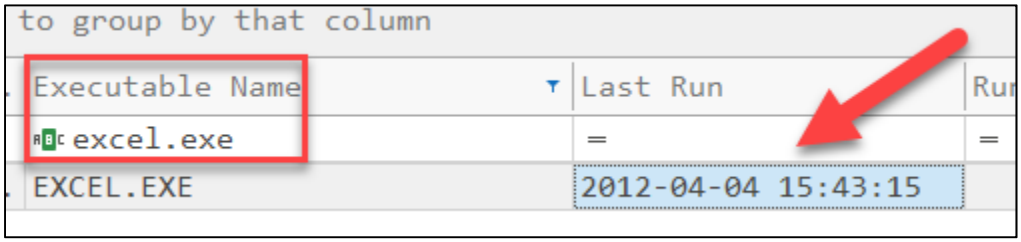
Filtering for a.exe shows that it ran 26 + 1541 times, or 1567 times total

b. For **A.EXE** that was executed out of the **Windows\temp** directory, what was the last date and time it was executed? (**Hint**: use the Power filter criteria ***windows\temp***)

Adding a Power filter criteria of windows\temp removes one of the two entries from above, which leaves us with one row. The Run Count of this row is 26 and the last run time is 2012-04-04 00:43:07.

c. When was the last date and time **EXCEL.EXE** was run? **BONUS**: When was the first date and time **EXCEL.EXE** was executed?

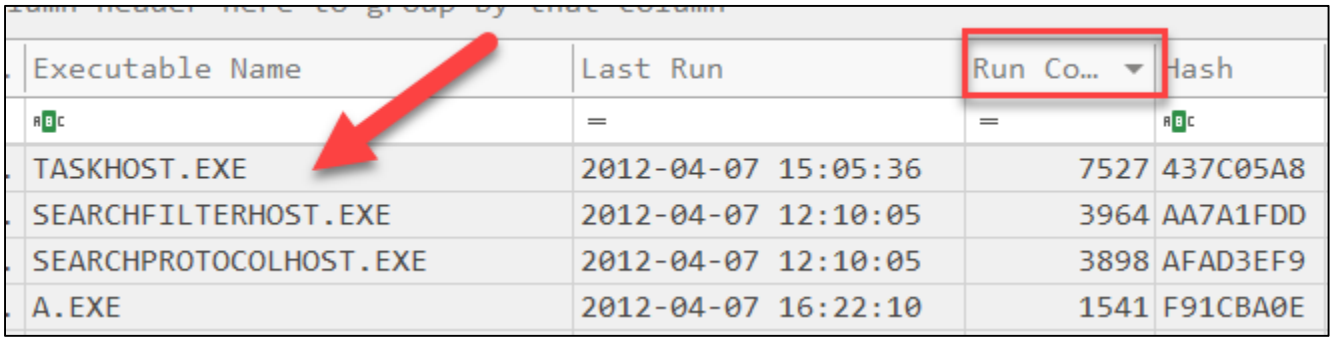
Filtering for EXCEL.EXE shows that the Last Run time is 2012-04-04 15:43:15



The first time Excel was executed would be 2012-04-04 15:42:03 – 10 seconds based on SourceCreated timestamp.

d. What is the name of the executable that was run the most?

Sorting on the Run Count column and looking for the largest number shows us that the corresponding executable is TASKHOST.EXE



Executable Name	Last Run	Run Co…	Hash
TASKHOST.EXE	2012-04-07 15:05:36	7527	437C05A8
SEARCHFILTERHOST.EXE	2012-04-07 12:10:05	3964	AA7A1FDD
SEARCHPROTOCOLHOST.EXE	2012-04-07 12:10:05	3898	AFAD3EF9
A.EXE	2012-04-07 16:22:10	1541	F91CBA0E

Exercise—Key Takeaways

- Storage devices are growing faster than our ability to image them efficiently.
- Battlefield forensics is the answer to this problem, as it allows for quickly collecting key data to move your case forward.
- **KAPE** is a powerful and flexible tool that allows for rapid collection and processing of key artifacts.
- By leveraging **KAPE** in your process you can focus on the important details vs. waiting hours for a full disk image to complete.
- Even in a very limited triage collection, a wide range of questions can be answered which can help drive your case forward efficiently.

This page intentionally left blank.

Exercise 4.3A—Network Acquisition - Cyberduck

Background

Network collection is a methodology that is becoming more and more necessary as our world becomes more interconnected, with data residing in far more places than on a hard drive in a desktop computer. From hosted email to web sites to cloud storage, the range of situations that involve network acquisition is a mile wide and deep.

For this reason, it is important to be proficient with both command line and GUI tools, as well as open source and commercial tools. In this lab, we will explore several options that allow for collecting against a wide range of network-based storage, from computers on the same network to remote machines in the far reaches of the Internet. We want to understand how the process works as well as how to do the process. We will see examples of how to effectively use CLI and GUI based tools, as well as get a feel for why we might prefer one over the other.

Due to the nature of this type of collection, it may not be possible in every situation, to successfully complete these exercises in the classroom. This is primarily true on air-gapped networks without Internet access, or on networks with very strict controls on ingress and egress as it relates to TCP/IP traffic. If this exercise is not possible to complete for one of the reasons above, we highly encourage you to complete the exercise in an environment where you do have Internet access, such as your home or using a Wi-Fi hotspot. This way you get a feel for not only how the collection tools work, but also understand the prerequisites you may have to ensure are in place before a successful collection can occur.

Objectives

- Use Cyberduck and AWS CLI to acquire an Amazon S3 bucket
- Understand how to use Google Takeout to acquire data from one or more Google services

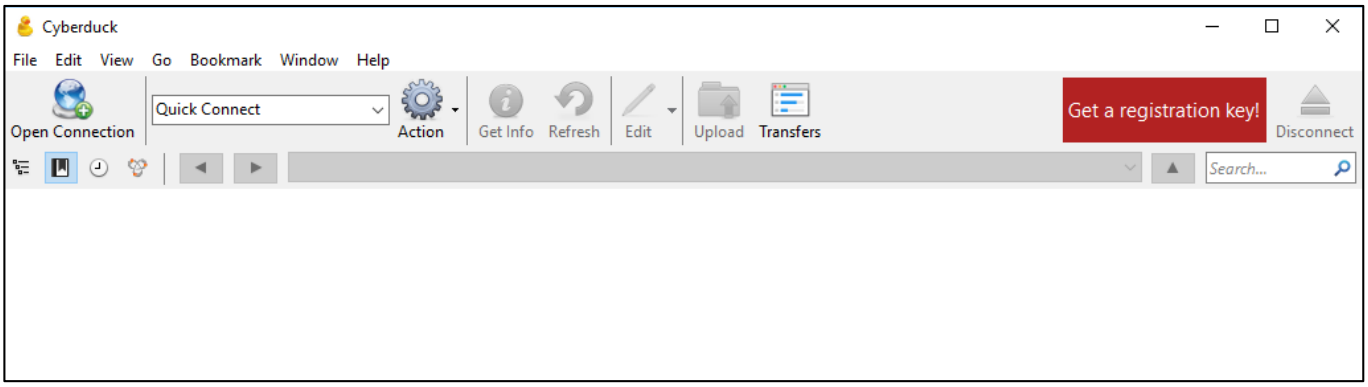
Exercise Preparation

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

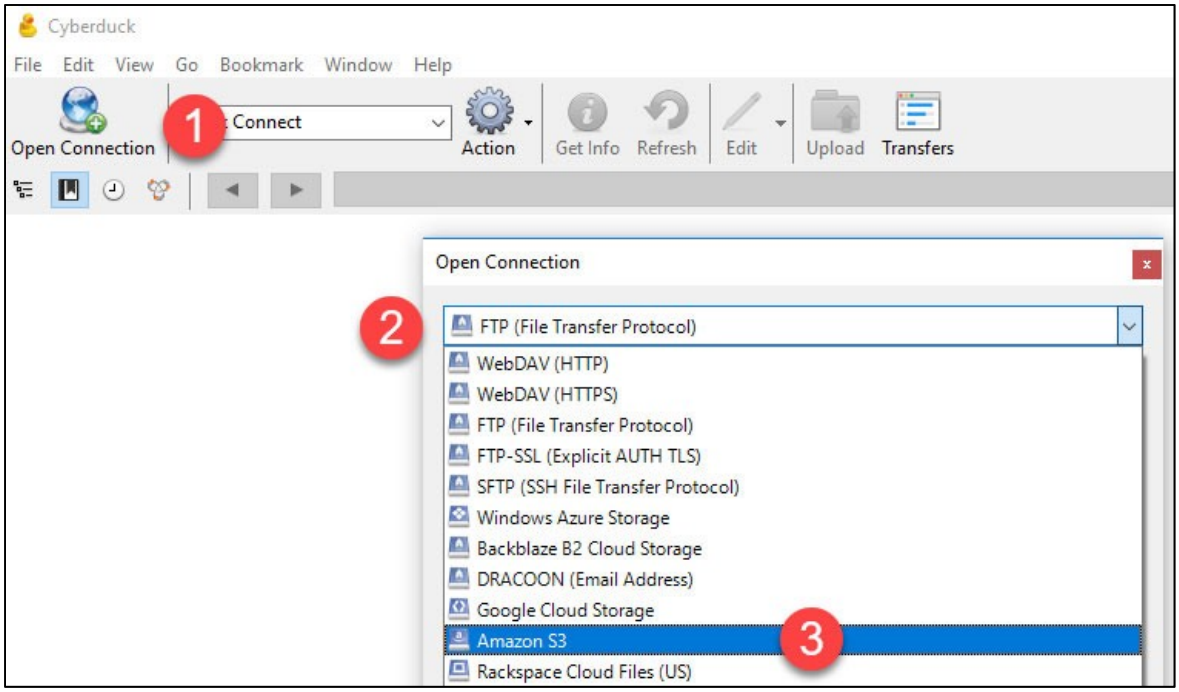
3. Open **File Explorer** and navigate to **C:\Installers**

4. Start the **Cyberduck** application by double clicking the icon in the **Network Tools** fence on your **Desktop**. The main interface is then displayed.



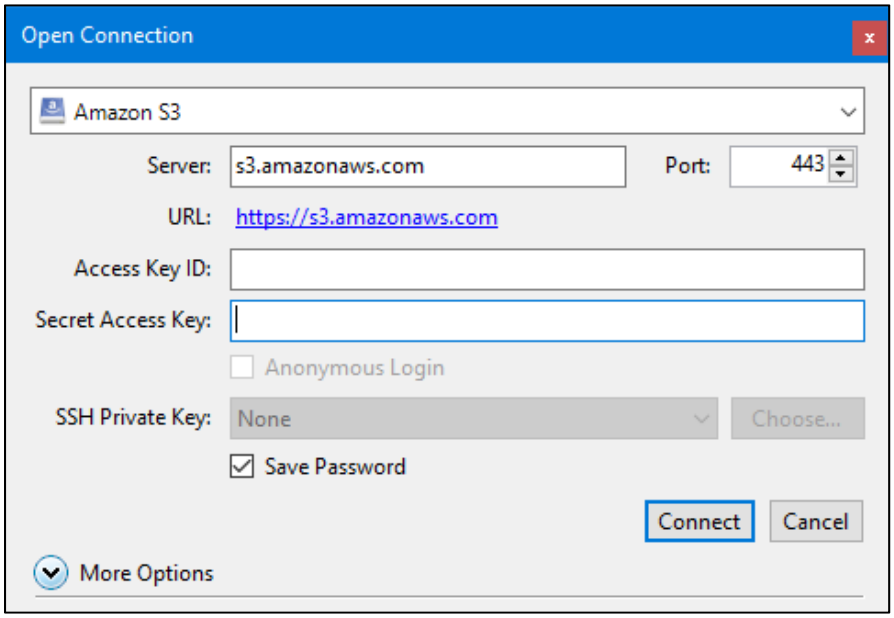
NOTE: If you are prompted to install an update, it is recommended to skip the update.

5. Click the **Open Connection** button in the upper left. In the **Open Connection** dialog, use the drop down to select **Amazon S3** from the list.



6. Configure the connection using the information below.

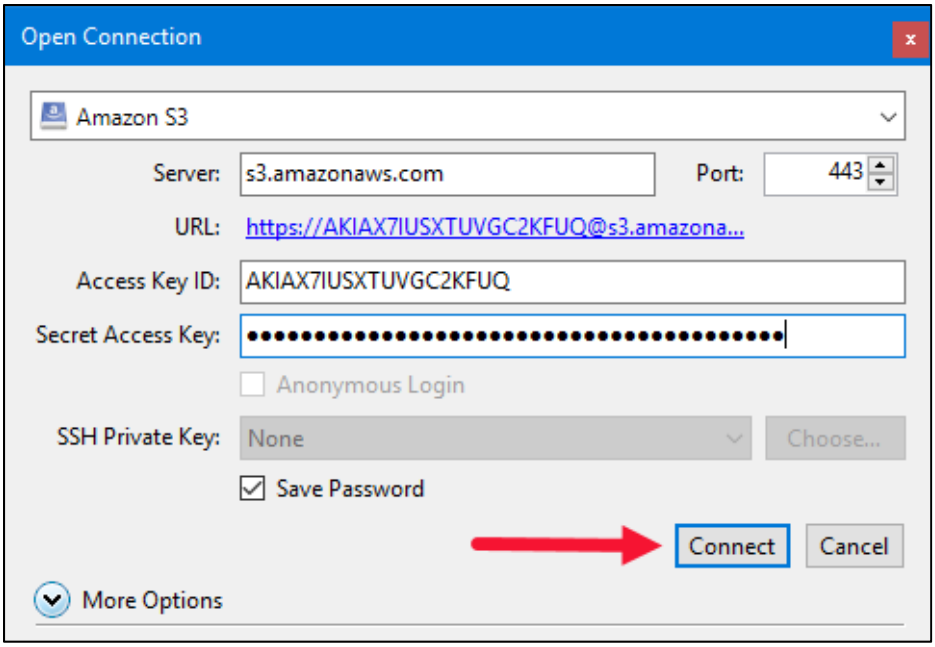
NOTE: **AWS** is very sensitive to time discrepancies. If you get a message related to there being too much time between your machine and **S3**, reboot your **VM** to update your **VM** clock.



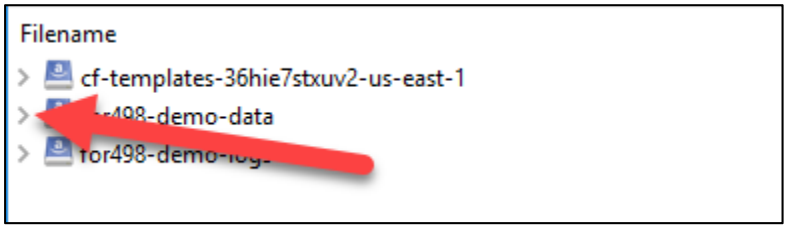
NOTE: These credentials are stored on your **VM** under the **C:\Credentials** directory in a file named **AmazonS3Credentials.txt**. Open this file and **copy/paste** the values unless you really like typing long strings.

Access Key ID	AKIAX7IUSXTUVGC2KFUQ
Secret Access Key	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

7. Click the **Connect** button to connect to **S3**.

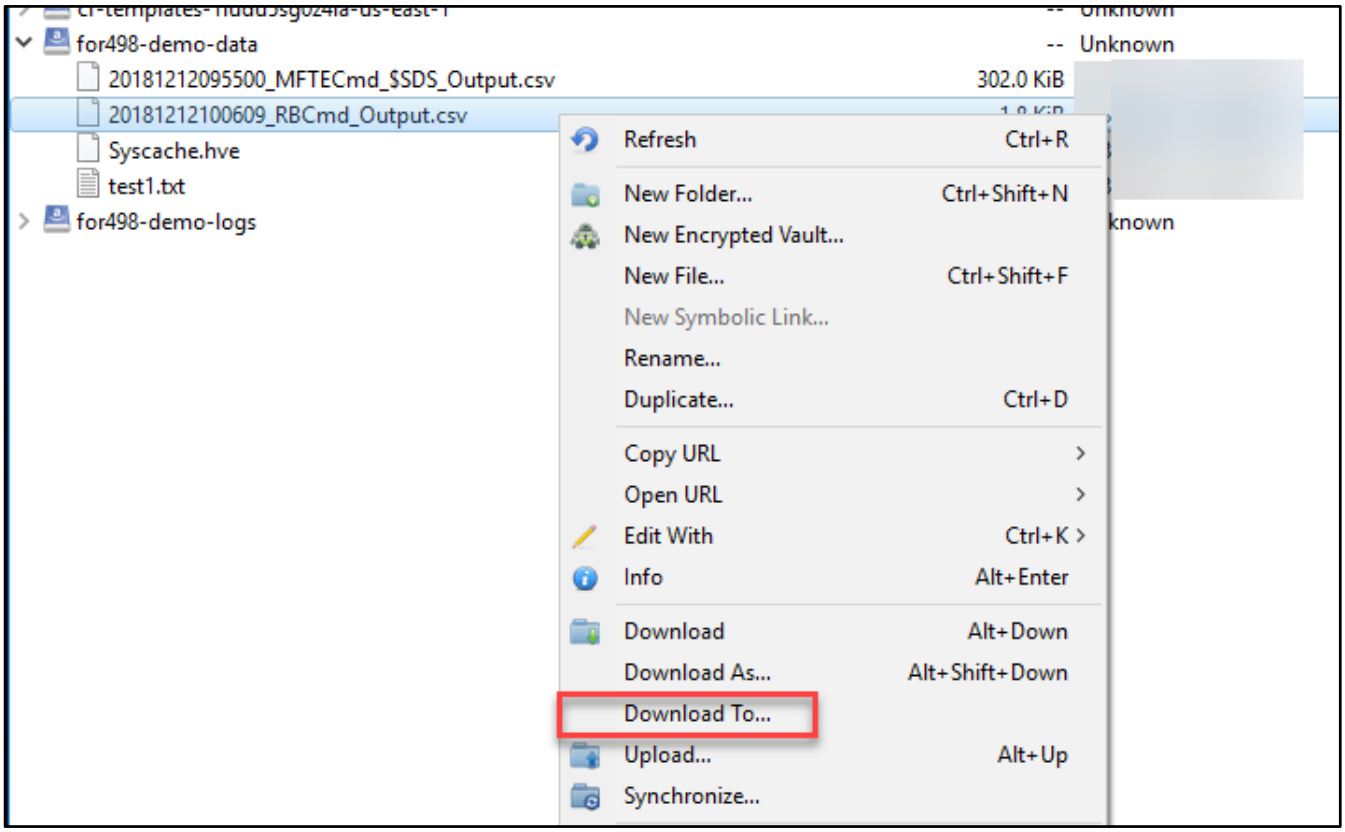


8. If the **Key ID** and **Access Key** are correct, you will see a list of buckets that **Key** has access to. Click the arrow next to the **for498-demo-data** bucket to expand it.



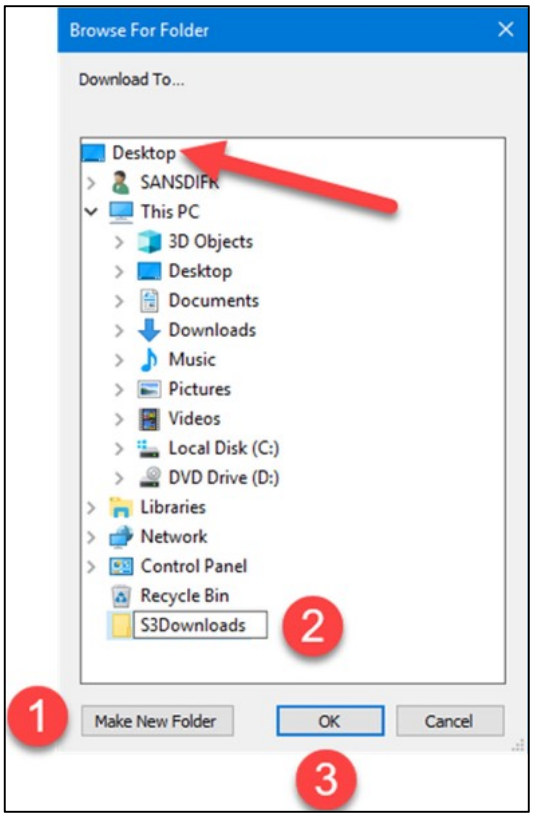
9. The bucket's files and folders are then shown.

10. To download a file, select it by clicking it with the left mouse button, then right-click to bring up a context menu. Select the file named **20181212100609_RBCmd_Output.csv** and right-click on the file to display a context menu.

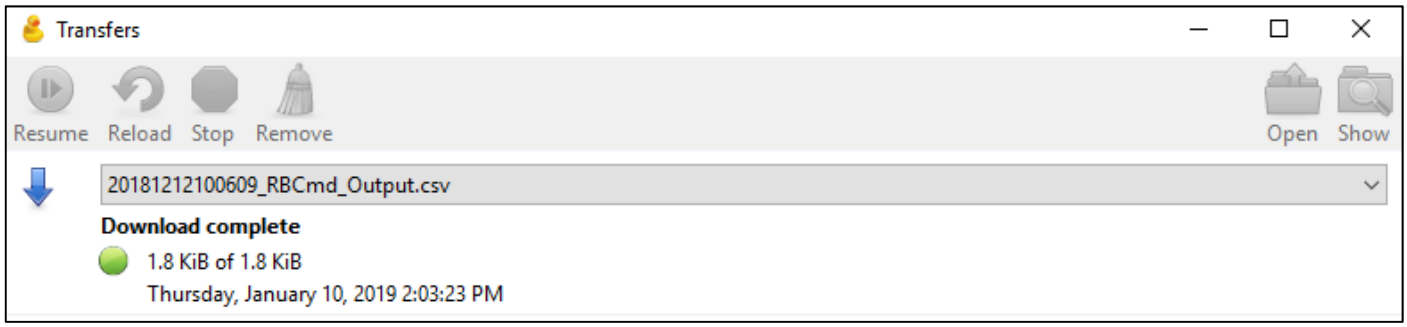


If you wanted to save the file under a different name than what it is stored as on **S3**, use the **Download As** button instead. The **Download** button will download the file using its existing name to your **Downloads** folder under your profile.

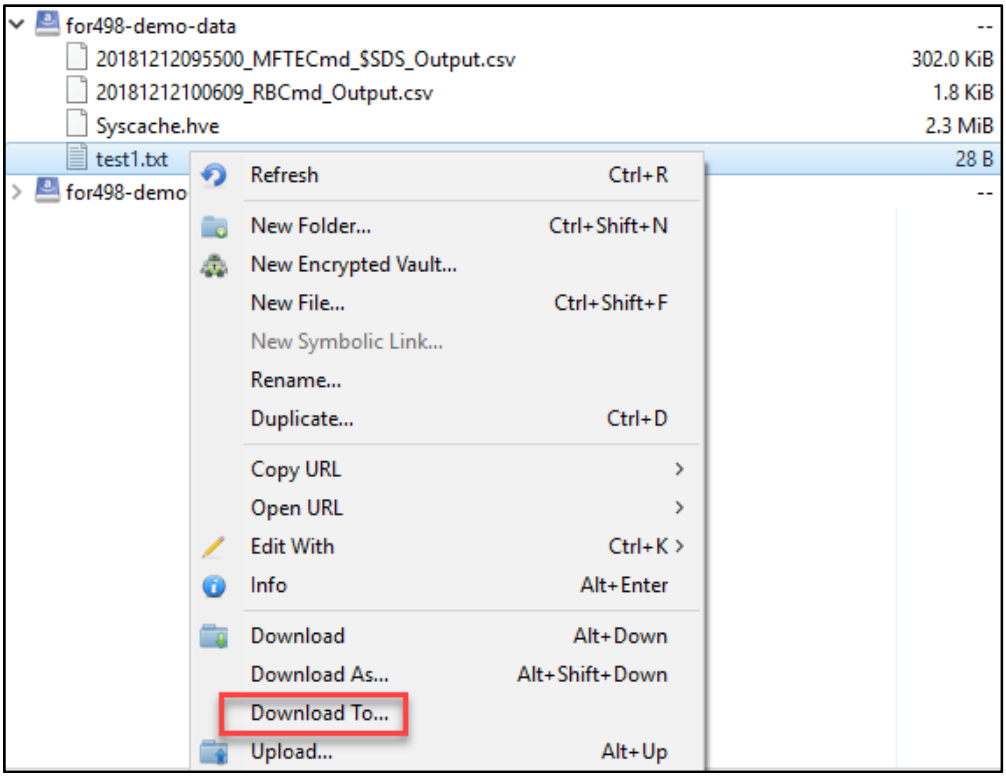
11. Click the **Download To** option, select the **Desktop** top level folder, then click **Make New Folder**. Name the new folder **S3Downloads**, then click **OK** to start the transfer.



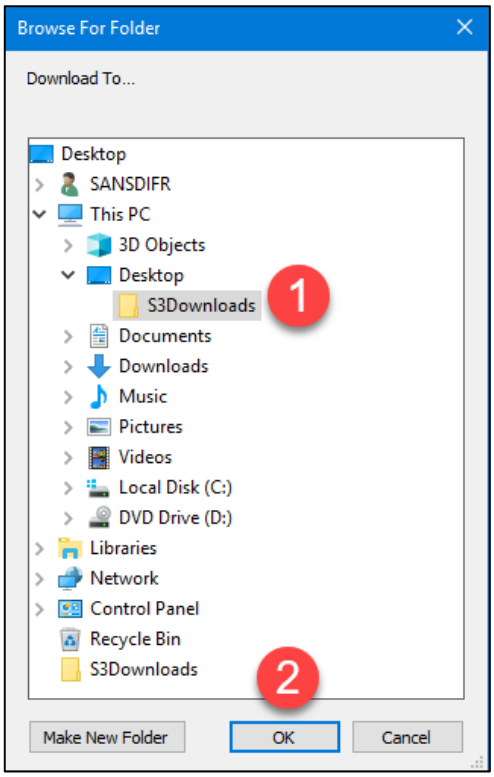
12. **Cyberduck** then begins to transfer the file locally to the specified folder.



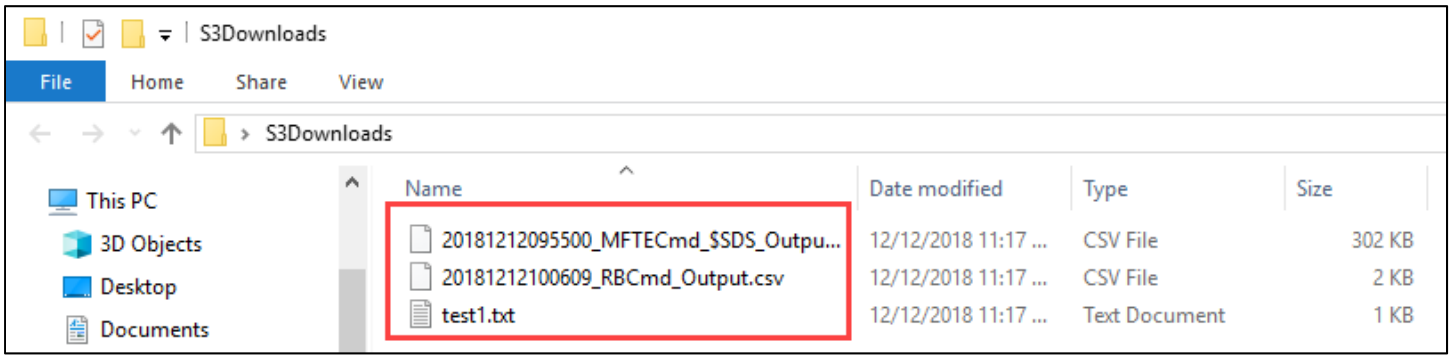
13. Select **test1.txt** and **20181212095500_MFTECmd_$SDS_Output.csv** from the list of files. To select more than one file, hold **CTRL** while clicking. Once the files are selected, right-click and choose **Download To…** (notice the **Download As**… option is disabled since you have more than one file selected).



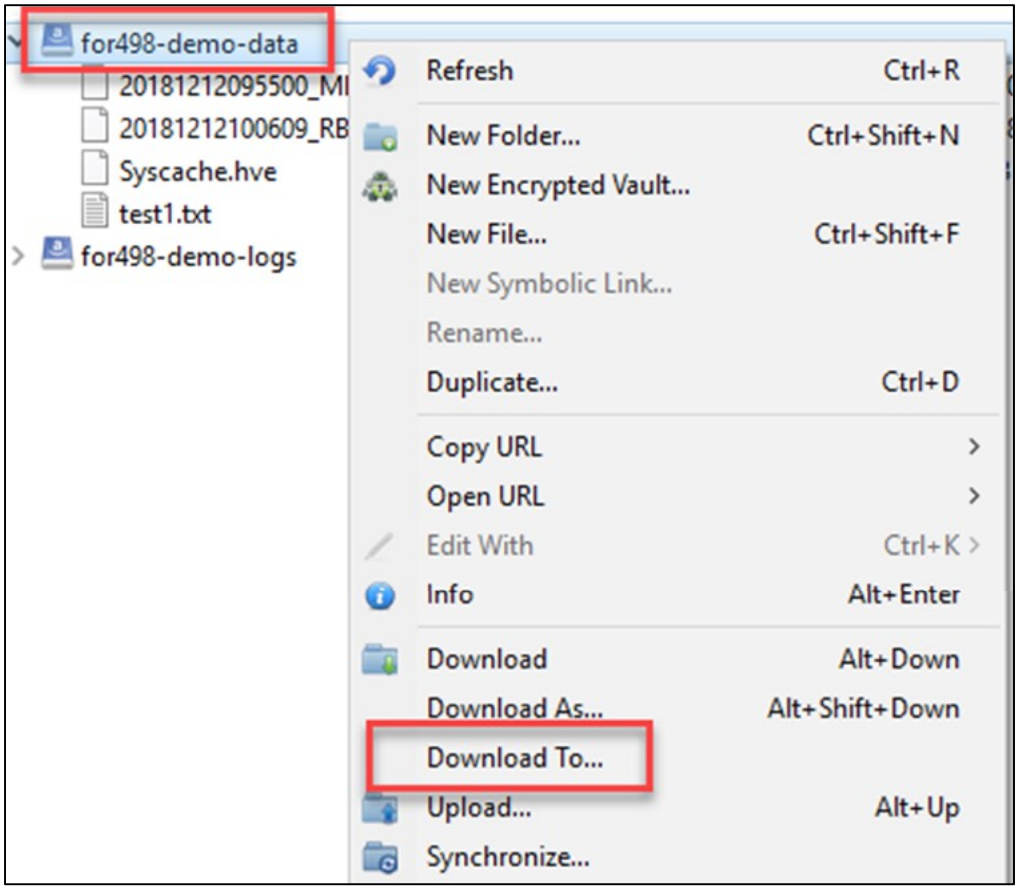
14. Select the same folder as before, the **S3Downloads** folder on your **Desktop**, then click **OK** to start downloading these files.



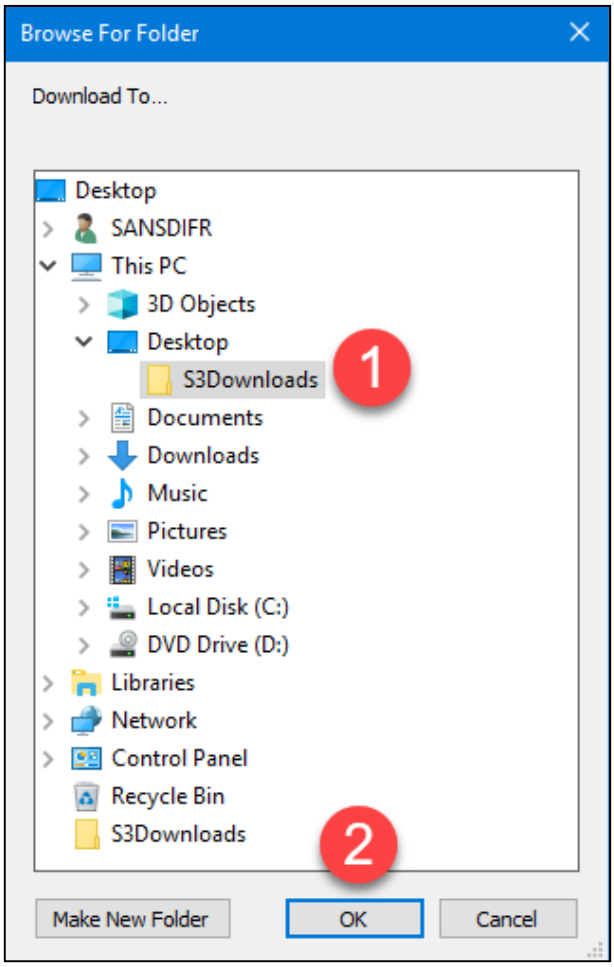
15. Using **File Explorer**, open the **S3Downloads** folder on your **Desktop**. You should see three files.



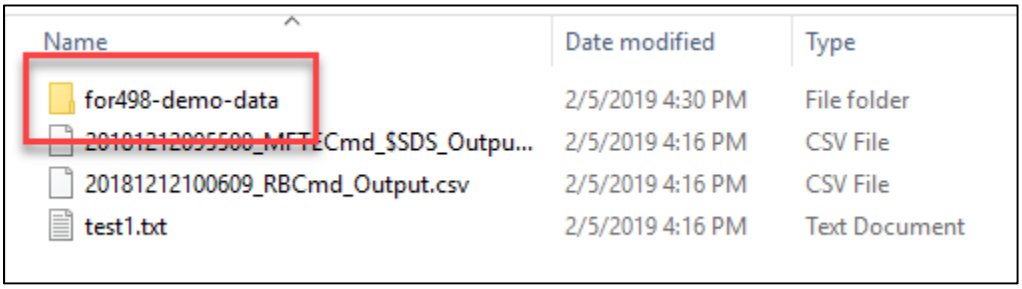
16. **Cyberduck** allows you to download files individually or in groups, but it also allows you to download the entire bucket at once. To download the entire bucket at once, select the bucket name (**for498-demo-data** in the example below), right click, and choose **Download To...** as we did previously.



17. Select the same folder as before, the **S3Downloads** folder on your **Desktop**, then click **OK** to start downloading these files.



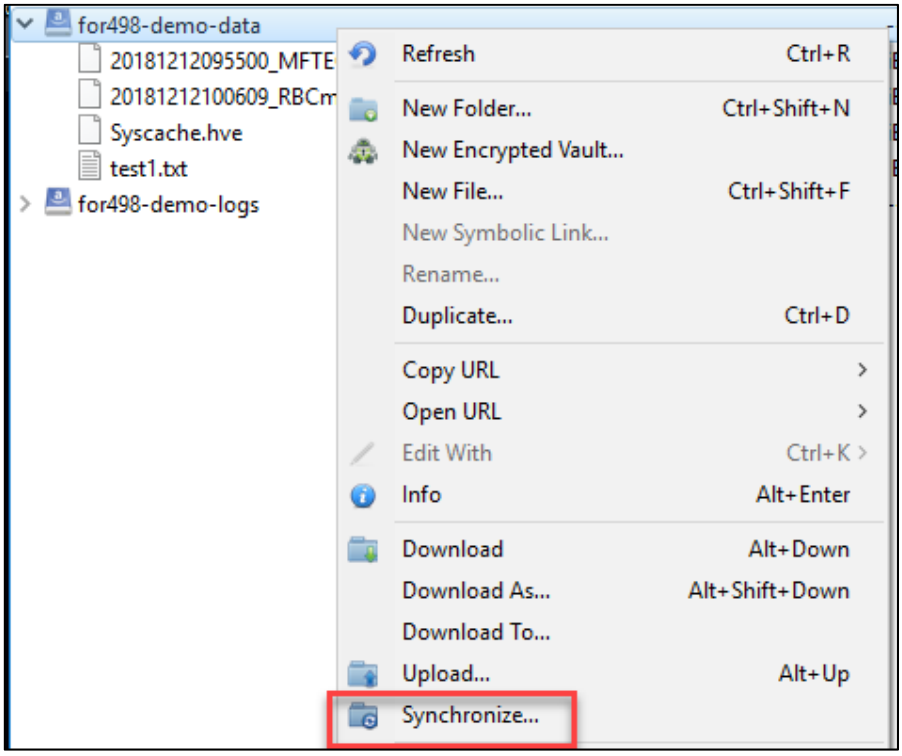
18. **Cyberduck** will then download the entire bucket. When it finishes, open the **S3Downloads** folder using **File Explorer**.



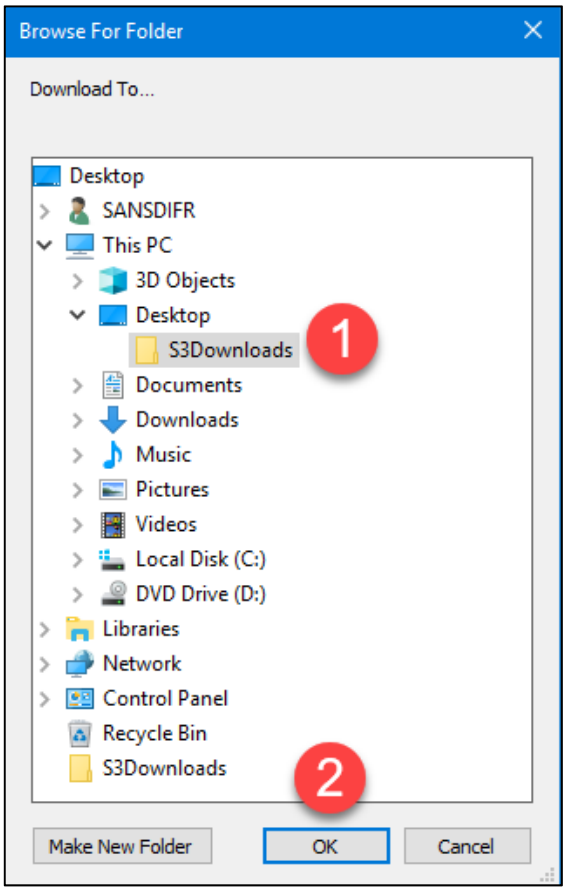
Notice that when downloading an entire bucket, a new directory is created in the download directory named after the bucket being downloaded.

19. In the previous screenshot, notice that we have three files and one directory. Exploring the directory shows we have four files. **Cyberduck** can also synchronize an **S3** bucket with a local directory. This is useful if you already have a set of files (like our three files shown above) and want to get only new files vs. downloading the entire bucket again.

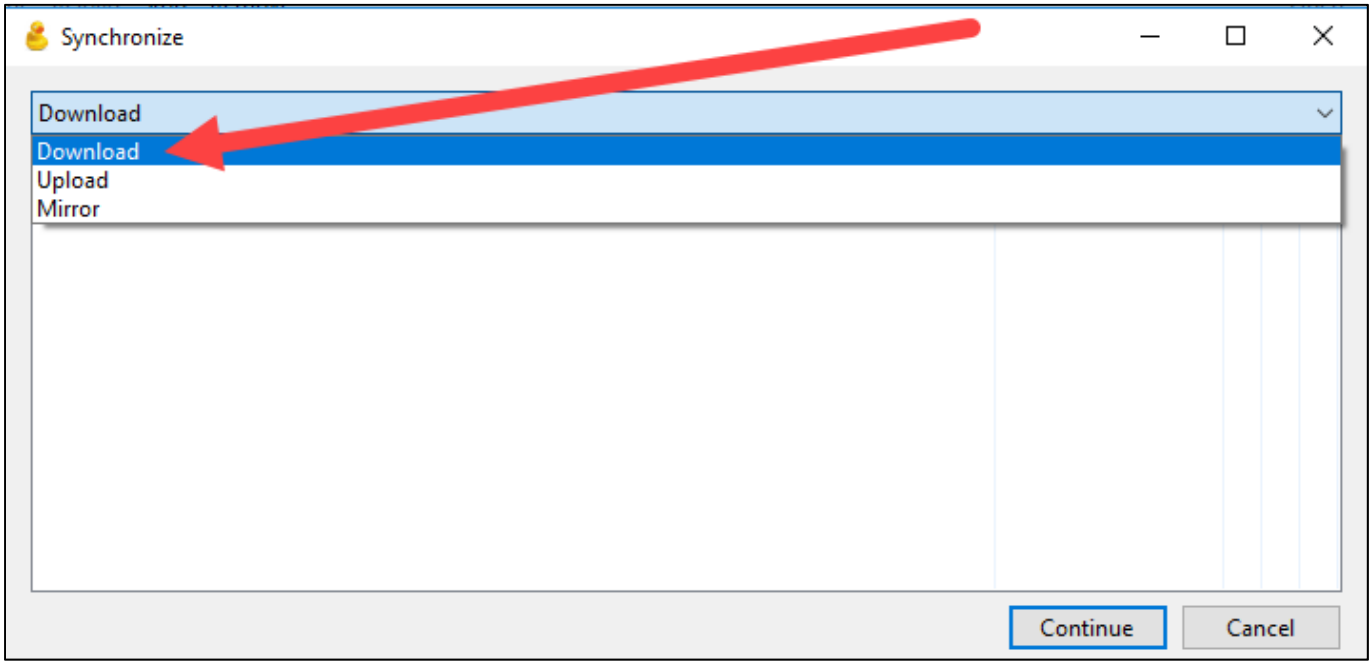
Right-click on the bucket, but this time choose **Synchronize…**



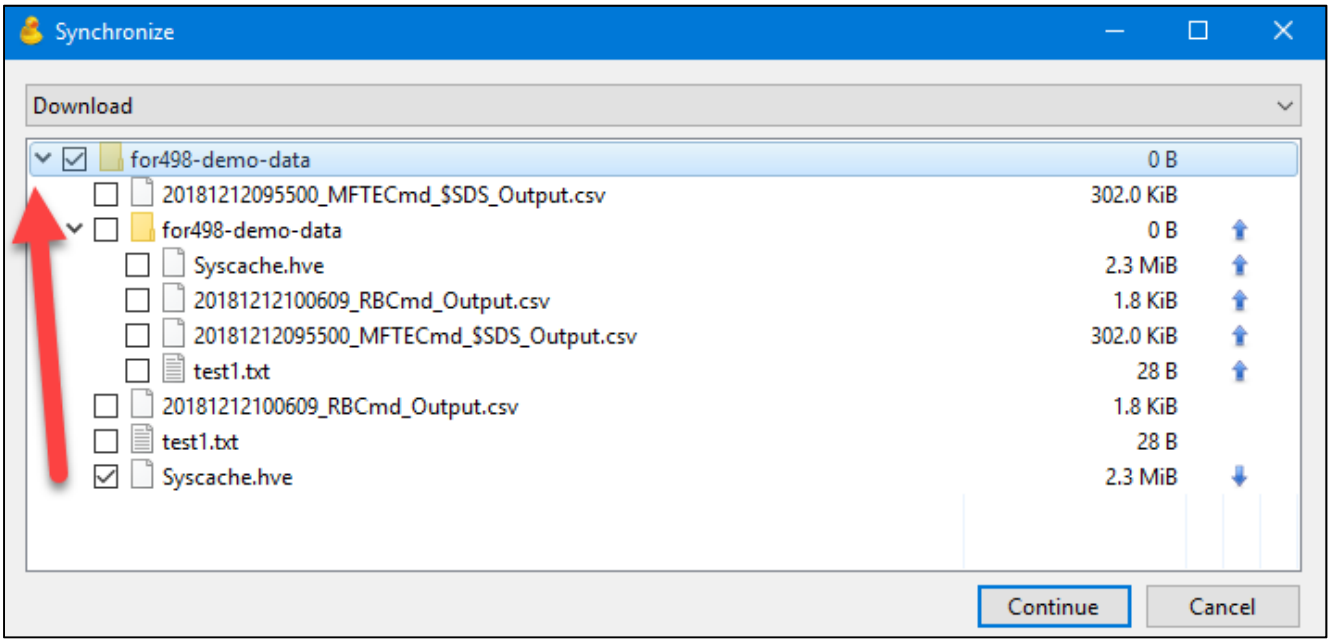
20. Select the same folder as before, the **S3Downloads** folder on your desktop, then click **OK**.



21. In the **Synchronize** dialog, select **Download** from the list.



22. Expand each branch of the tree with the arrows.

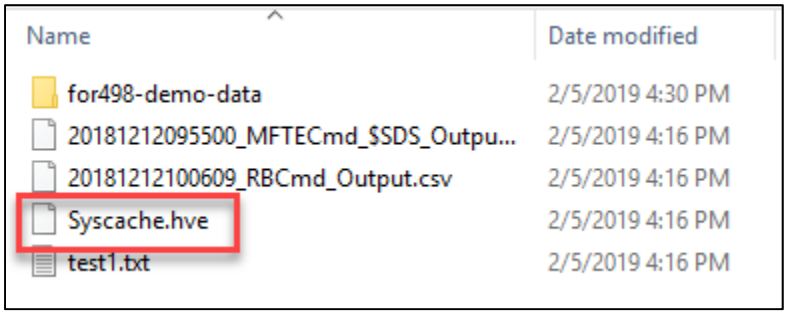


23. Notice what **Cyberduck** is showing here. The blue arrow on the far right shows which files need to be downloaded to get the bucket in sync with what we already have. If we chose **Upload**, or **Mirror**, **Cyberduck** would have checked the boxes next to the files it needs to complete that kind of operation.

Try changing the operation at the top from **Download** to **Mirror** and notice how the selected files are adjusted.

When you are done, make sure the operation is set to **Download**, then click **Continue**.

24. When **Cyberduck** finishes downloading, use **File Explorer** and open **S3Downloads**. Notice how we now have a new file which **Cyberduck** just synchronized.



Cyberduck is a useful tool to visually explore an **S3** bucket to see what is available. It does a great job of hiding the complexities of the **S3 API**, but if you need more fine-grained control, the **AWS CLI** is also very useful. Also, depending on how many files you need to transfer, you may find **Cyberduck** is slower than the **AWS CLI**.

With that said, let's look at how we can interact with **S3** using a command line interface next.

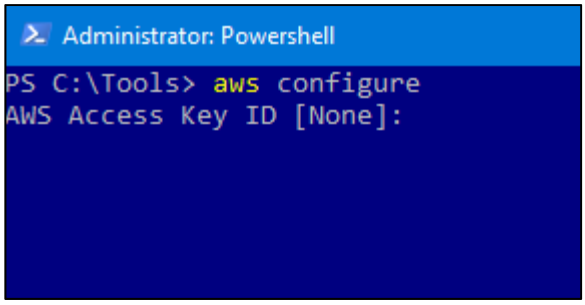
25. Open **File Explorer** and navigate to **C:\Installers**

26. Install the **AWS CLI** by double clicking **AWSCLIx64.msi**. When prompted, agree to the terms and use the defaults for everything else. Click **Finish** when the installer completes.

27. Using **File Explorer**, create a new folder named **C:\Temp\S3CLI**

28. Open a **PowerShell** prompt using the **Desktop** shortcut and type the following and press **Enter** to begin the configuration process. Use the information in the table below when prompted.

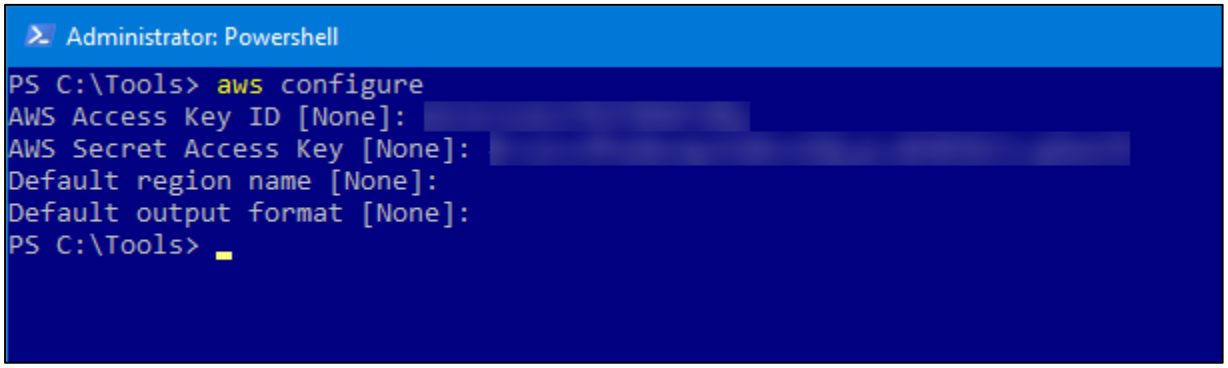
```
aws configure
```



NOTE: These credentials are stored on your **VM** under the **C:\Credentials** directory in a file named **AmazonS3Credentials.txt**. Open this file and **copy/paste** the values unless you really like typing long strings.

Access Key ID	AKIAX7IUSXTUVGC2KFUQ
Secret Access Key	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

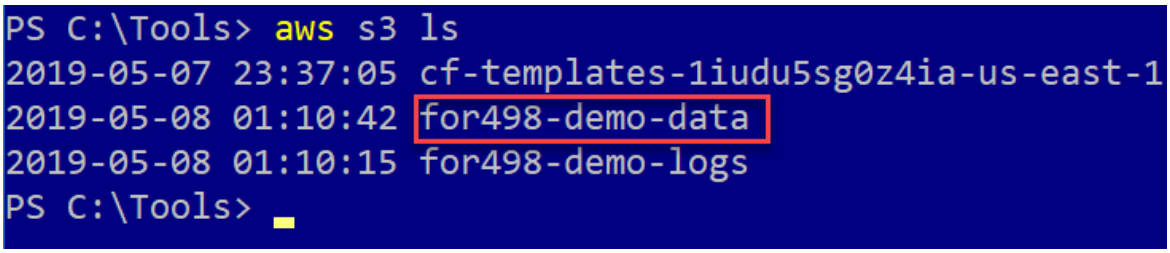
When prompted for **Default region name** and Default **output format**, press **Enter** to skip.



Exercise Questions

1. With the credentials supplied, we can now interact with any **S3** buckets we have access to. To get a listing of available **S3** buckets, we need to use the **ls** command. Type the following and press **Enter** in the **PowerShell** window.

```
aws s3 ls
```



This displays a listing of all the **S3** buckets that can be accessed.

a. When was the **for498-demo-data** bucket created?

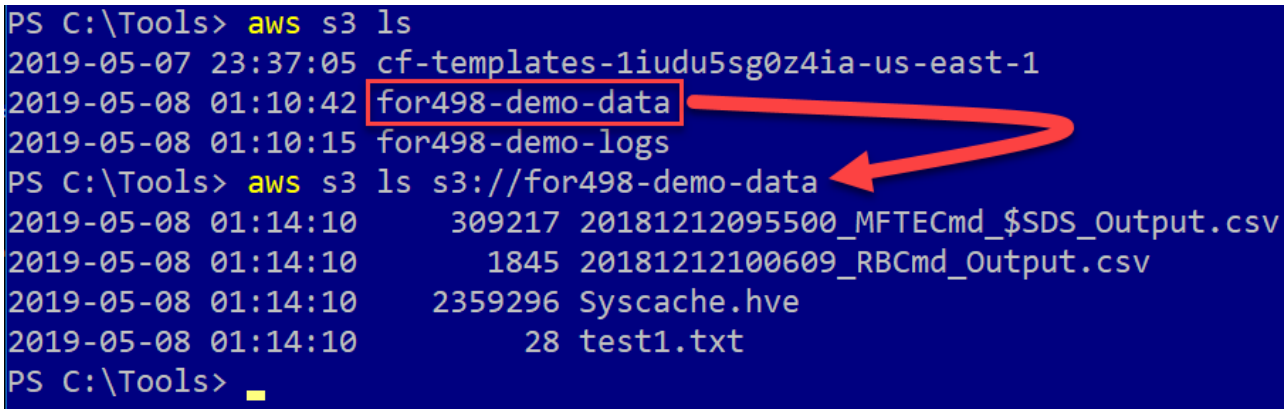
_____

NOTE: The timestamp returned is in *local* time based on your machine's current settings!

2. To see what is inside a bucket, we need to change the format of our command slightly by adding the name of the bucket to the command. The general format for the command is:

   aws s3 ls s3://<bucketname>

   where <**bucketname**> is one of the buckets we have access to. In the example below, we are listing the contents of the **for498-demo-data** bucket.



   To see this for yourself, enter the following command:

```
aws s3 ls s3://for498-demo-data
```

   As can be seen above, the command lists all the files in the bucket, very similar to how **Cyberduck** showed us.

   a. What is the size of the largest file in the bucket?

      _____

   b. What is the total size of all files in the bucket *in bytes*?

      _____

      HINT: While you can certainly just add the sizes up manually, we can automate this with the following command:

```
aws s3 ls s3://for498-demo-data --summarize
```

      To make it even easier, try this command:

```
aws s3 ls s3://for498-demo-data --summarize --human-readable
```

c. How many *megabytes* (Listed in Mebibits in the output) in size do all the files consume?

_____

3. To download a file, we need to use a different command than we used to list the bucket. The **cp** command is used for copying files, so enter the following command to download **test1.txt** from **S3** to the directory we created earlier.

```
aws s3 cp s3://for498-demo-data/test1.txt C:\Temp\S3CLI
```

```
PS C:\Tools> aws s3 cp s3://for498-demo-data/test1.txt C:\Temp\S3CLI
download: s3://for498-demo-data/test1.txt to ..\Temp\S3CLI\test1.txt
PS C:\Tools>
```

4. In many cases you may only be interested in downloading certain files from an **S3** bucket. Because of the way the **AWS CLI** works, we must add a few additional options as the example below shows. Run the following command:

```
aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive
--exclude "*" --include "*.csv"
```

This command is still downloading files from the same bucket and placing them in the same destination folder, but in order to filter by a wild card, we must use the following additional switches:

Switch	Purpose
--recursive	Search for files recursively in the bucket
--exclude "*"	Removes all files from being included
--include "*.csv"	Include only the supplied mask for downloading

Note that the order matters too, so when using this technique, supply the **`--recursive`**, **`--exclude`**, and **`--include`** options in that order.

The image below shows you an example of what this will look like.

```
PS C:\Tools> aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive --exclude "*"
--include "*.csv"
download: s3://for498-demo-data/20181212100609_RBCmd_Output.csv to ..\Temp\S3CLI\20181
212100609_RBCmd_Output.csv
download: s3://for498-demo-data/20181212095500_MFTECmd_$SDS_Output.csv to ..\Temp\S3CL
I\20181212095500_MFTECmd_$SDS_Output.csv
```

5. Open **File Explorer** and navigate to **C:\Temp\S3CLI**.

   a. How many CSV files exist?

   _____

   b. What happens if you repeat the above command (press the up-arrow key to bring up the last executed command)?

   _____

   _____

6. To download an entire bucket, we can simply drop the **`--exclude`** and **`--include`** switches. First, delete all the files in **C:\Temp\S3CLI** using **File Explorer**.

7. Run the following command to download the entire bucket:

```
aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive
```

```
PS C:\Tools> aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive
download: s3://for498-demo-data/test1.txt to ..\Temp\S3CLI\test1.txt
download: s3://for498-demo-data/20181212100609_RBCmd_Output.csv to ..\Temp\S3CLI\20181
212100609_RBCmd_Output.csv
download: s3://for498-demo-data/20181212095500_MFTECmd_$SDS_Output.csv to ..\Temp\S3CL
I\20181212095500_MFTECmd_$SDS_Output.csv
download: s3://for498-demo-data/Syscache.hve to ..\Temp\S3CLI\Syscache.hve
```

   a. How many files were successfully transferred?

   _____

   b. Did you find this easier, or more difficult, than **Cyberduck**? Why or why not?

   _____

   _____

   _____

While there are a large number of additional capabilities available in the **AWS CLI**, we are primarily interested in downloading files from **S3**. Should you need any additional functionality, such as uploading, review the **AWS** manual for the **S3** service via the "**`aws s3 help`**" command.

BONUS: Using "**`aws s3 help`**", craft the command necessary to sync the **S3** bucket to a local folder. Before executing the command, delete all files in **C:\Temp\S3CLI** so you can be sure the sync command worked.

OPTIONAL Out of Class— Using Google Takeout

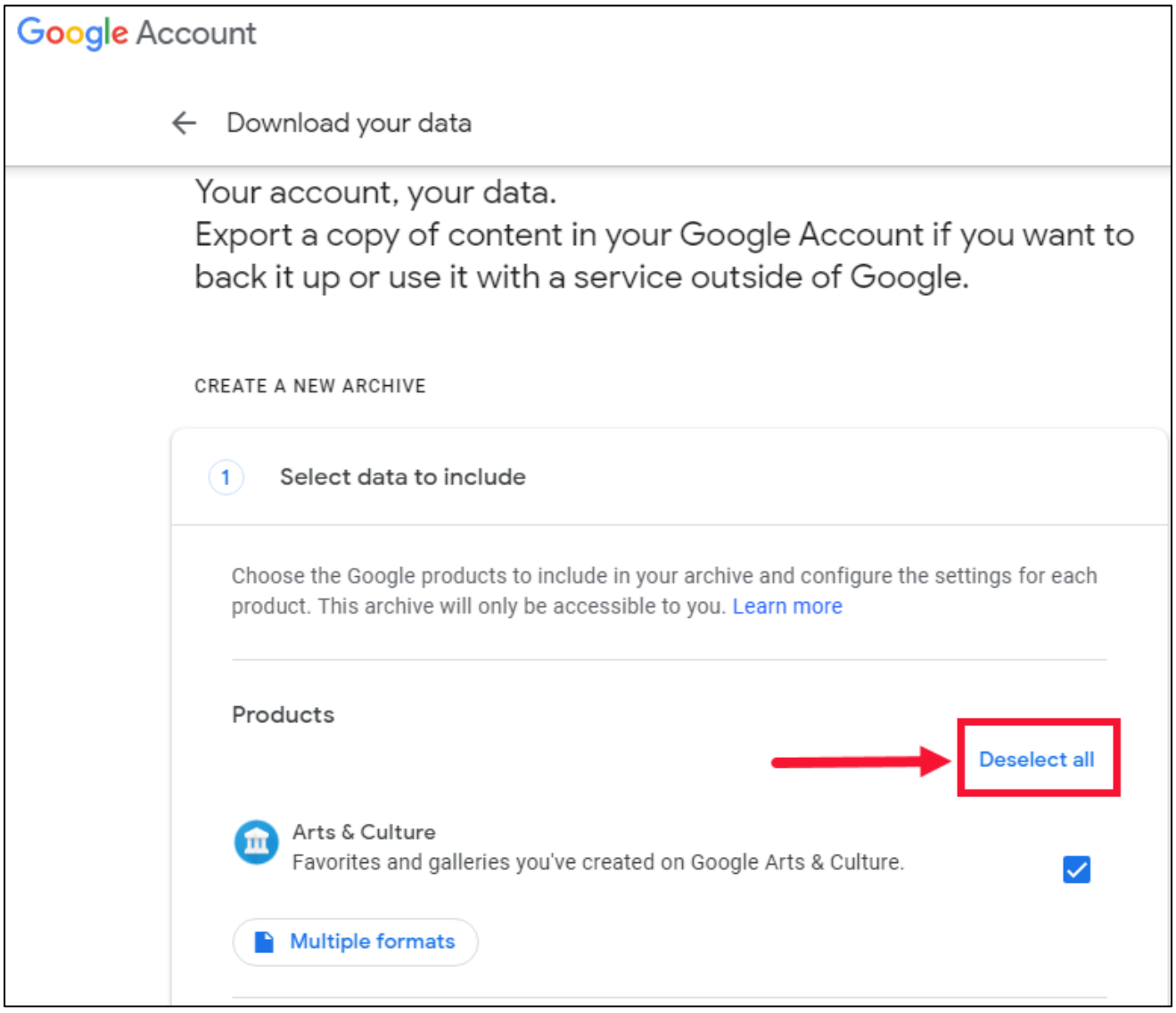
This option, being Google specific, also requires a Google account. If you have a gmail.com account, this will work fine.

NOTE: The screen shots and interface to use Google Takeout may not match 100% against what is shown below as this feature is updated by Google from time to time.

1. Open a browser and visit **https://takeout.google.com/settings/takeout**

2. Sign in using your account credentials.

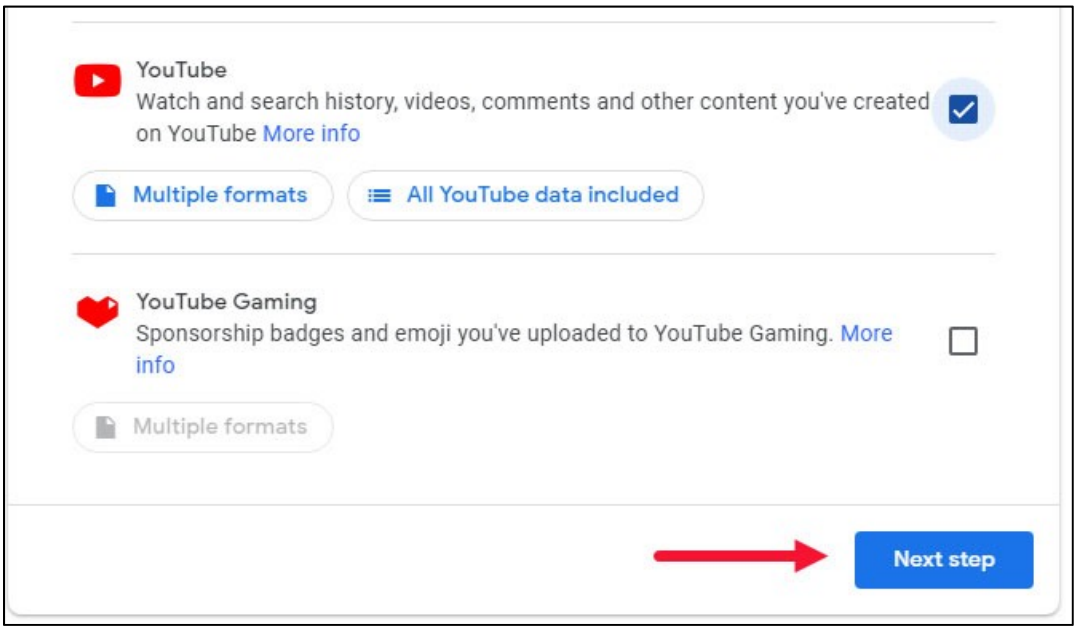
3. By default, everything is included, but to give you a feel for what is there, click the **SELECT NONE** button.



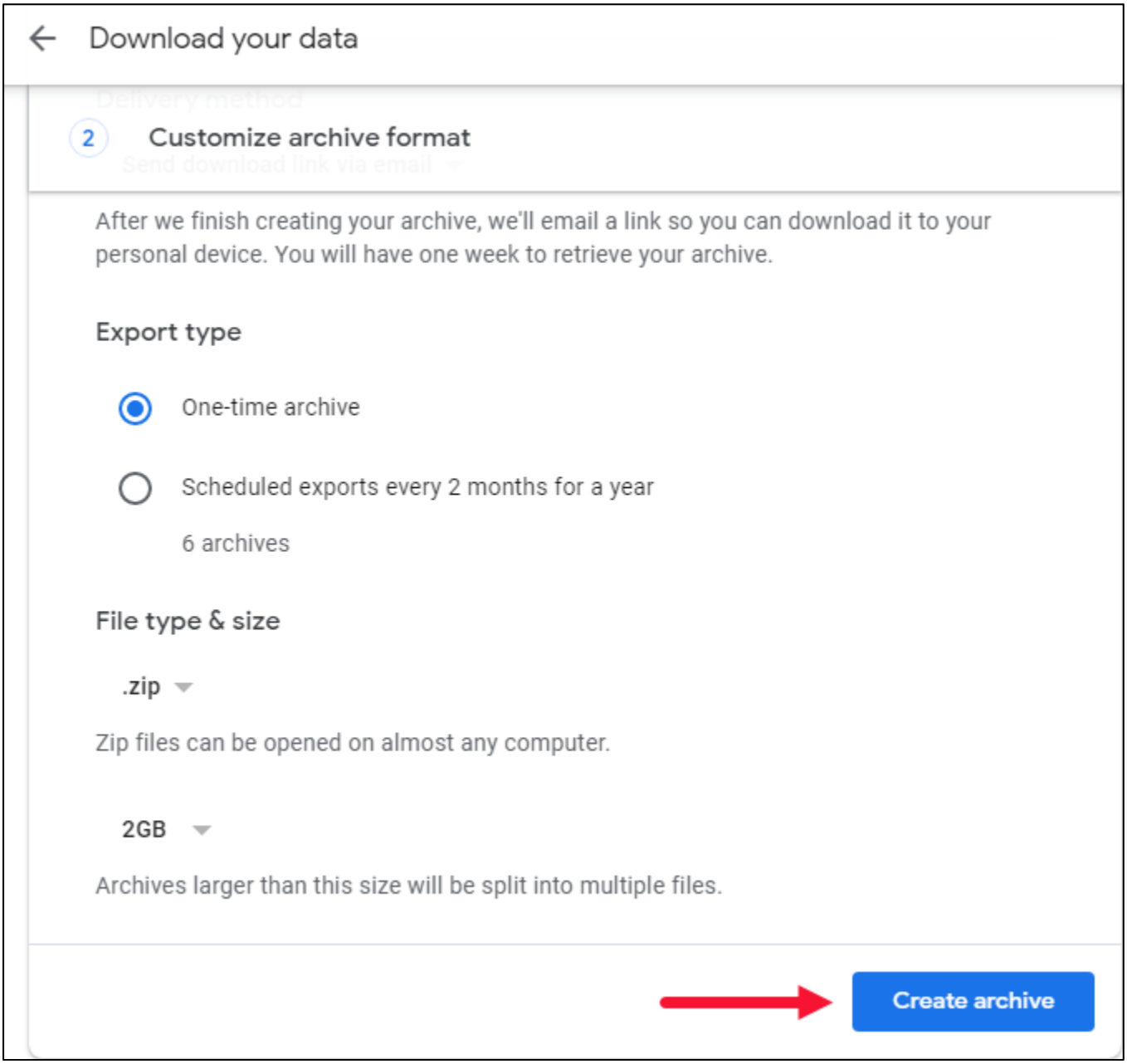
4. Scroll down and review the various Google services. Select a few, like **Mail**. Notice there is a drop down that lets you choose from all mail, or only selected labels.

The contents will be provided in **MBOX** format.

5. When done selecting services, click the **Next step** button.



6. You can optionally adjust things like file type, archive size, etc. but the defaults are generally OK. Click **Create archive** to begin the process.

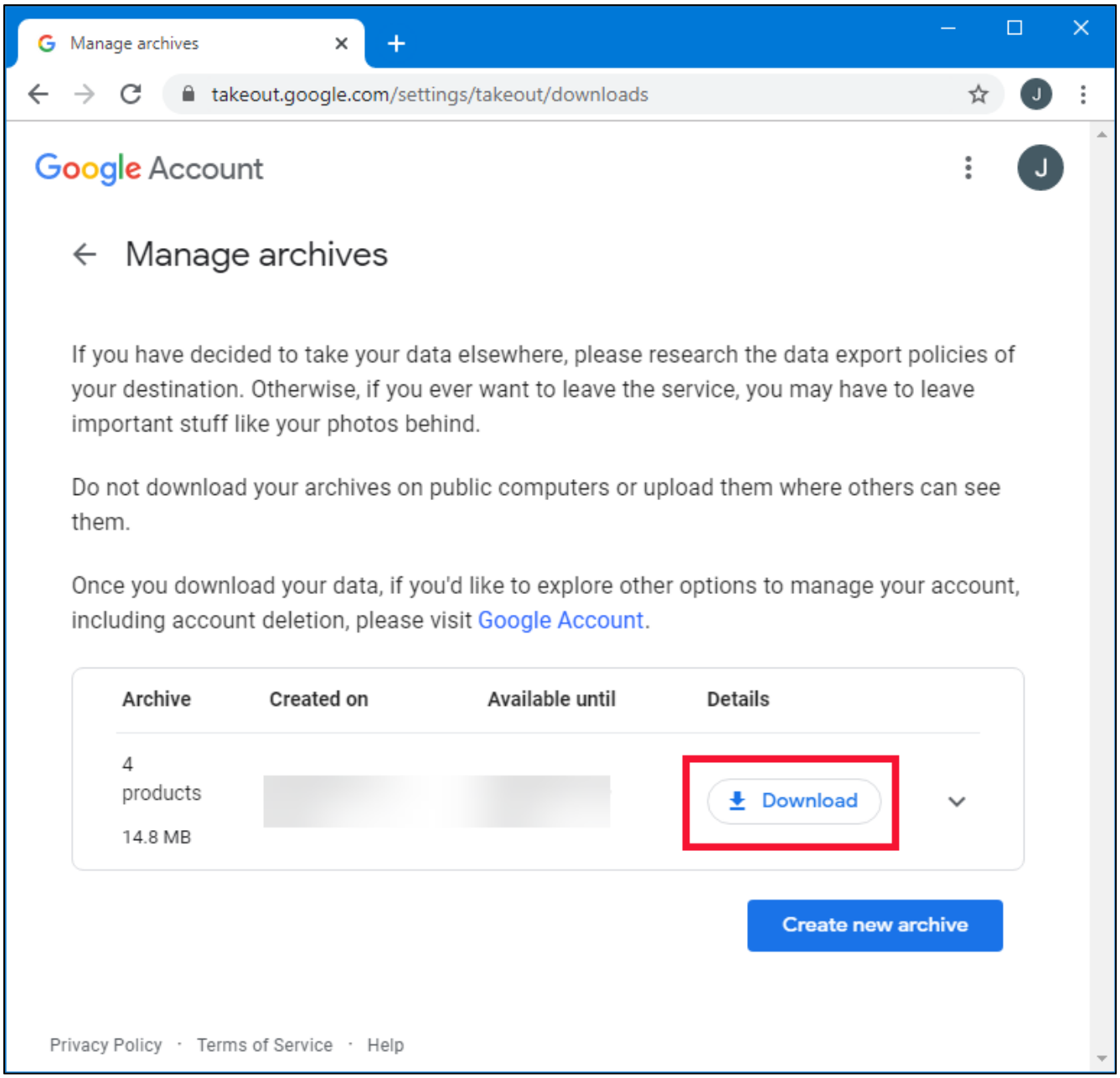


7. Wait for the confirmation email saying the data is available.

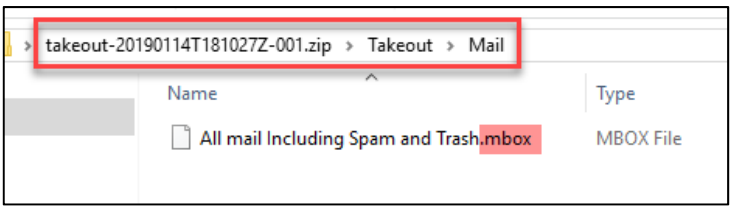
8. This may take a while…

9. Once you are emailed that the data is ready, visit **https://takeout.google.com/settings/takeout/downloads**

10. Click the **Download** link next to the archive you want to download



11. If prompted, re-authenticate with your password.
12. Once the archive is downloaded, open it with **7-Zip**, then drill down into the **Takeout\Mail** directory.



13. To view the contents of the email, for example, extract the **mbox** files and view in your email viewer of choice.

1. With the credentials supplied, we can now interact with any **S3** buckets we have access to. To get a listing of available **S3** buckets, we need to use the **ls** command. Type the following and press **Enter** in the **PowerShell** window.

```
aws s3 ls
```

```
PS C:\Tools> aws s3 ls
2019-05-07 23:37:05 cf-templates-1iudu5sg0z4ia-us-east-1
2019-05-08 01:10:42 for498-demo-data
2019-05-08 01:10:15 for498-demo-logs
PS C:\Tools> _
```

This displays a listing of all the **S3** buckets that can be accessed.

a. When was the **for498-demo-data** bucket created?

2019-05-08 01:10:42

NOTE: The timestamp returned is in *local* time based on your machine's current settings!

2. To see what is inside a bucket, we need to change the format of our command slightly by adding the name of the bucket to the command. The general format for the command is:

aws s3 ls s3://<bucketname>

where <**bucketname**> is one of the buckets we have access to. In the example below, we are listing the contents of the **for498-demo-data** bucket.

```
PS C:\Tools> aws s3 ls
2019-05-07 23:37:05 cf-templates-1iudu5sg0z4ia-us-east-1
2019-05-08 01:10:42 for498-demo-data
2019-05-08 01:10:15 for498-demo-logs
PS C:\Tools> aws s3 ls s3://for498-demo-data
2019-05-08 01:14:10     309217 20181212095500_MFTECmd_$SDS_Output.csv
2019-05-08 01:14:10       1845 20181212100609_RBCmd_Output.csv
2019-05-08 01:14:10    2359296 Syscache.hve
2019-05-08 01:14:10         28 test1.txt
PS C:\Tools> _
```

To see this for yourself, enter the following command:

```
aws s3 ls s3://for498-demo-data
```

As can be seen above, the command lists all the files in the bucket, very similar to how **Cyberduck** showed us.

a. What is the size of the largest file in the bucket?

2359296 bytes

b. What is the total size of all files in the bucket *in bytes*?

2670386 bytes

HINT: While you can certainly just add the sizes up manually, we can automate this with the following command:

```
aws s3 ls s3://for498-demo-data --summarize
```

To make it even easier, try this command:

```
aws s3 ls s3://for498-demo-data --summarize --human-readable
```

c. How many *megabytes* (Listed in Mebibits in the output) in size do all the files consume?

2.5 MiB

3. To download a file, we need to use a different command than we used to list the bucket. The **cp** command is used for copying files, so enter the following command to download **test1.txt** from **S3** to the directory we created earlier.

```
aws s3 cp s3://for498-demo-data/test1.txt C:\Temp\S3CLI
```

```
PS C:\Tools> aws s3 cp s3://for498-demo-data/test1.txt C:\Temp\S3CLI
download: s3://for498-demo-data/test1.txt to ..\Temp\S3CLI\test1.txt
PS C:\Tools>
```

4. In many cases you may only be interested in downloading certain files from an **S3** bucket. Because of the way the **AWS CLI** works, we must add a few additional options as the example below shows. Run the following command:

```
aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive
--exclude "*" --include "*.csv"
```

This command is still downloading files from the same bucket and placing them in the same destination folder, but in order to filter by a wild card, we must use the following additional switches:

Switch	Purpose
--recursive	Search for files recursively in the bucket
--exclude "*"	Removes all files from being included
--include "*.csv"	Include only the supplied mask for downloading

Note that the order matters too, so when using this technique, supply the **`--recursive`**, **`--exclude`**, and **`--include`** options in that order.

The image below shows you an example of what this will look like.

```
PS C:\Tools> aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive --exclude "*"
--include "*.csv"
download: s3://for498-demo-data/20181212100609_RBCmd_Output.csv to ..\Temp\S3CLI\20181
212100609_RBCmd_Output.csv
download: s3://for498-demo-data/20181212095500_MFTECmd_$SDS_Output.csv to ..\Temp\S3CL
I\20181212095500_MFTECmd_$SDS_Output.csv
```

5. Open File Explorer and navigate to **C:\Temp\S3CLI**.

   a. How many CSV files exist?

   Two CSV files exist.

   b. What happens if you repeat the above command (press the up-arrow key to bring up the last executed command)?

   Files with identical names are overwritten.

6. To download an entire bucket, we can simply drop the **`--exclude`** and **`--include`** switches. First, delete all the files in **C:\Temp\S3CLI** using **File Explorer**.

7. Run the following command to download the entire bucket:

```
aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive
```

```
PS C:\Tools> aws s3 cp s3://for498-demo-data/ C:\Temp\S3CLI --recursive
download: s3://for498-demo-data/test1.txt to ..\Temp\S3CLI\test1.txt
download: s3://for498-demo-data/20181212100609_RBCmd_Output.csv to ..\Temp\S3CLI\20181
212100609_RBCmd_Output.csv
download: s3://for498-demo-data/20181212095500_MFTECmd_$SDS_Output.csv to ..\Temp\S3CL
I\20181212095500_MFTECmd_$SDS_Output.csv
download: s3://for498-demo-data/Syscache.hve to ..\Temp\S3CLI\Syscache.hve
```

   a. How many files were successfully transferred?

      Four files were transferred.

   b. Did you find this easier, or more difficult, than **Cyberduck**? Why or why not?

      While this comes down to personal preference, in our experience, using the AWS CLI is easier once you know the commands you need to use. It is also scriptable and much faster to download files, especially when you have a large amount of data to pull from S3.

While there are a large number of additional capabilities available in the **AWS CLI**, we are primarily interested in downloading files from **S3**. Should you need any additional functionality, such as uploading, review the **AWS** manual for the **S3** service via the "**`aws s3 help`**" command.

BONUS: Using "**`aws s3 help`**", craft the command necessary to sync the **S3** bucket to a local folder. Before executing the command, delete all files in **C:\Temp\S3CLI** so you can be sure the sync command worked.

The `aws s3 sync help` command shows the syntax for the sync command. Reviewing the options available shows that the following command will sync the remote bucket to the local directory:

```
aws s3 sync s3://for498-demo-data/ C:\Temp\S3CLI
```

Any files in the bucket will be downloaded if the size of the remote file is different from the local one, the file does not exist at all, or the last modified time of the file on the bucket is newer than the local copy. Because of this, simply repeating this command as often as needed will keep things synchronized.

Exercise—Key Takeaways

- Network acquisition allows for using a wide range of tools to collect data.
- Be flexible! If one tool does not work for you, pivot to another.
- Be sure to take into consideration the bandwidth available to you when performing acquisition over the network. If you are on a slow link, consider collecting only the most critical data you need first.
- For collections that you may do more than once, consider using a command line interface program which lets you automate via a script. This can save considerable time (as well as prevent mistakes!) over other access methods.

Optional - Out of Class
Exercise 4.3B—Network Acquisition Thunderbird & F-Response

Background

Network collection is a methodology that is becoming more and more necessary as our world becomes more interconnected, with data residing in far more places than on a hard drive in a desktop computer. From hosted email, to web sites, to cloud storage, the range of situations that involve network acquisition is a mile wide and deep.

This lab looks at how to collect IMAP email using Thunderbird portable and F-Response. The F-Response section also covers collecting from other devices such as computers as well as cloud storage.

Objectives

- Use Thunderbird portable to collect IMAP data
- Use F-Response to target and collect network-based assets

Exercise Preparation

1. Boot your **FOR498 Windows VM**

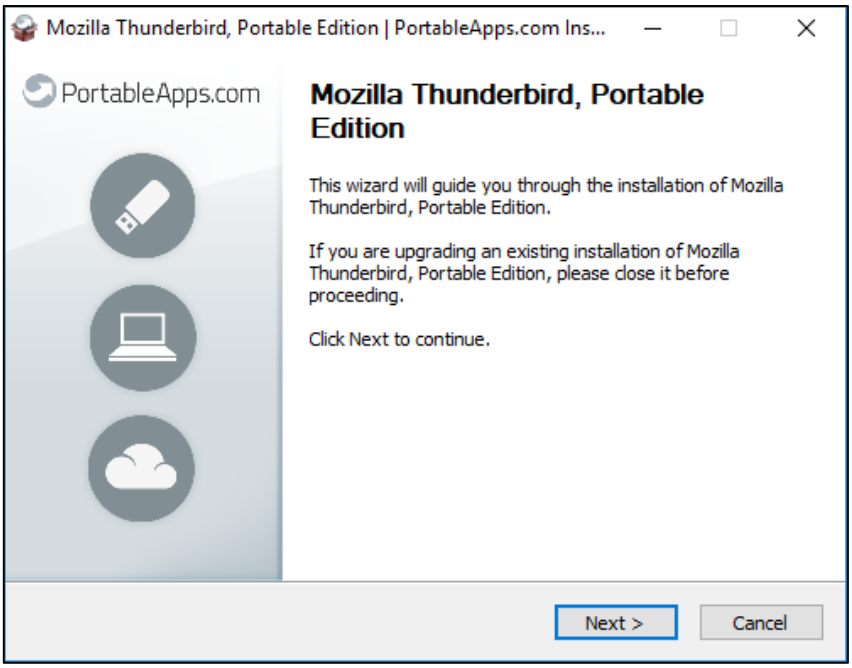
2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

Exercise— Collecting email via IMAP using Thunderbird Portable

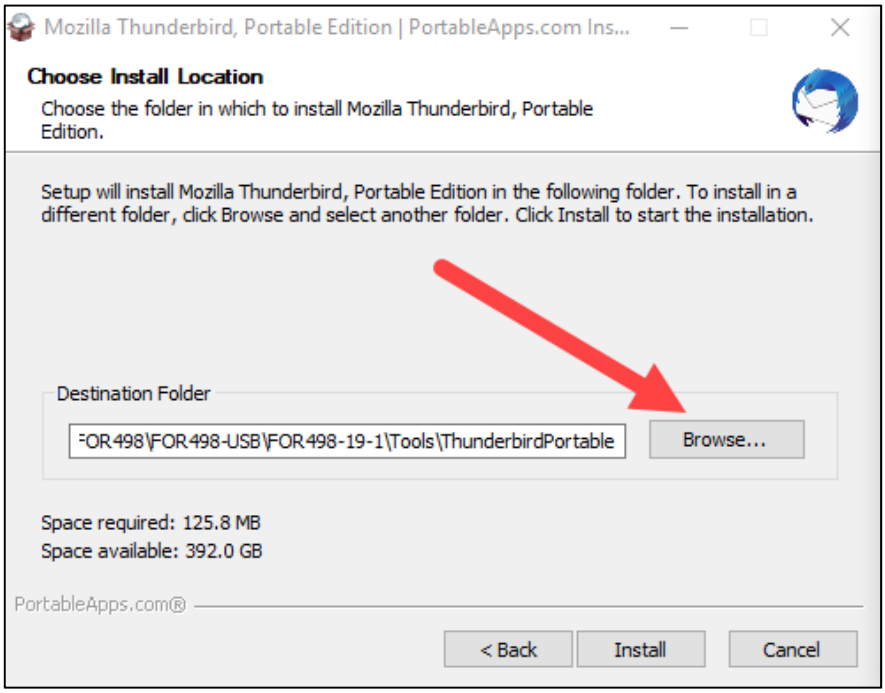
NOTE: This stage requires you to have access to an IMAP based email account, such as Gmail. If you already have a Gmail.com account, you can use this account for this part of the lab. If you do not have a Gmail account, create one at gmail.com for use in this lab. Any email account on an IMAP server will work fine for this lab.

1. Open **File Explorer** and navigate to **C:\Installers**

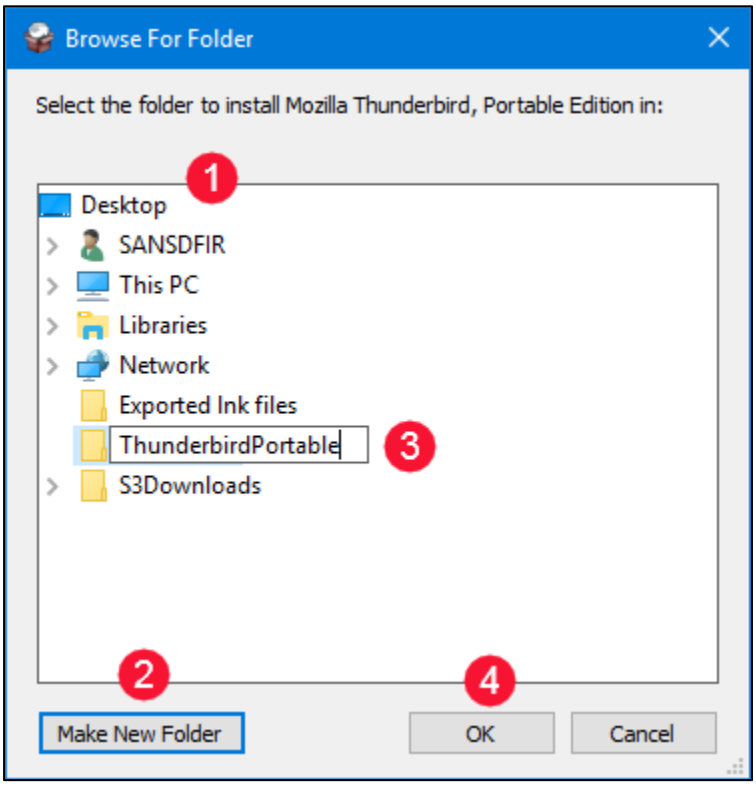
2. Double click on **ThunderbirdPortable.exe** to start the wizard.



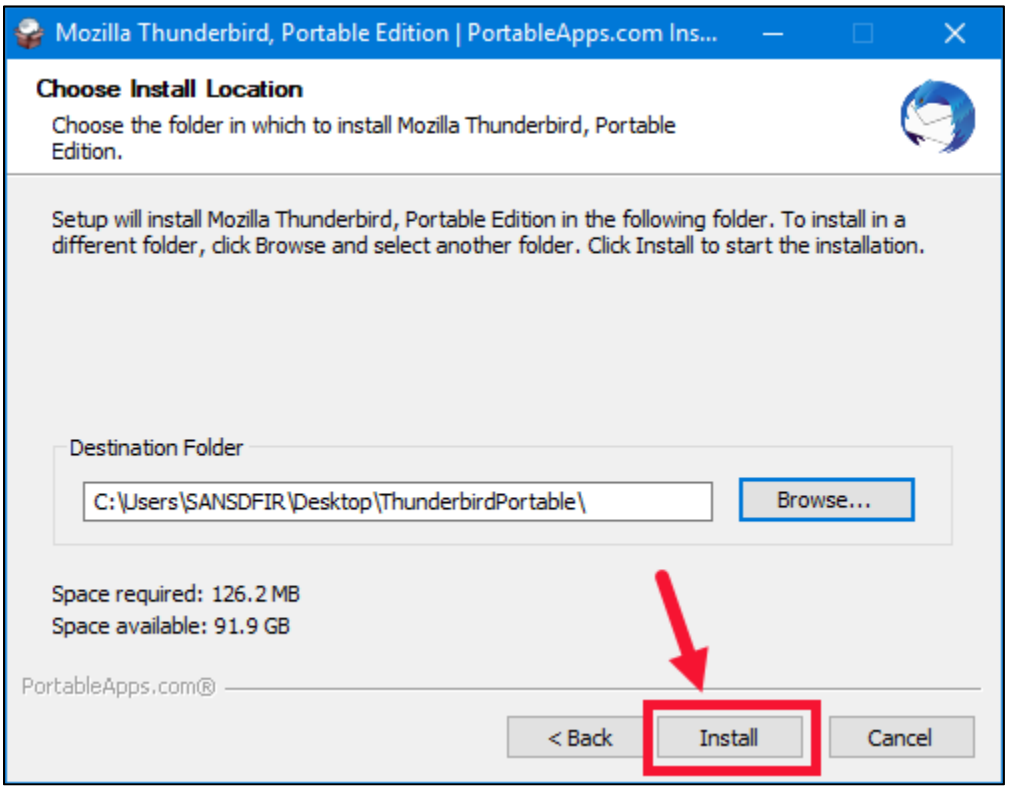
3. Click **Next**, then click the **Browse** button to choose a folder to extract **Thunderbird** to.



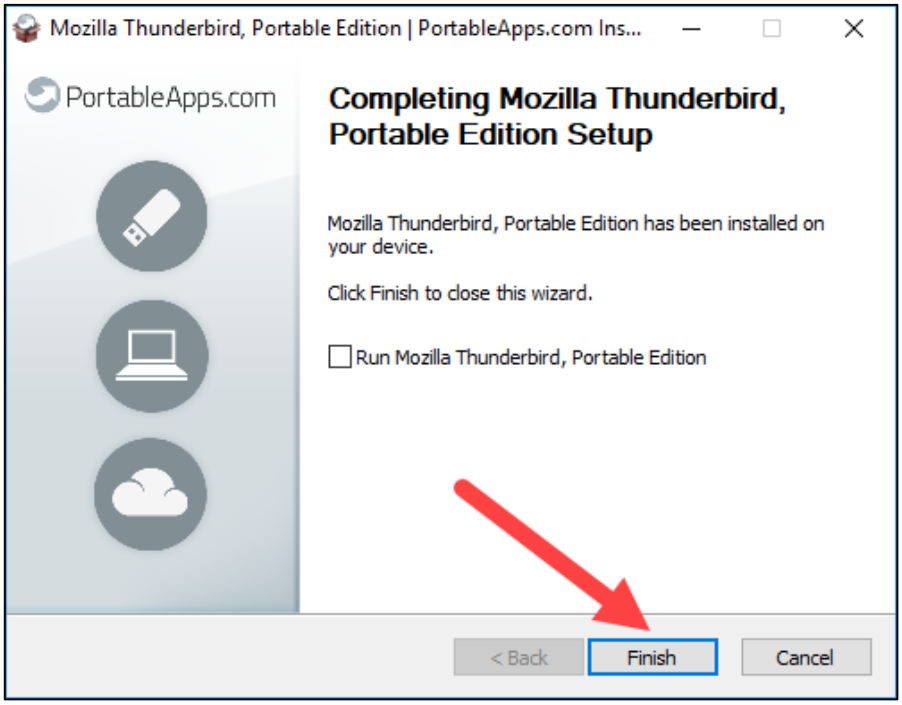
4. In the folder browser dialog, click on the **Desktop** folder, then click the **Make New Folder** button in the lower left. Name the folder **`ThunderbirdPortable`**, then click **OK**.



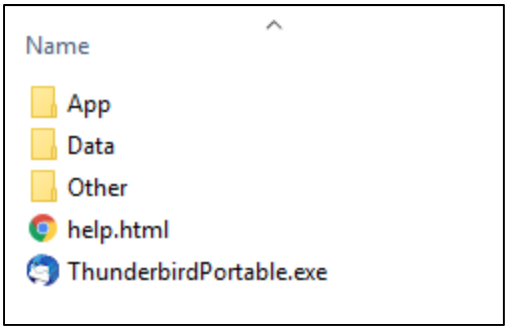
5. Click the **Install** button and wait for the files to be extracted.



6. When the extraction is finished, click the **Finish** button.



7. Open **File Explorer** and navigate to the **ThunderbirdPortable** folder on your **Desktop**.



This folder is a self-contained instance of **Thunderbird**. All data used by **Thunderbird** will be created and maintained under this folder, which means you can simply move it to a different computer, and it will work just like it does on the current machine.

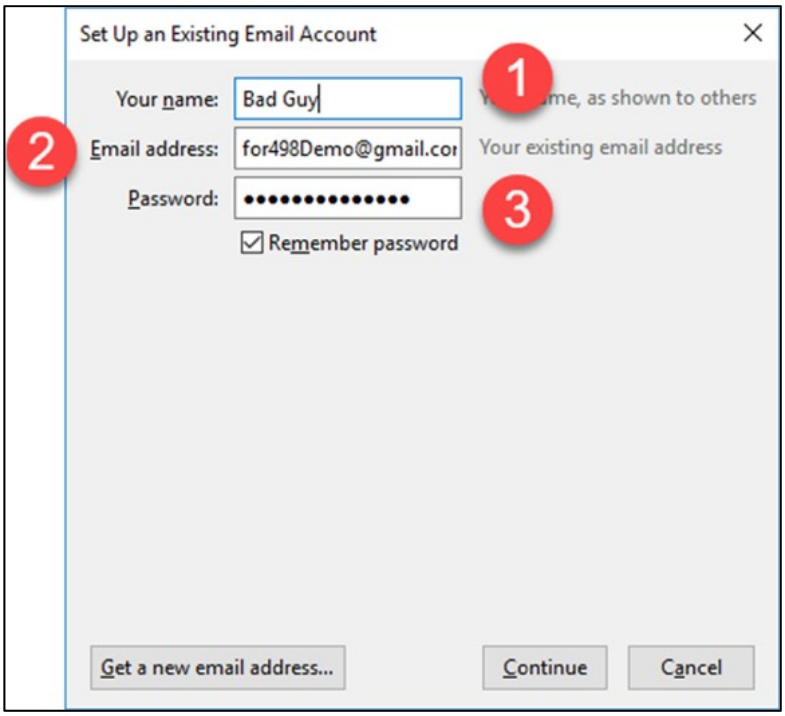
This makes a very convenient way to collect email data and have it in a self-contained program that lets you search, interact with attachments, organize data, and so on.

8. Double-click on **ThunderbirdPortable.exe** to begin configuration.

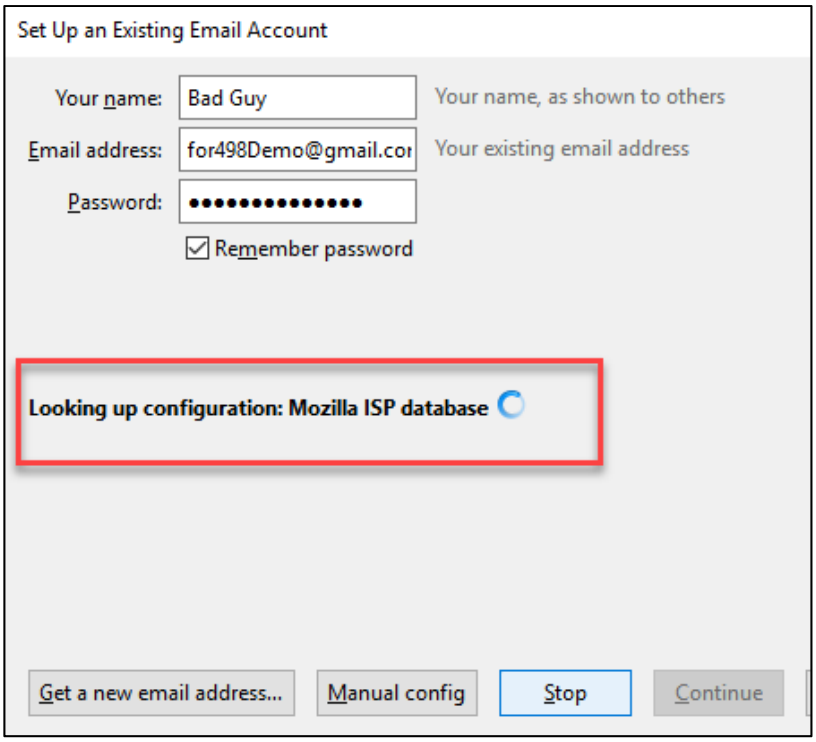
9. A wizard will be shown for initial configuration.

   Enter the name of the target account in the **Your name** field. This can be anything you want, but generally, if you know the user's name, enter it here.

   The other two fields are obvious. You will need the subject email address and account password.

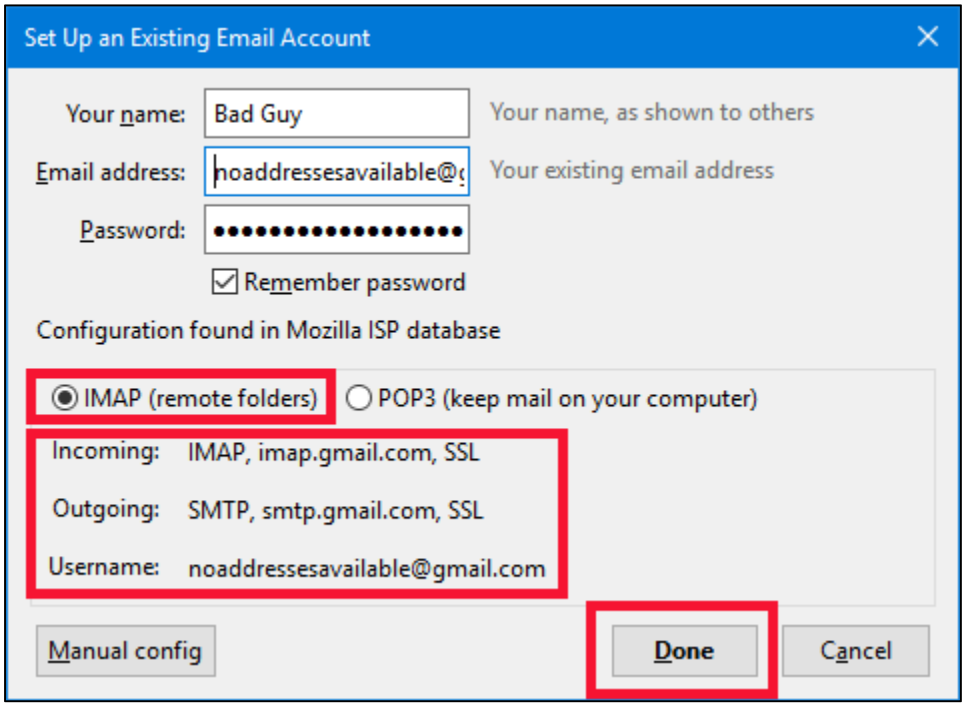


10. Click the **Continue** button and **Thunderbird** will attempt to auto-configure based on the information provided.

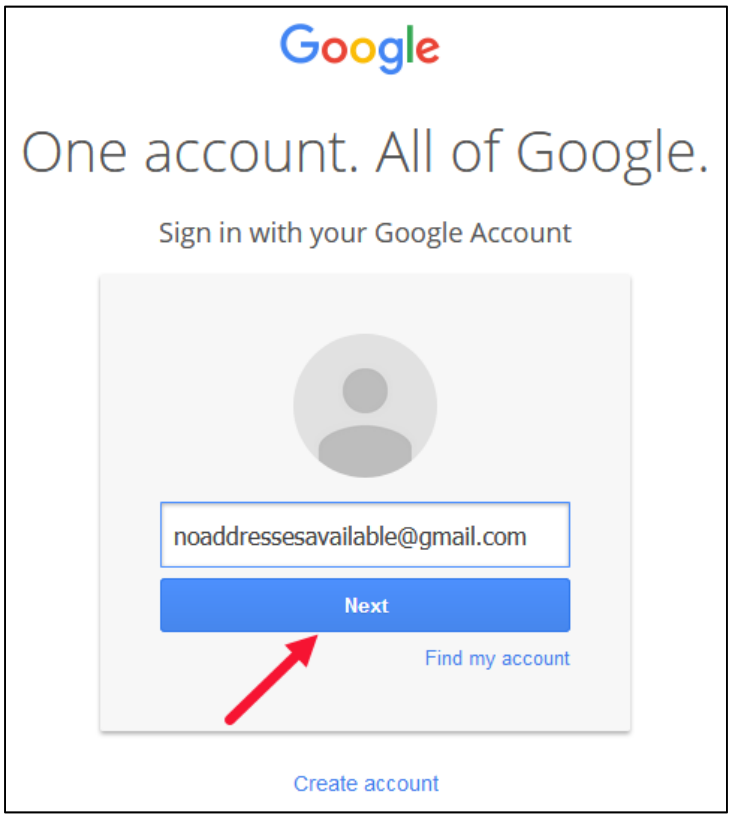


11. Once it finds the proper settings, they are displayed for review. If **Thunderbird** was not able to automatically configure itself, you can use the **Manual config** button to enter server settings manually.

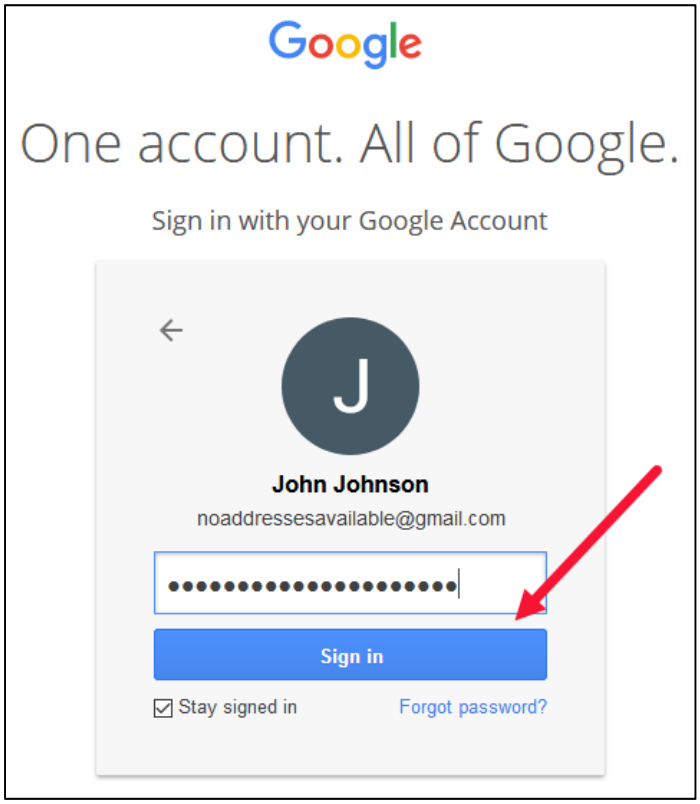
    Review the configuration, then click **Done**.



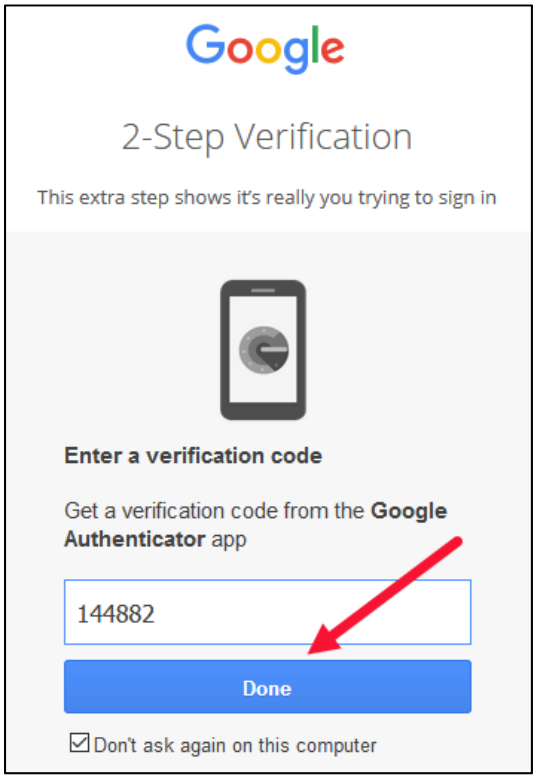
12. Depending on the email provider, **Thunderbird** may prompt for additional details, including two-factor authentication. In the case of **gmail.com**, Google requests more information. In the first dialog, click **Next**.



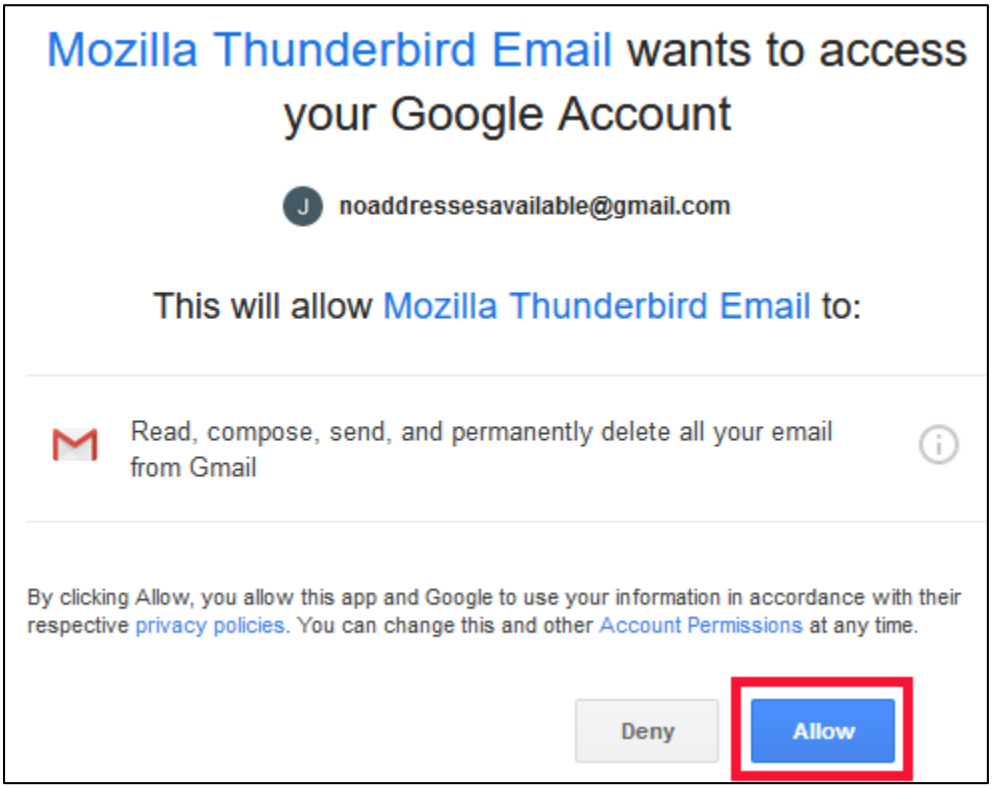
13. In the next dialog, enter the password for the account and click **Sign In**.



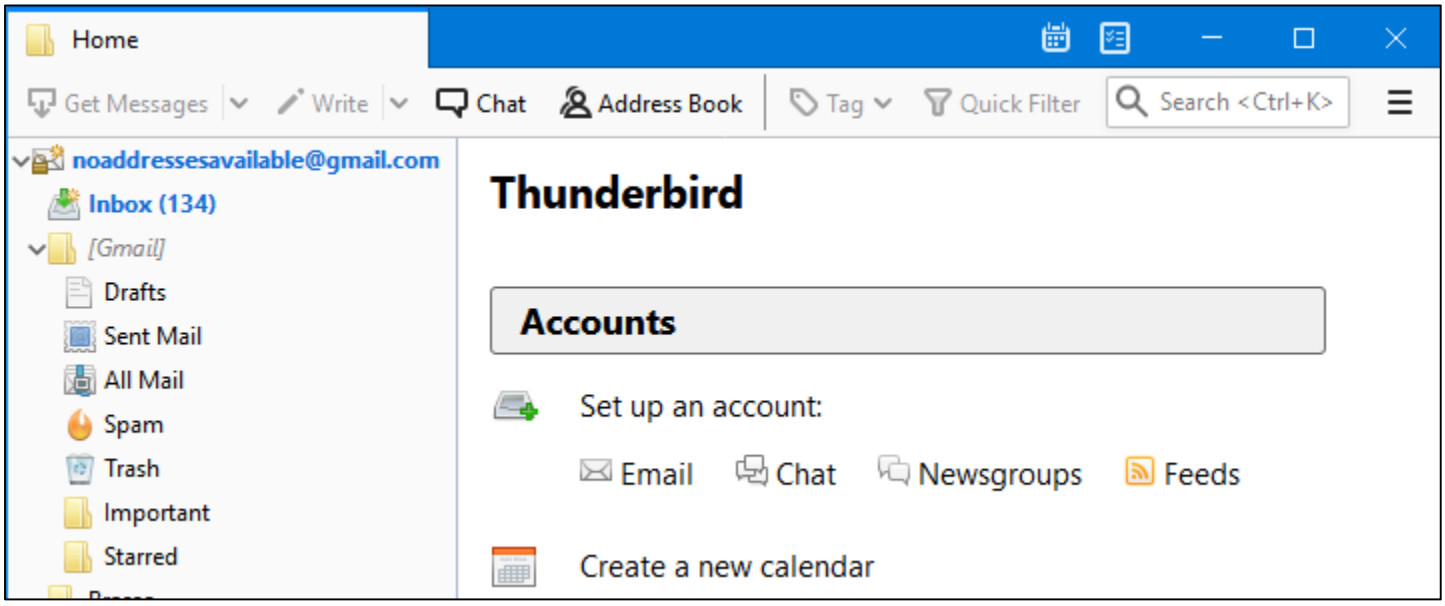
14. If two factor authentication (2FA) is set up (AND IT SHOULD BE!!!), you will have to enter the verification code, then click **Done**.



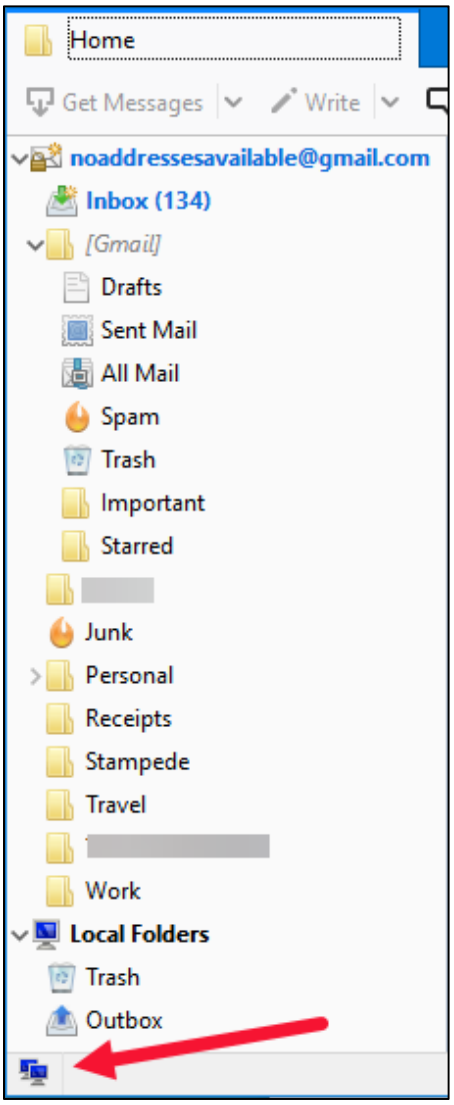
15. In the final dialog, you need to allow **Thunderbird** access to the email account. Click **Allow** and Thunderbird will then connect to the email box and start pulling email.



16. **Thunderbird** will automatically look for common folders on the email account and display information about it. In the example below, notice there are four emails in the **Inbox**.

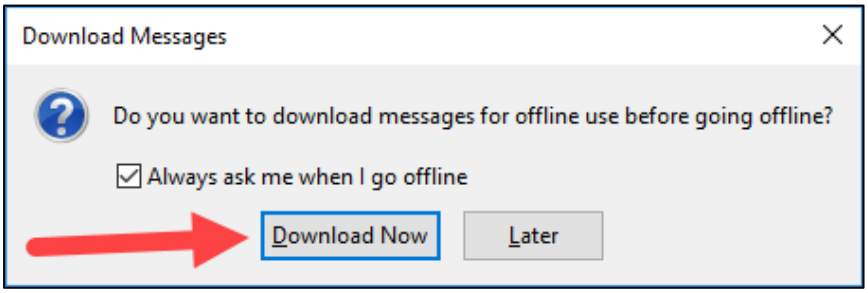


17. While you can now start clicking on folders to interact with the content, it is more useful to download a copy of the entire mailbox before doing so. In order to do this, click the button in the lower left corner (it looks like two monitors).



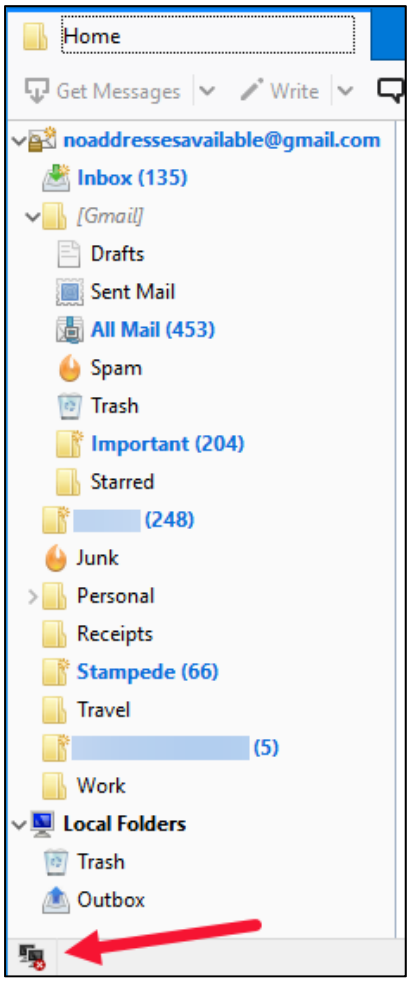
18. **Thunderbird** will ask if you want to download messages for offline use.

    To download all of the email, click **Download Now**.

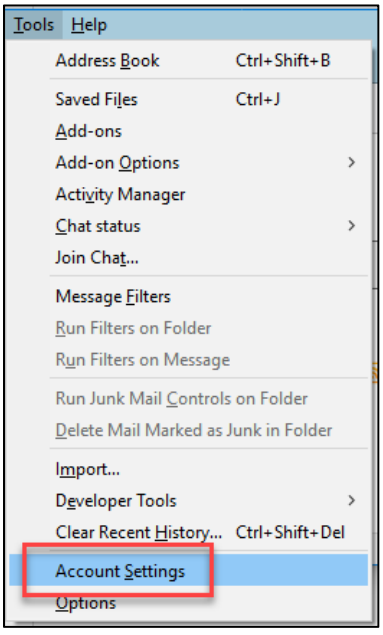


    Once **Download Now** is clicked, **Thunderbird** will iterate the entire mailbox and download email for every folder and subfolder. Depending on how much email is in the account, this can take a while, so be patient.

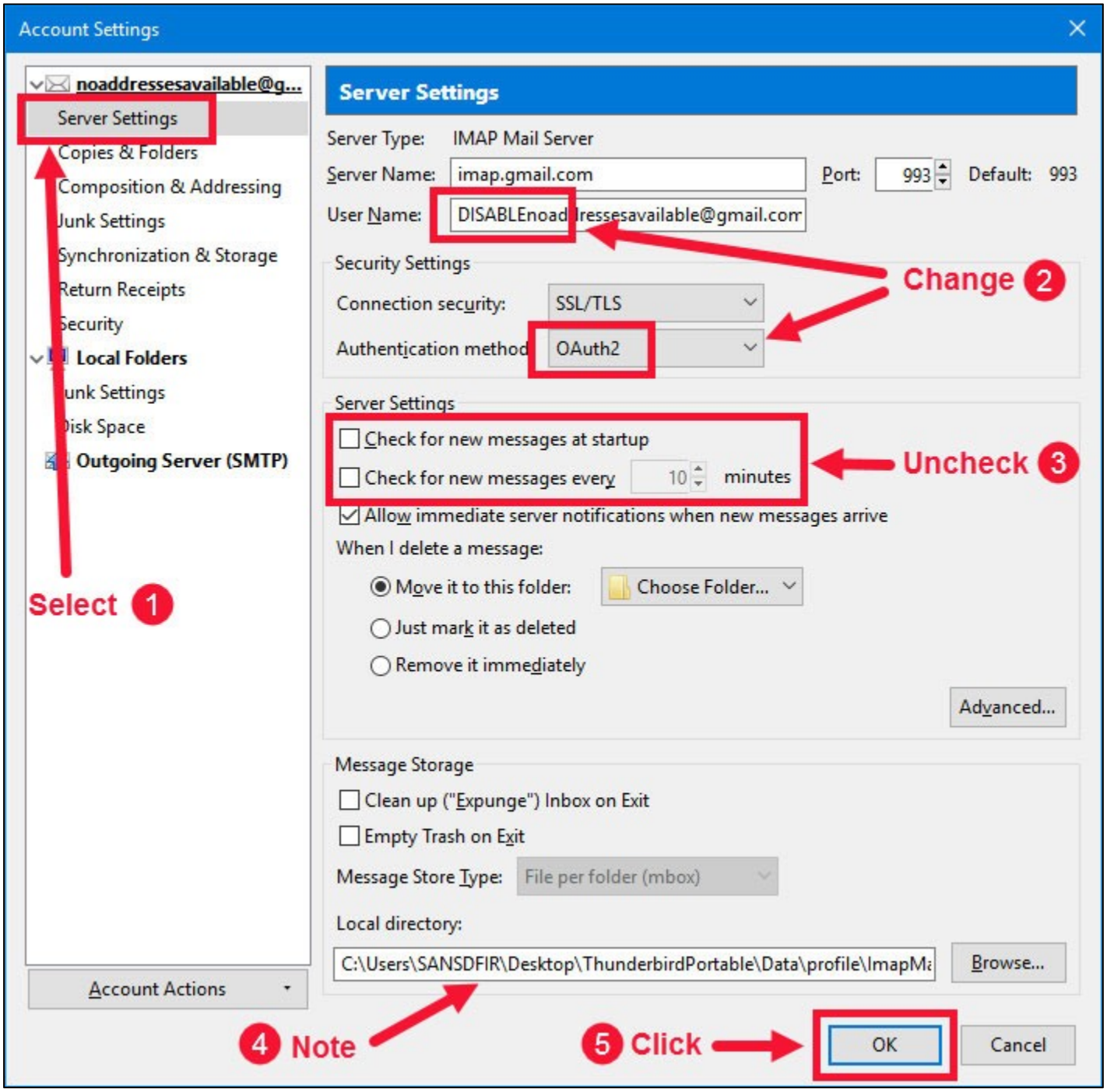
19. When finished, the icon in lower left changes to show you are now offline.



20. From here it is a good idea to tell Thunderbird to **NOT** connect to the account to update email contents. To show the menu, press **<ALT>**, then **Tools**, then choose **Account Settings**.



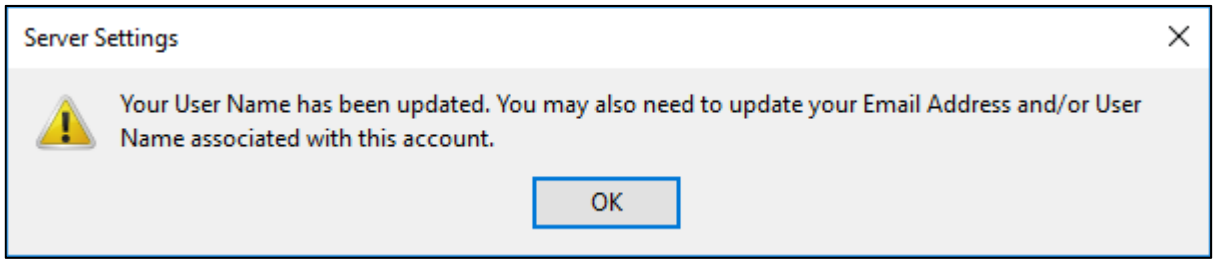
21. Click on the **Server Settings** tab on the left. To prevent connection to the account, we recommend you change the **User Name** (so the login will not work anymore at all), as well as unchecking the **Check for new messages** options. Finally, select something other than **OAuth2** for **Authentication method**, such as ***Normal password***.



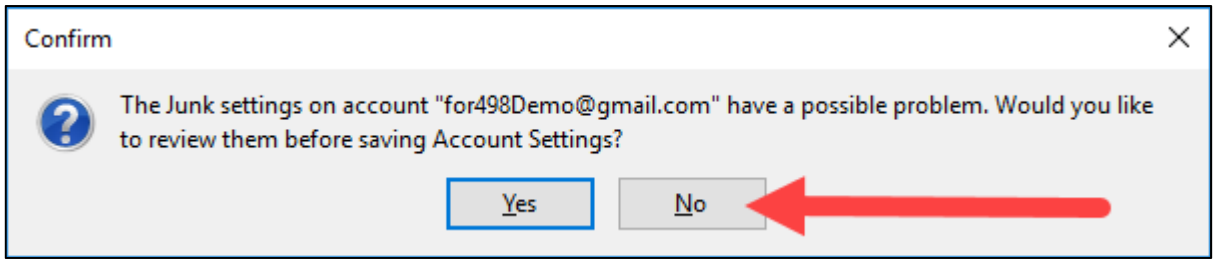
Also notice in the above screenshot where the profile lives for this particular account. As we mentioned earlier, this is a fully portable installation of **Thunderbird**, so all profiles exist in a subfolder of where the main executable was found. For this example, the full path is:

C:\Users\SANSDFIR\Desktop\ThunderbirdPortable\Data\profile\ImapMail\imap.gmail.com

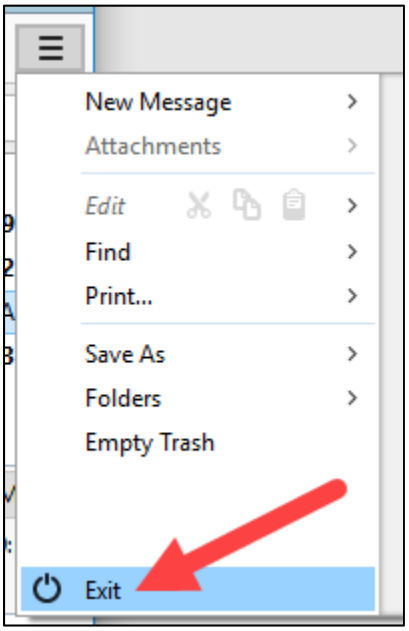
22. Click **OK** to close the dialog, then click **OK** to close the dialog informing you that the user name was changed.



23. If prompted about Junk settings, this can be ignored. Click the **No** button to return to the main interface.

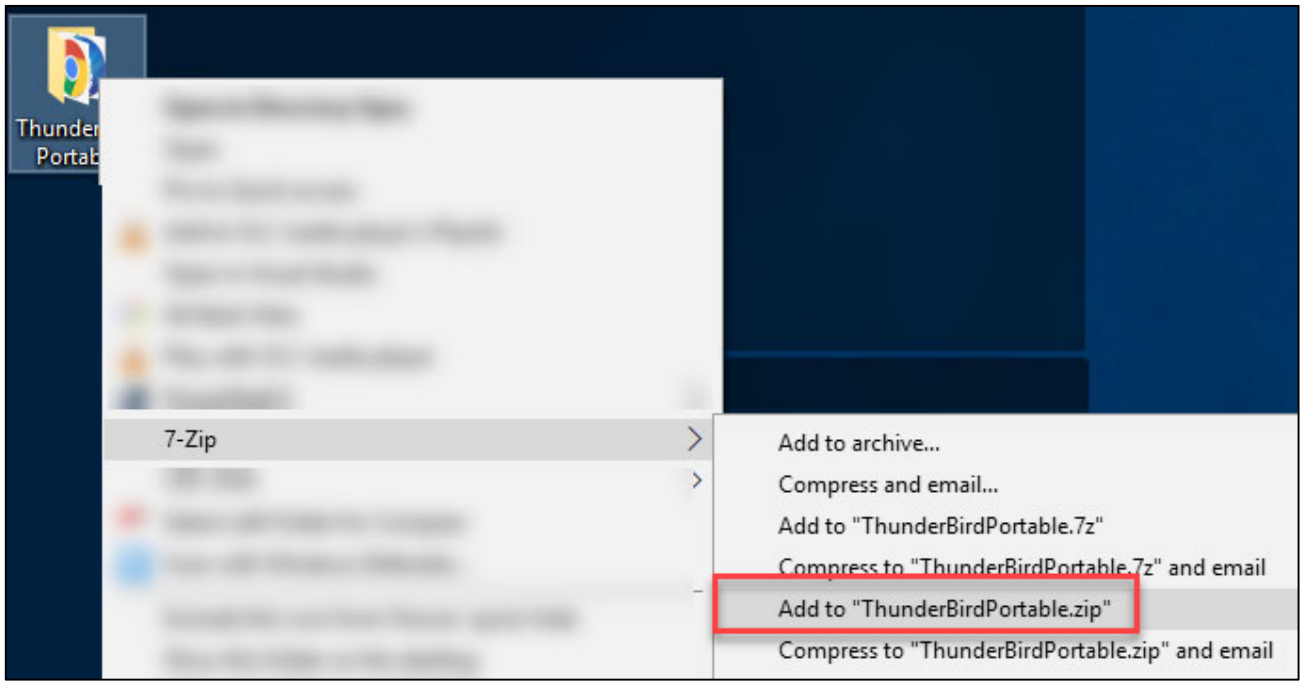


24. With collection complete, exit **Thunderbird** by using the menu in the upper right (the three stacked horizontal lines).



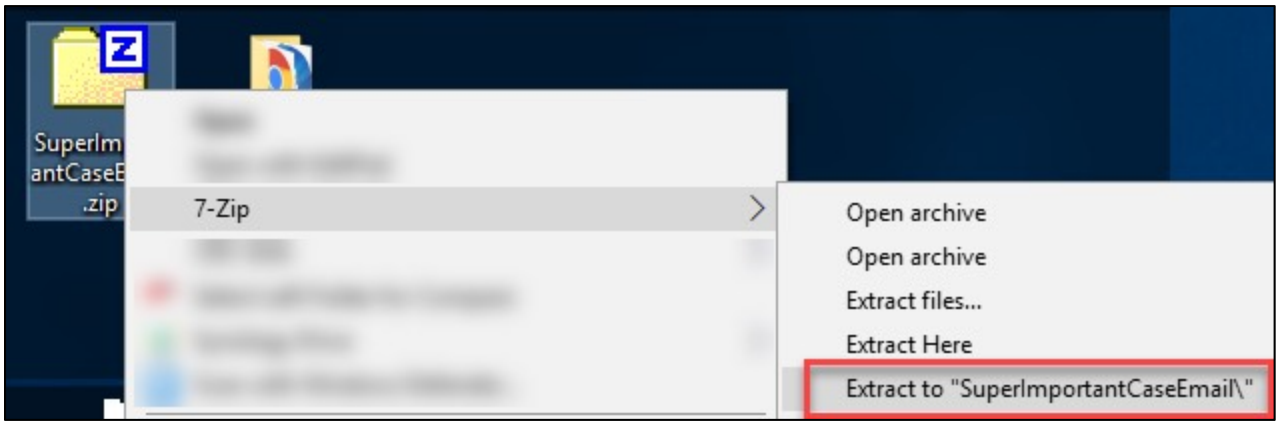
25. Once **Thunderbird** has exited, find the **ThunderbirdPortable** directory on your **Desktop.**

26. Right-click on the **ThunderbirdPortable** directory, choose **7-Zip**, then click **Add to "ThunderbirdPortable.zip"**



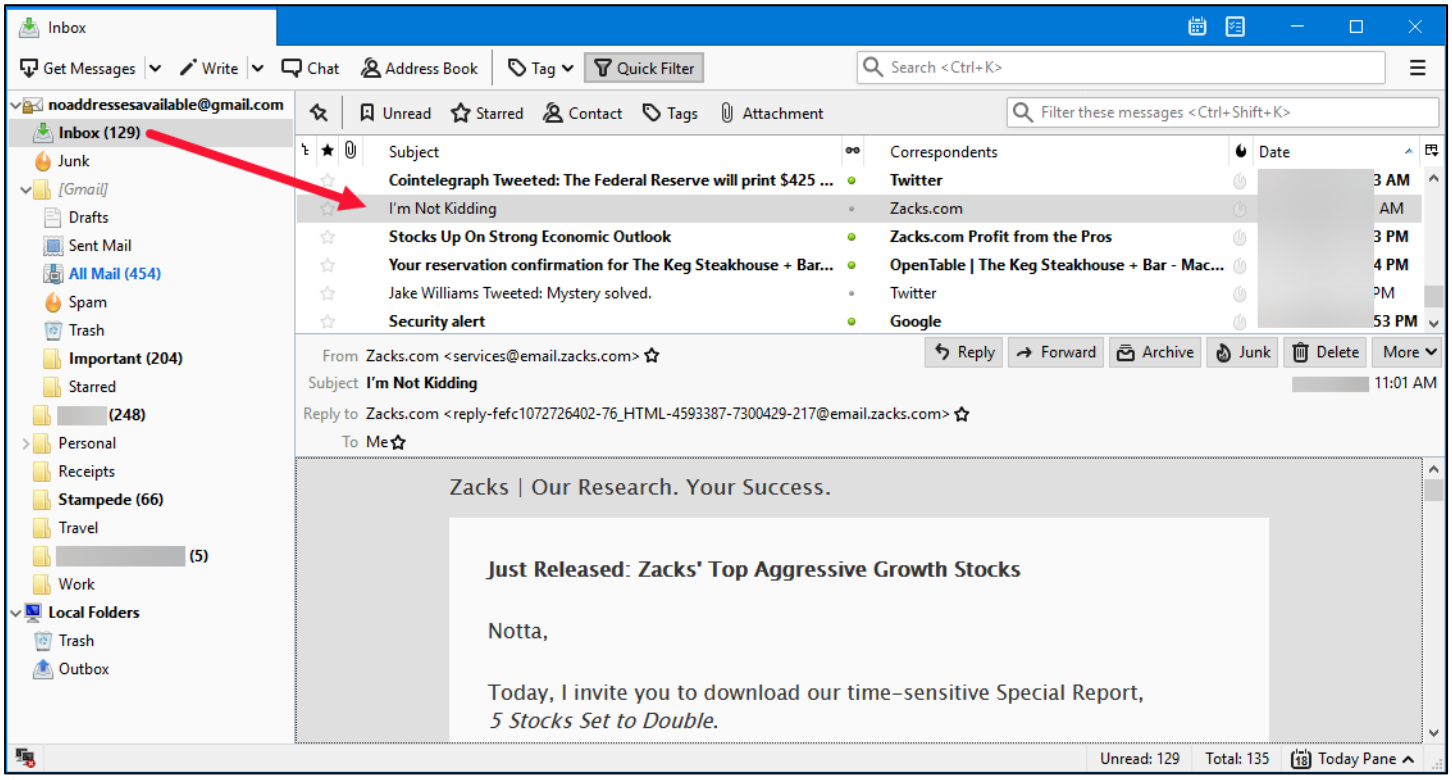
27. Once **7-Zip** finishes, rename **ThunderbirdPortable.zip** to something that is more meaningful to your investigation, then preserve this file as your original evidence per your policies.

28. Finally, let's look at what we have available in the email account we just preserved. Right-click on the name of the archive from the previous step, choose **7-Zip**, then choose **Extract to <zip name>**.



29. When the extraction is complete, double click the directory that **7-Zip** created and start **ThunderbirdPortable.exe** by double clicking the executable.

30. Once **Thunderbird** starts, look in the lower left to confirm you are still offline. Click on the **Inbox** and read some mail!



You now have a full archive of everything that was on the IMAP server and you can interact with it in a completely disconnected way. The original data, if you followed along with these steps, will never be changed, because the **User Name** was changed. This prevents **Thunderbird** from connecting back to the mailbox and changing things like read status, moving emails, etc.

At this point you can do many different things. In a lot of cases it helps to create new folders related to searches, emails of interest, and so on. You can then leverage the search feature in **Thunderbird** to find relevant emails and move those emails into these new folders, etc.

Of course, you would be doing this against a COPY of the email archive as outlined above to ensure that your initial collection is preserved exactly as it was when it was collected.

Exercise— Using F-Response for Network Acquisition

NOTE: This exercise will require at least two systems (laptop, VMs, etc.)

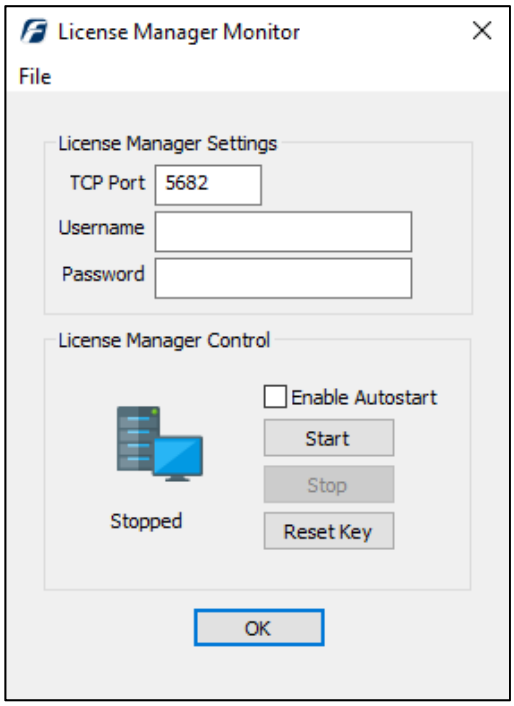
NOTE 2: *These directions outline how to use F-Response. You will need to tailor things like the machine names, usernames and passwords, and so on to match *your* setup*.

1. Open **File Explorer** and navigate to **C:\Installers**

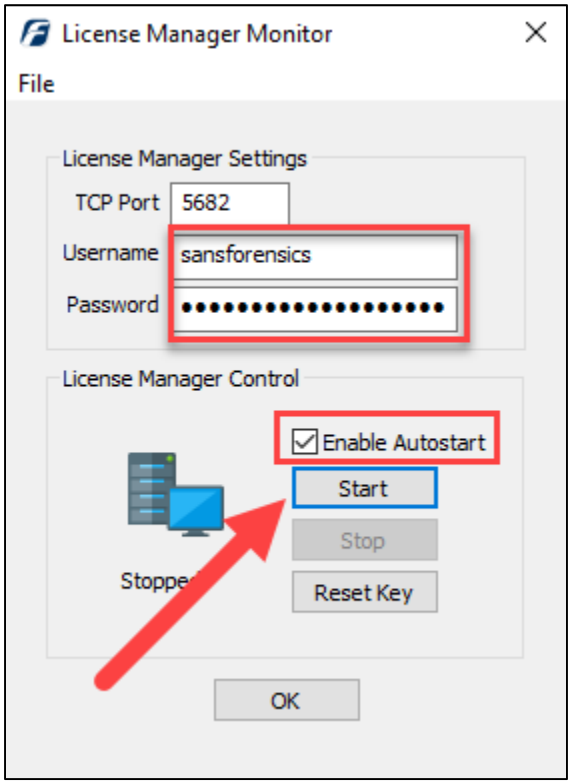
2. Run the **F-Response** installer, **F-Response-Installer.exe**, and use default options for everything. This will install all the software and create shortcuts under the Start menu.

3. Plug your **F-Response** Dongle into a USB port on your computer.

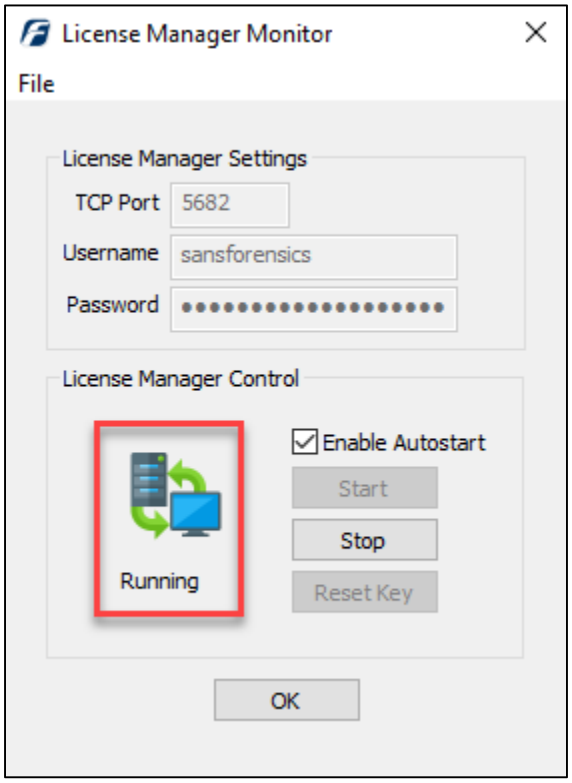
4. Using the **Start** menu, open the **F-Response License Manager Monitor** under **F-Response**. There are several steps to do here.



5. Enter a **Username** and a **Password**. These are the credentials that agents will use to connect. **F-Response** will enforce a strong password, so pick a decent one. Check the **Enable Autostart** checkbox.

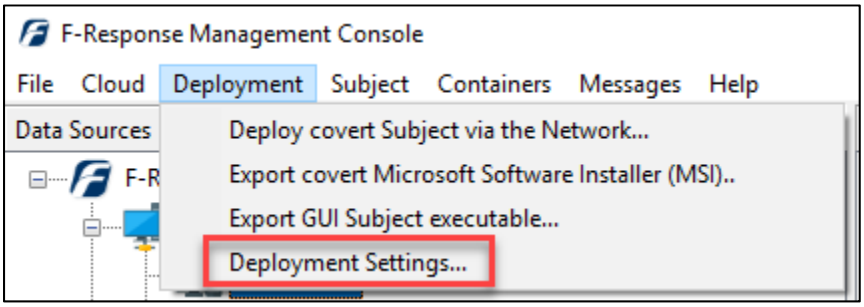


6. Click **Start** to save the setting and bring the service up.



Note that the TCP port used for the license monitor must be reachable by whatever computers you wish to interact with. If this is a LAN, ensure local firewalls are configured to allow this port to be passed through the firewall.

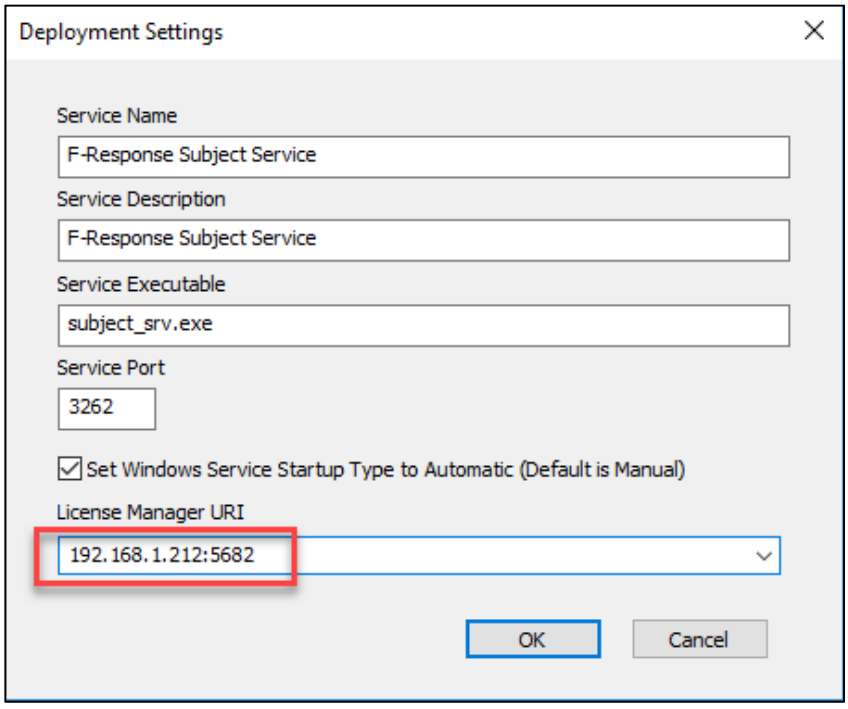
7. Using the **Start** menu, open the **F-Response Management Console** under **F-Response**, then click **Deployment** -> **Deployment Settings…**



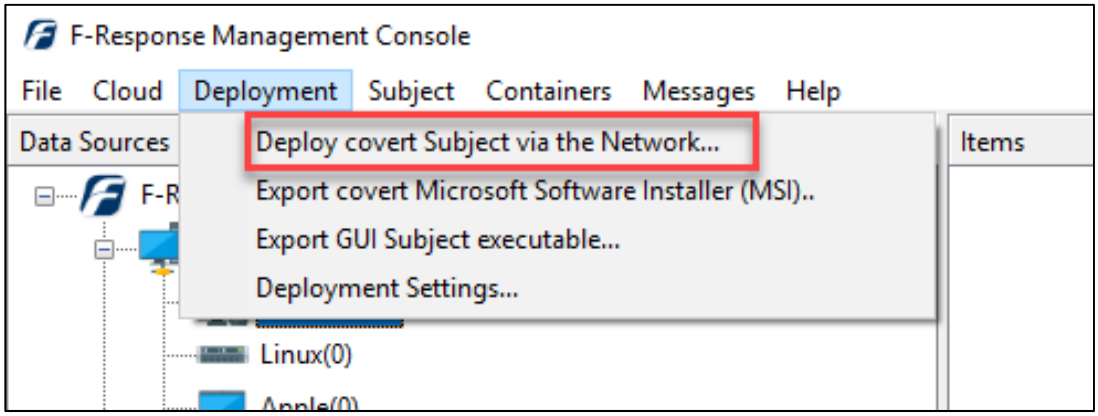
8. In the **License Manager URI** drop down, select the hostname and port (or IP address and port) you want the client to use, then click **OK** to save the settings.

   To find your current IP Address, open a **PowerShell** prompt and type:

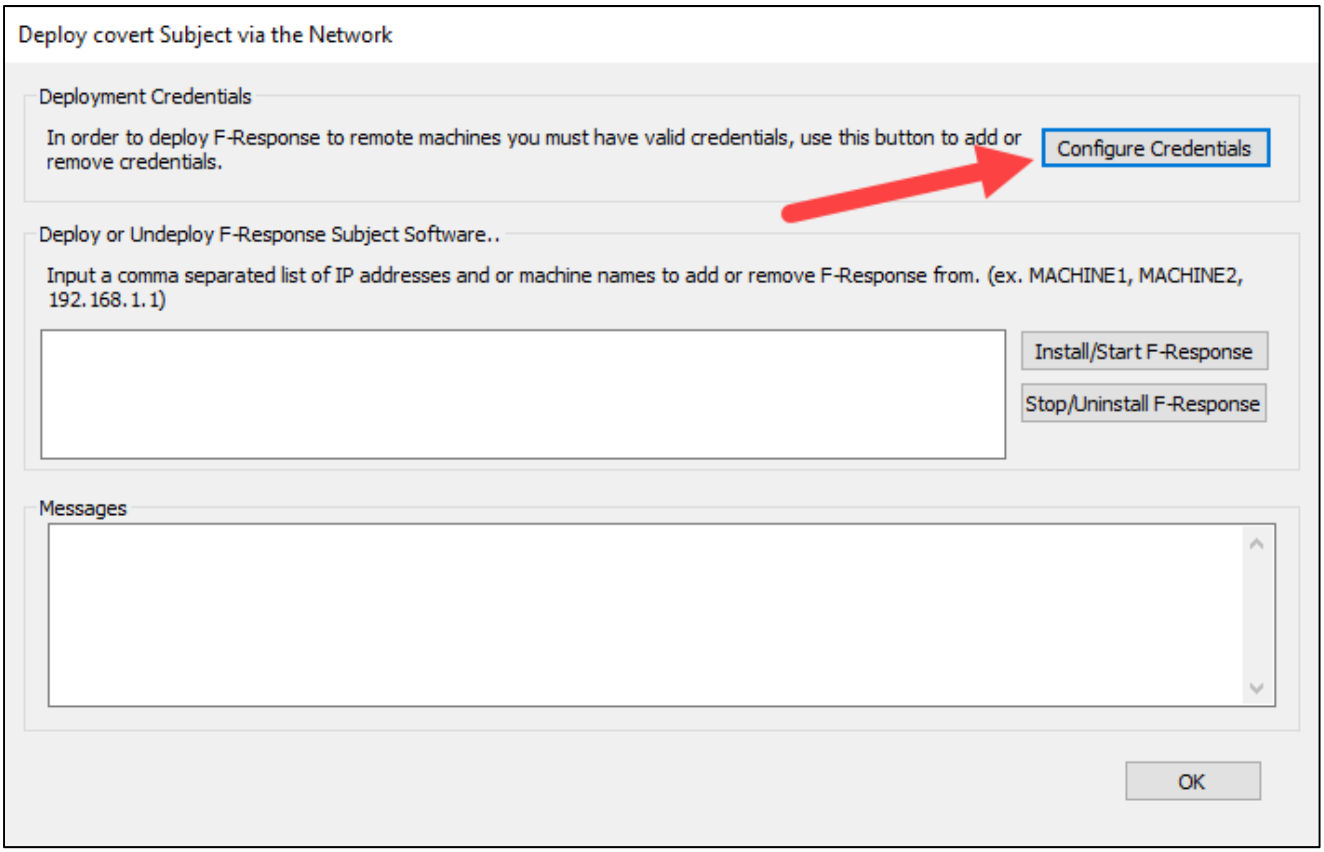
```
Ipconfig.exe
```



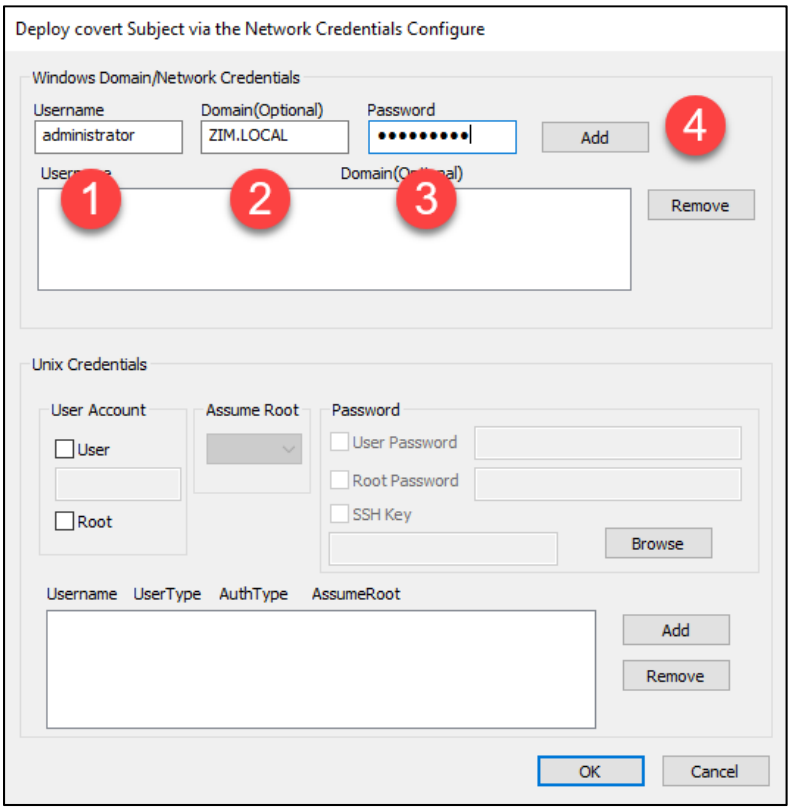
9. In the main Console view, click **Deployment** -> **Deploy covert Subject via the Network…**



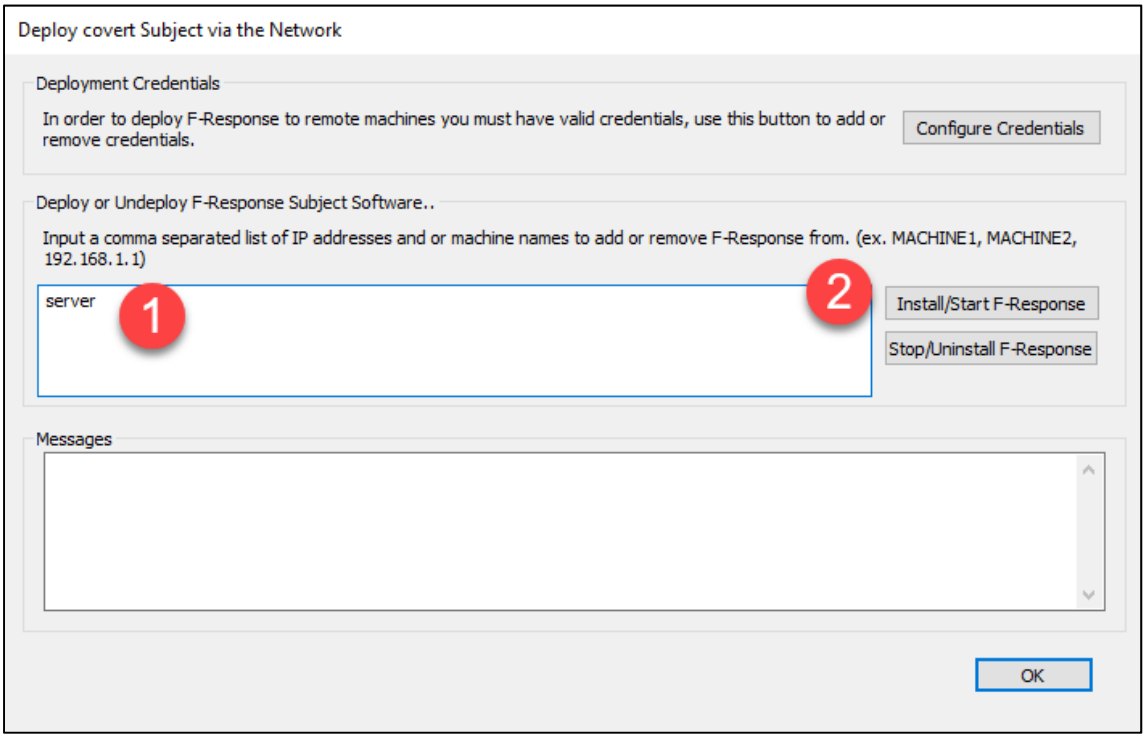
10. Click the **Configure Credentials** button



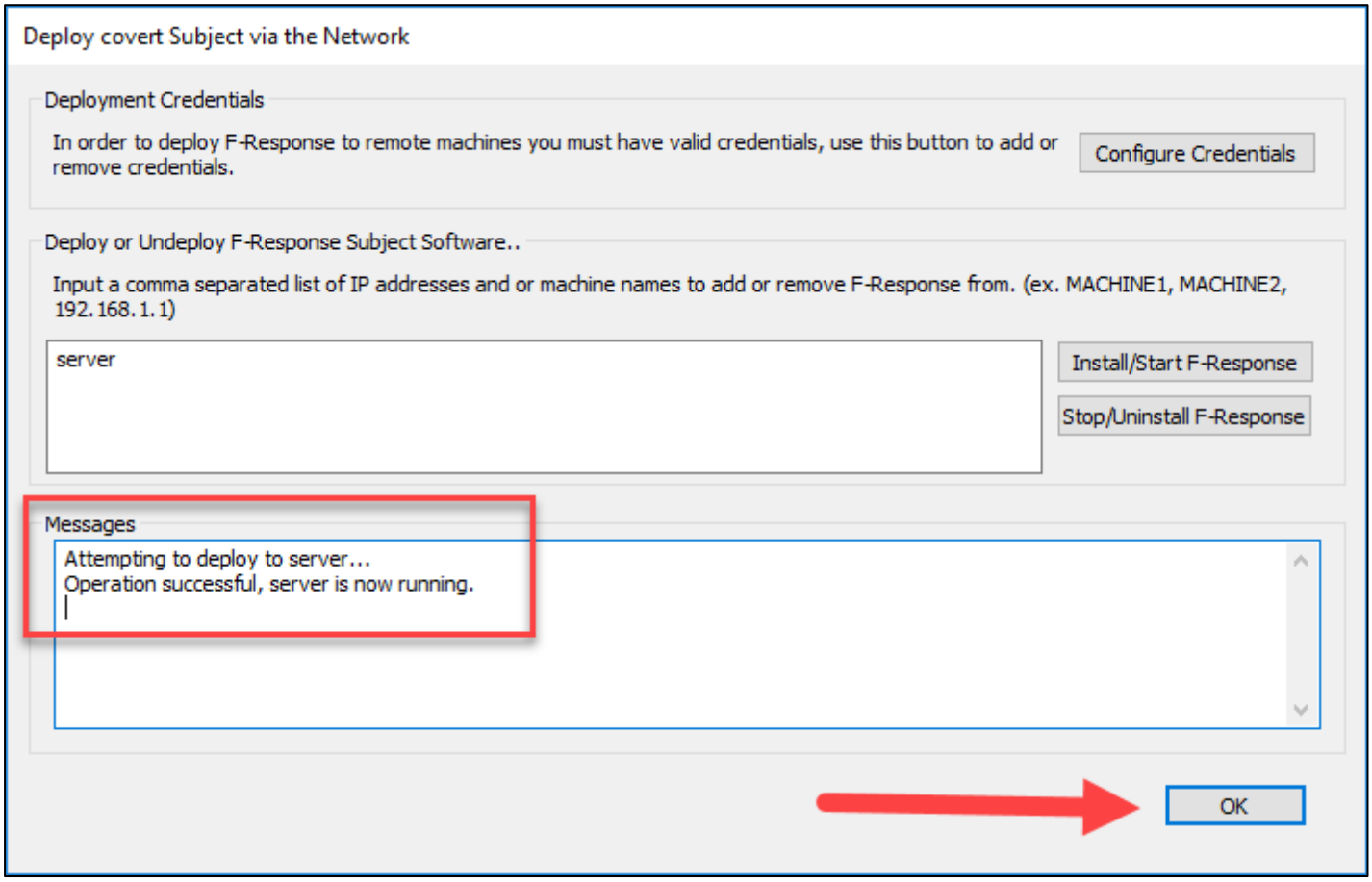
11. Depending on your environment, enter the credentials to use when connecting to and installing the **F-Response** agent. In the example below, a Windows domain admin level account is being used. Once the information is entered, click the **Add** button, then click **OK** to save.



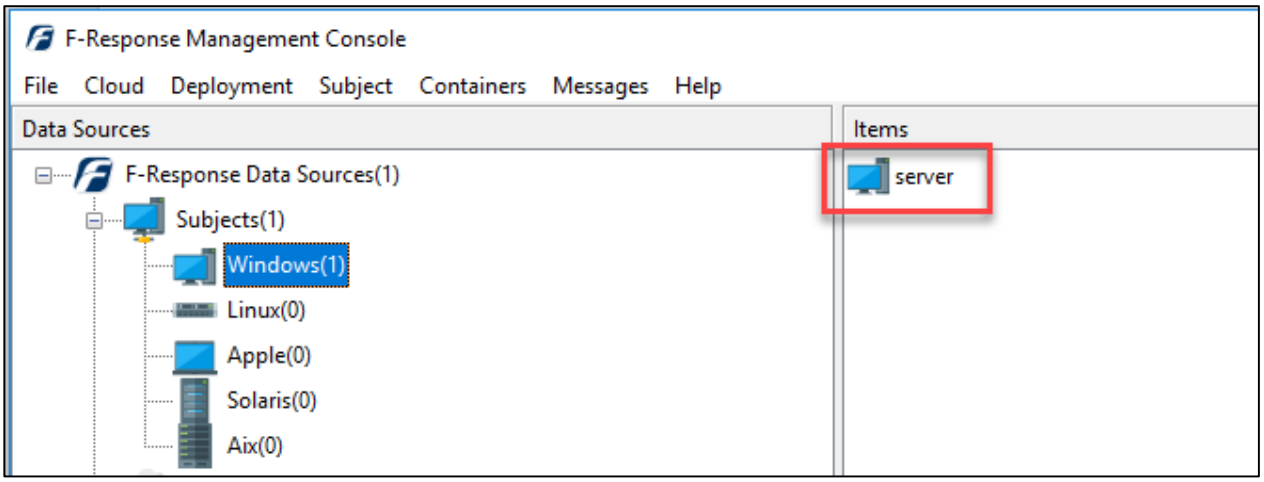
12. In the **Deploy or Undeploy F-Response Subject Software** section, enter one or more computers to install the F-Response agent to, then click **Install/Start F-Response**.



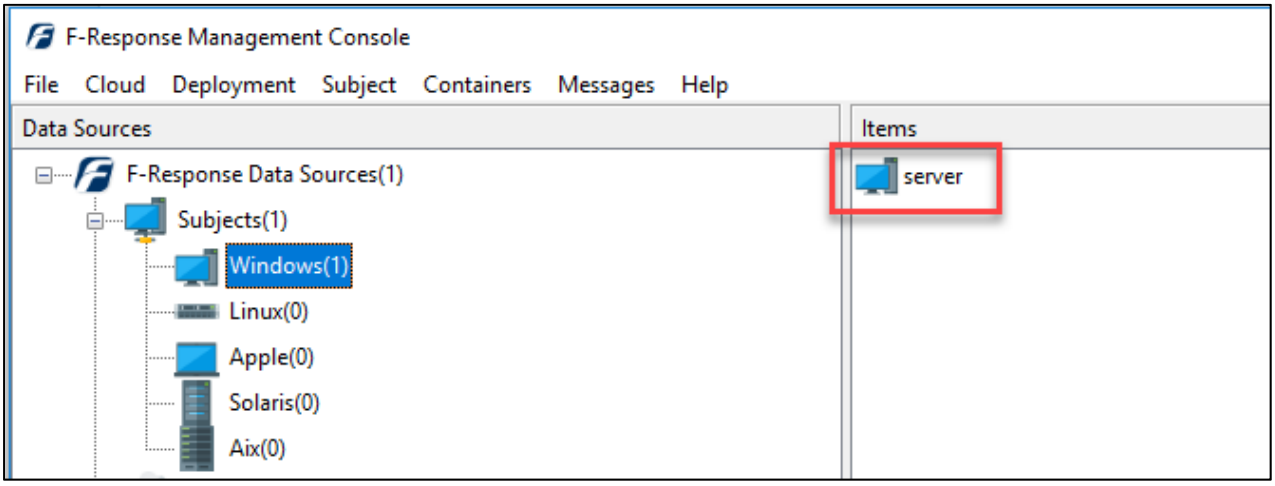
13. The agent is then pushed to each machine. Check the **Messages** box for status. When all installations have finished, click **OK**.



14. Back at the main Console, check for entries under the operating system you just deployed against. In the example above, we pushed to a Windows machine, which would look like this:

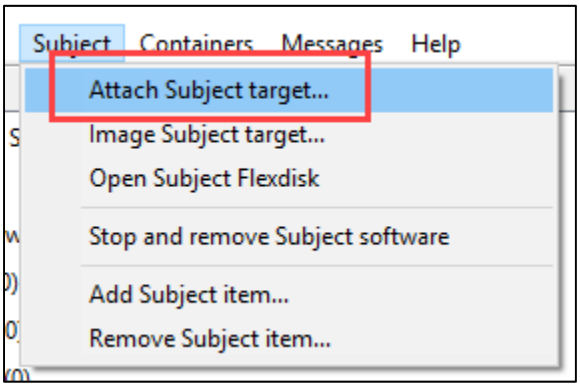


15. Under the **Items** group, click an entry to select it, then right-click to bring up a context menu.



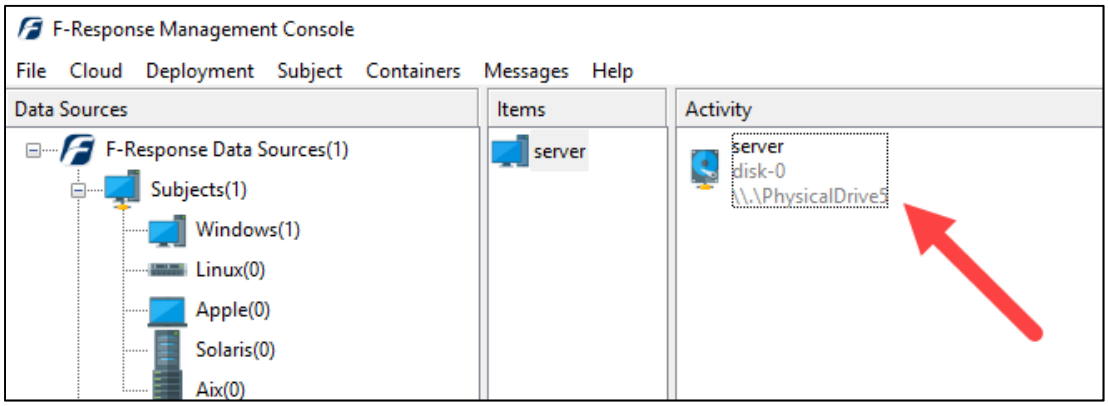
The **Subject** menu at the top can also be used to access these options.

16. From the context menu, click **Attach Subject target…**

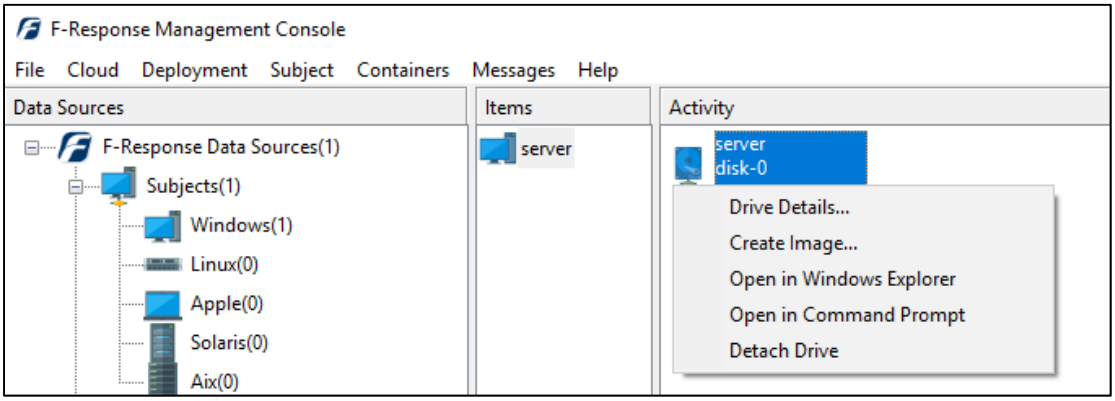


17. In the **Attach Drive** dialog, select a target to connect to, then click **Attach Drive**.

18. **F-Response** will then connect the selected target to the local machine and display this information in the **Activity** list on the right side.

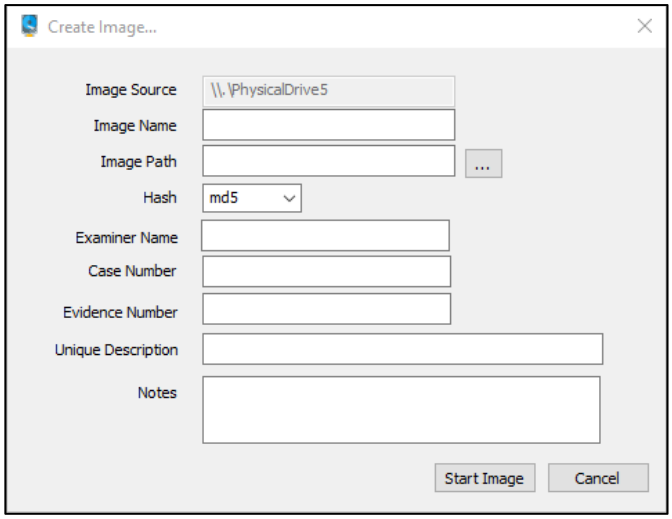


19. Select an item under **Activity** and right-click to interact with the target.

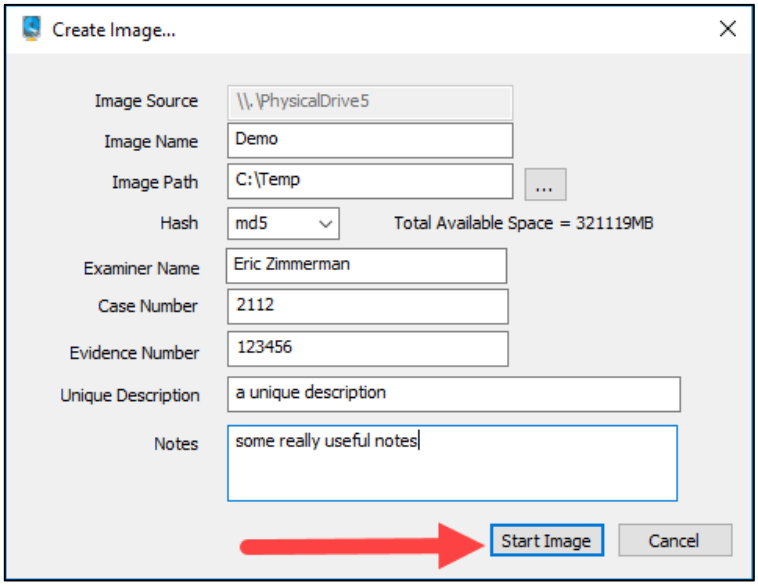


20. The remote target is now available on the local machine for browsing (via **Open** in **Windows Explorer** for example) if you mounted a physical disk. If you mounted a logical volume or memory, you can image this device (**\\.\PhysicalDisk5** in the example above) using whatever means you like.

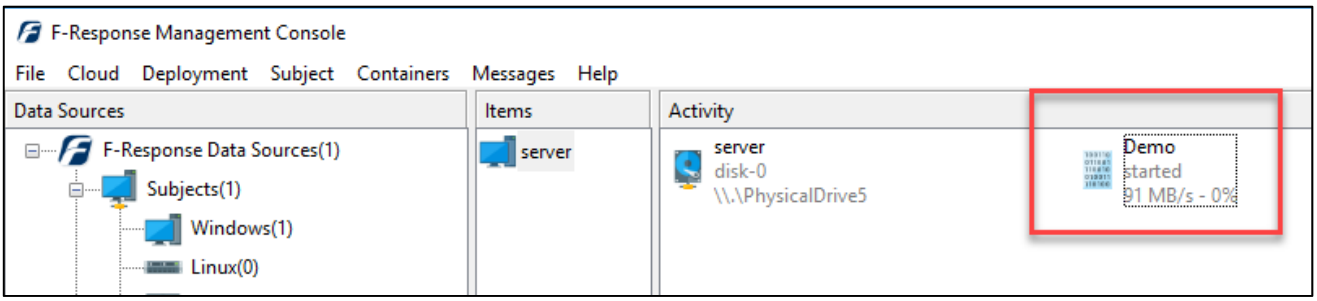
21. **F-Response** can be used to image disks directly from the Console. Right-click on an **Activity** and choose **Create Image…** from the menu to display the **Create Image** dialog



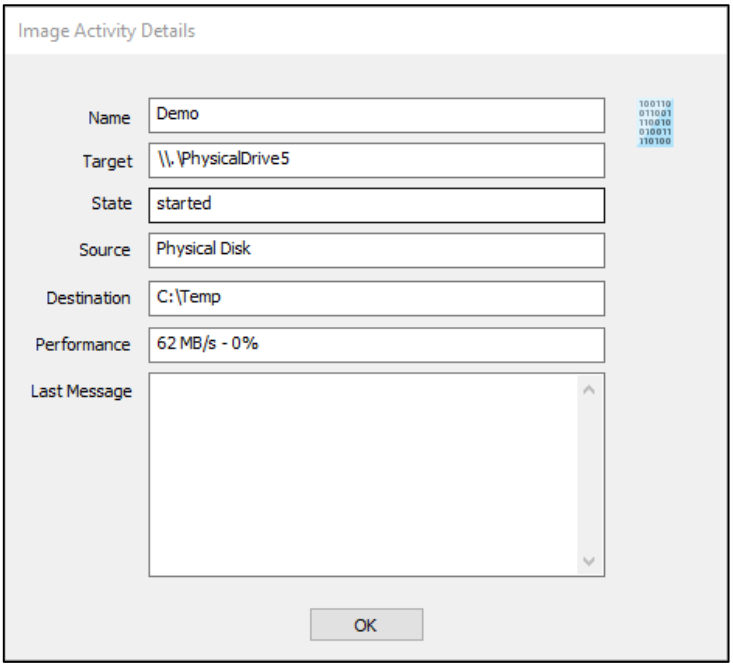
22. Fill in the details and click **Start Image**.



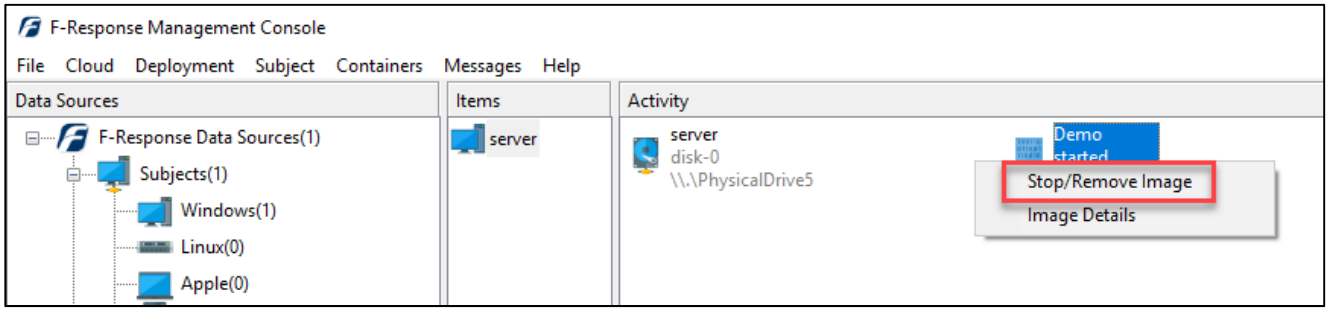
23. **F-Response** starts imaging and displays progress next to the **Activity** being imaged.



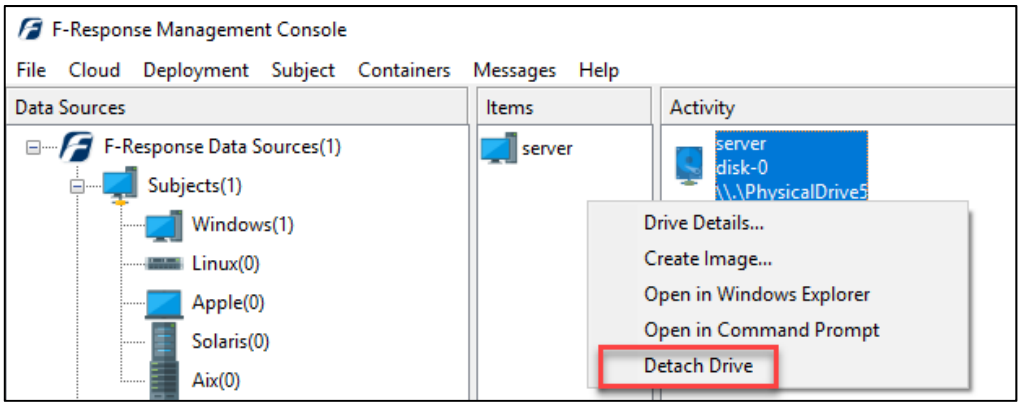
24. Double click on the status for more details.



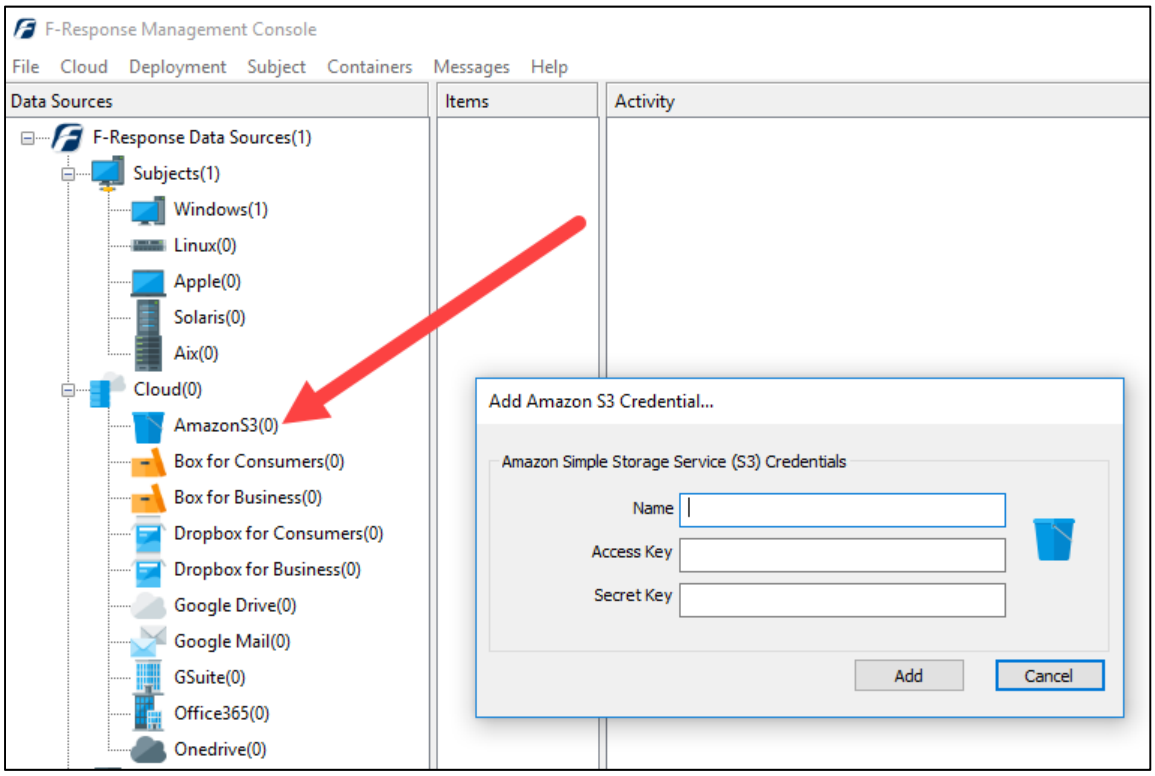
25. To stop the imaging process, right-click on the status display and select **Stop/Remove Image**.



26. When finished with the target, right-click and select **Detach Drive**.



DOUBLE BONUS: Use **F-Response** to access and collect the **S3** bucket from above, as well as the Gmail account from the **Thunderbird** section of this exercise. Double click on an entry under **Cloud** to begin the process.



Exercise—Key Takeaways

- Network acquisition allows for using a wide range of tools to collect data.
- Be sure to take into consideration the bandwidth available to you when performing acquisition over the network. If you are on a slow link, consider collecting only the most critical data you need first.
- F-Response makes it easy to target many different operating systems and cloud providers in the same interface.

This page intentionally left blank.

Exercise 5.1A—Online Attribution – Sam Spade

Background

Most everything today has an online component. Hardly a digital forensics investigation goes by that doesn't have evidence pointing off the storage device and into cyberspace. Emails, web history, addresses, URLs and IPs. What can we do to further our investigation?

As a last resort, we always have the option of subpoena or warrant, but given the inherent difficulties (not to mention cost) in pursuing either, this might not be the avenue we wish to pursue initially, if at all. Surely there must be a mechanism we can use in at least some cases to get us closer to our goal.

From online posts on social media, to comments on blogs or websites, people are "saying" more and more than ever before. In some cases, these comments cause real damage, and the perpetrators need to be tracked down.

Objectives

- Use Sam Spade to gather online identification information
- Use DomainTools to identify useful information about a domain or IP address

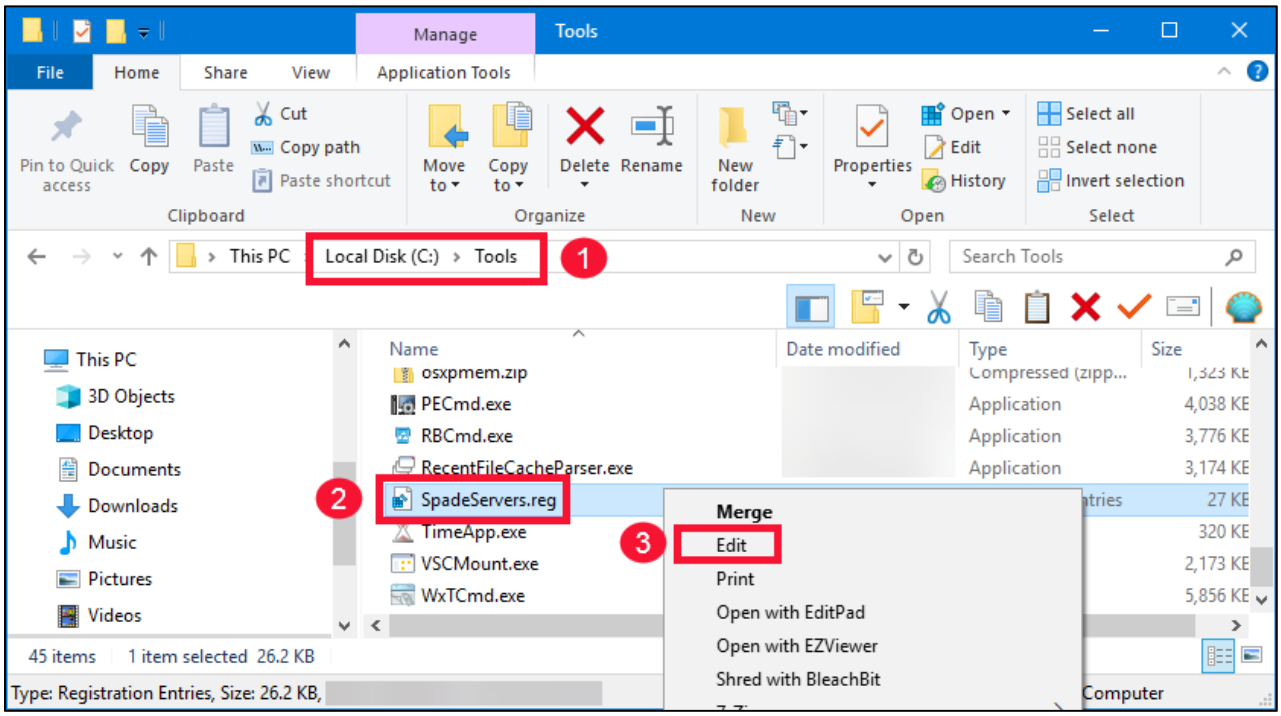
Exercise Preparation – Part 1

Sam Spade is a program that has been around for a great many years. It was very underused and completely unsupported, but it is still quite effective. Sam Spade accesses Whois servers in different ways to assist in extracting information. When Sam Spade was first released in 1997, it had a total of about 5 Whois servers in its database. Upon its final update in 1999, it had 30. Thanks to Brian Ingram at Consulting Investigation Services, who has kept the program database updated, it currently has 87, and Brian keeps updating the database as more servers become available.

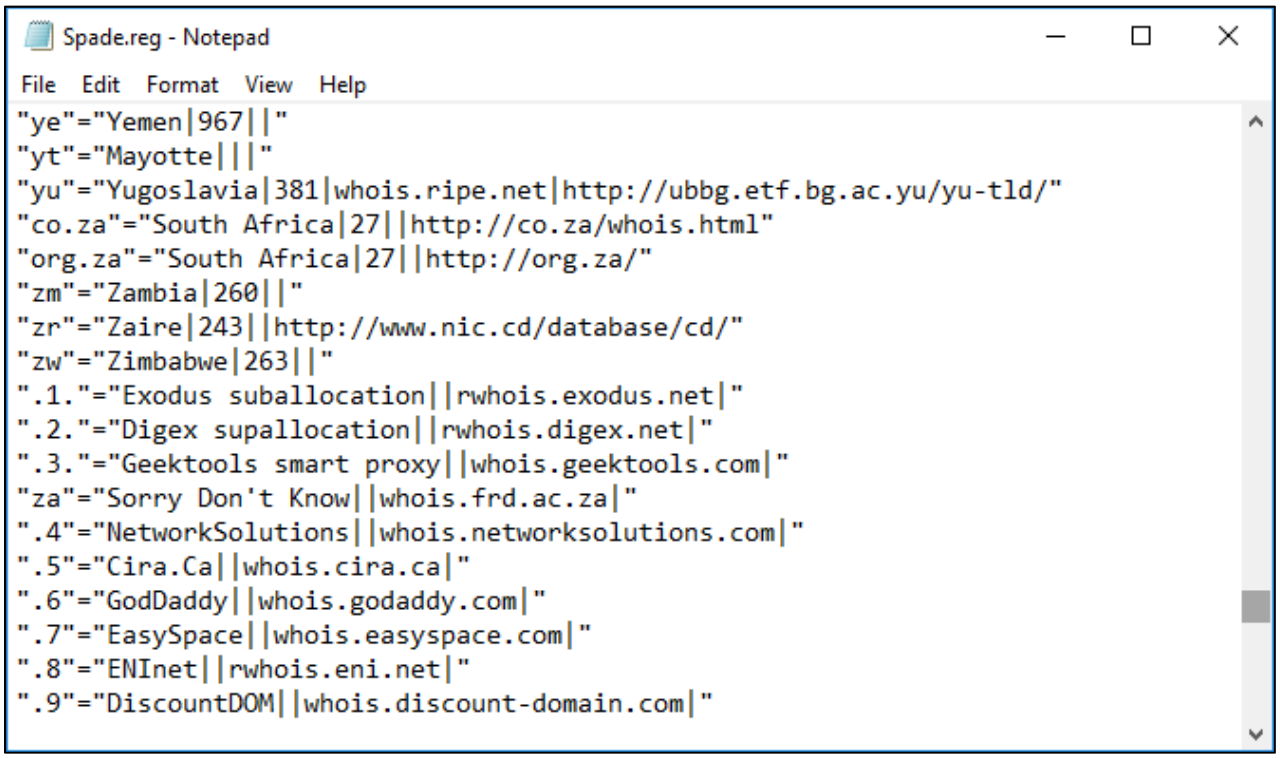
1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

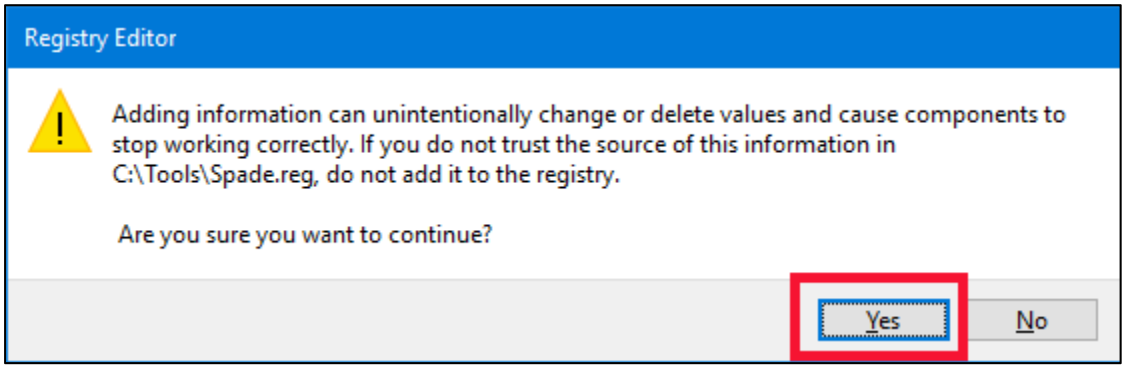
1. In your VM, use **File Explorer** to navigate to **C:\Tools** and note the file **SpadeServers.reg**. As previously stated, **Sam Spade** itself came with a limited amount of Whois servers. You are going to add over 50 more. First let's look at how this gets created. Right click on the **SpadeServers.reg** file and select **Edit**.



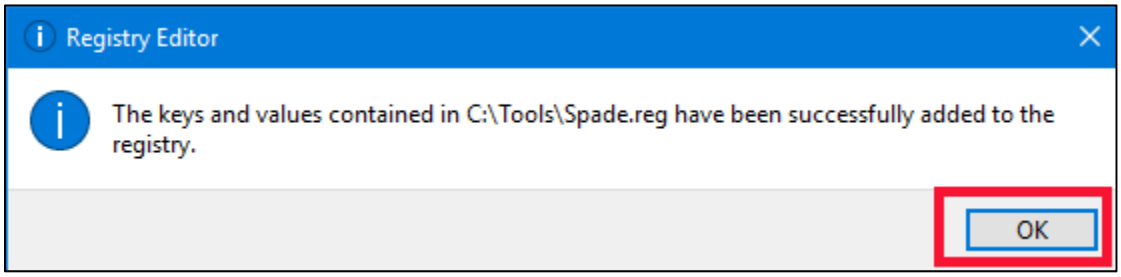
2. A **Notepad** window will open, and you will see the contents of the **.reg** file. Each line is a server. If you find a new Whois server that is not in the list, you can simply add it using the syntax in the screenshot, saving the file, and then committing the file to the Registry, which will be our next step. Take a moment to scroll through the list, and when done, close the file.



3. Let's commit this list of servers to the Registry, which will make them available in the program. Double click on the **Spade.reg** file and accept the warning box (You read it first, right?) by clicking **Yes**.

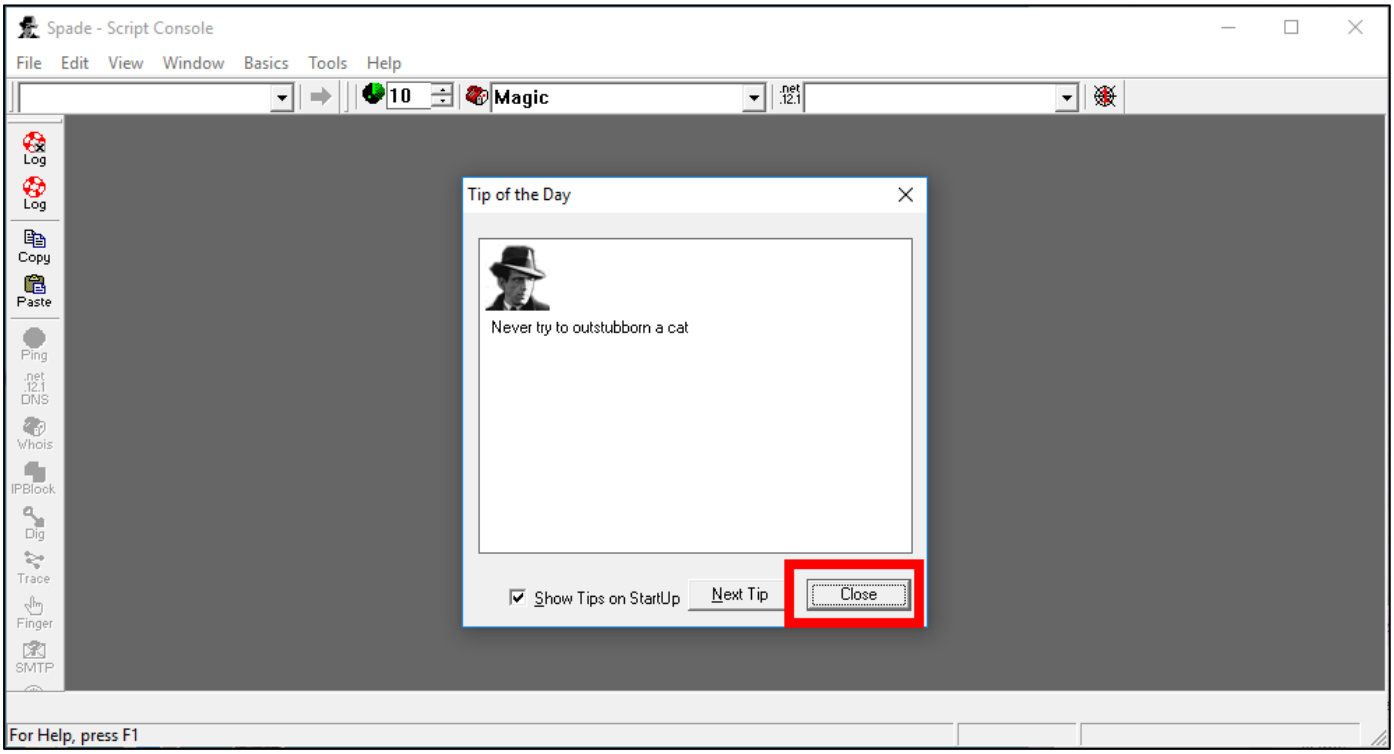


4. Click **OK**.



5. Close any open windows and double click the **Sam Spade** shortcut in the **Utilities** fence on your **Desktop**.

6. When the program starts, a "**Tip of the Day**" is displayed. You can stop this from happening by unchecking the appropriate box, but since this program is over 20 years old, it is kind of fun to read some of the tips. Just click on **Next Tip** a few times and read some of them. When you have read a few, click **Close**.



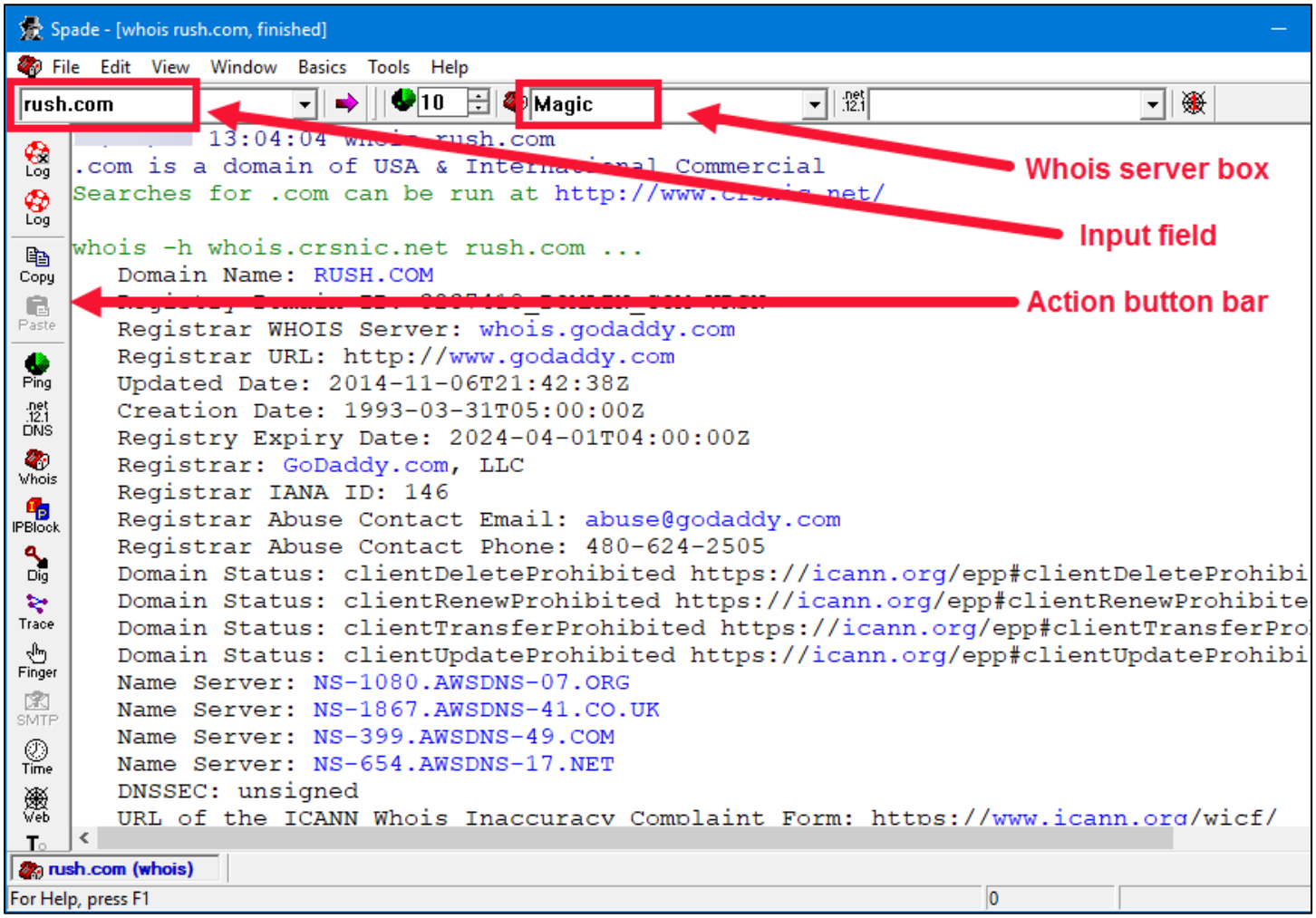
Sam Spade is not the best Whois program on the market. There is no single program that is good enough to be the "only" one. However, **Sam Spade** is a great one-stop shop, at least for a preliminary look, and can poll for information in ways that no other free resources can.

In our scenario, we have found out that the website of **www.rush.com** has a disparaging comment on its comments page about the Rush R40 World Tour. We want to find out who posted this comment, but we will need to contact the web site's owner to ask them for their connection logs.

But who is that? How do we find out? What if they don't want to co-operate and we need to find out where (physically) the website files are located?

Exercise – Part 1 Questions

1. In the upper left-hand search field, type in **`rush.com.`** For now, we will leave the Whois server field on **Magic**. **Magic** is not the last word in searching, but it does try to bring back quick wins. When not successful, it still gives us data that might further our investigation. The action buttons are probably on the left edge of your program window. You can drag and drop them to the top of the screen, if that is your preference. Click on the **Whois** icon. You will see some information appear.

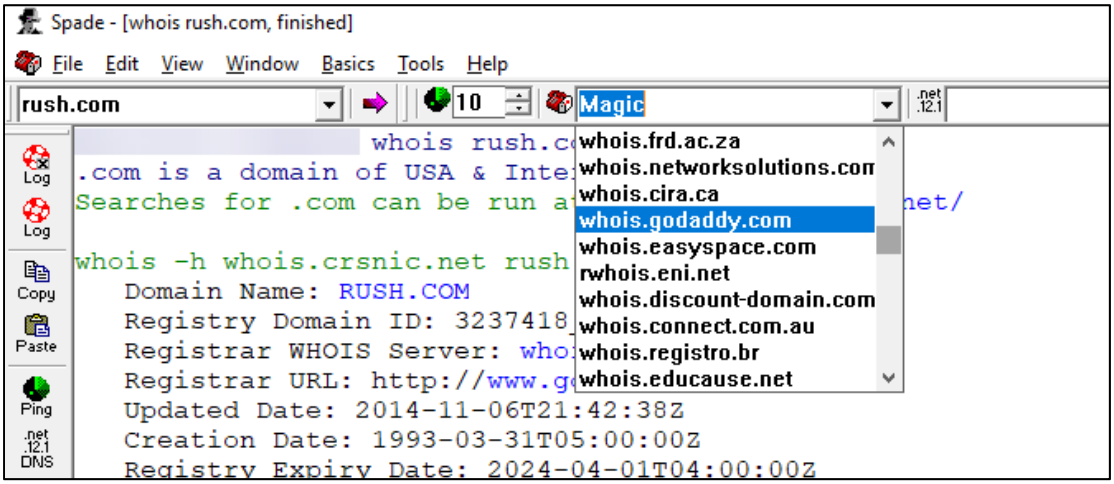


This is certainly not all that is available, nor is this indicative of the information you will always get. But we can extract clues to further our investigation. We already know when the domain name was created, which may be an important part of your investigation.

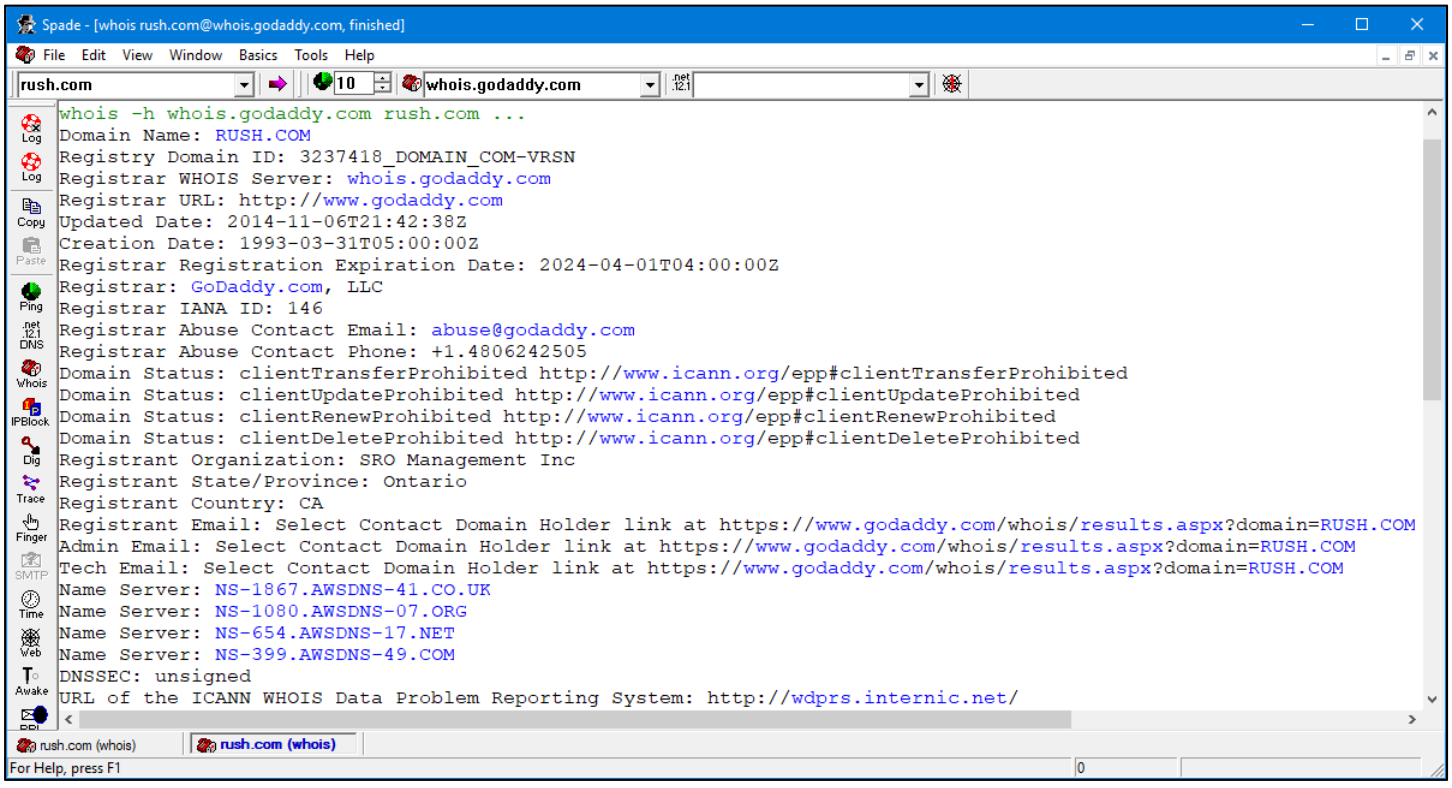
a. What is the **Registrar WHOIS Server**?

_____

2. Noting that **whois.godaddy.com** is listed as the **Registrar WHOIS Server**, and using the Whois server drop-down menu, find **whois.godaddy.com** and click on it.



3. It is extremely important to note that in **Sam Spade**, typing a search entry and pressing **Enter** is not the same as typing a search term and clicking an action button. Both return information, but it may be different. For example, clicking the **Whois** button at this point will return the information below, while merely pressing **Enter** would not get you the same data. It is always smart to do both in any situation.



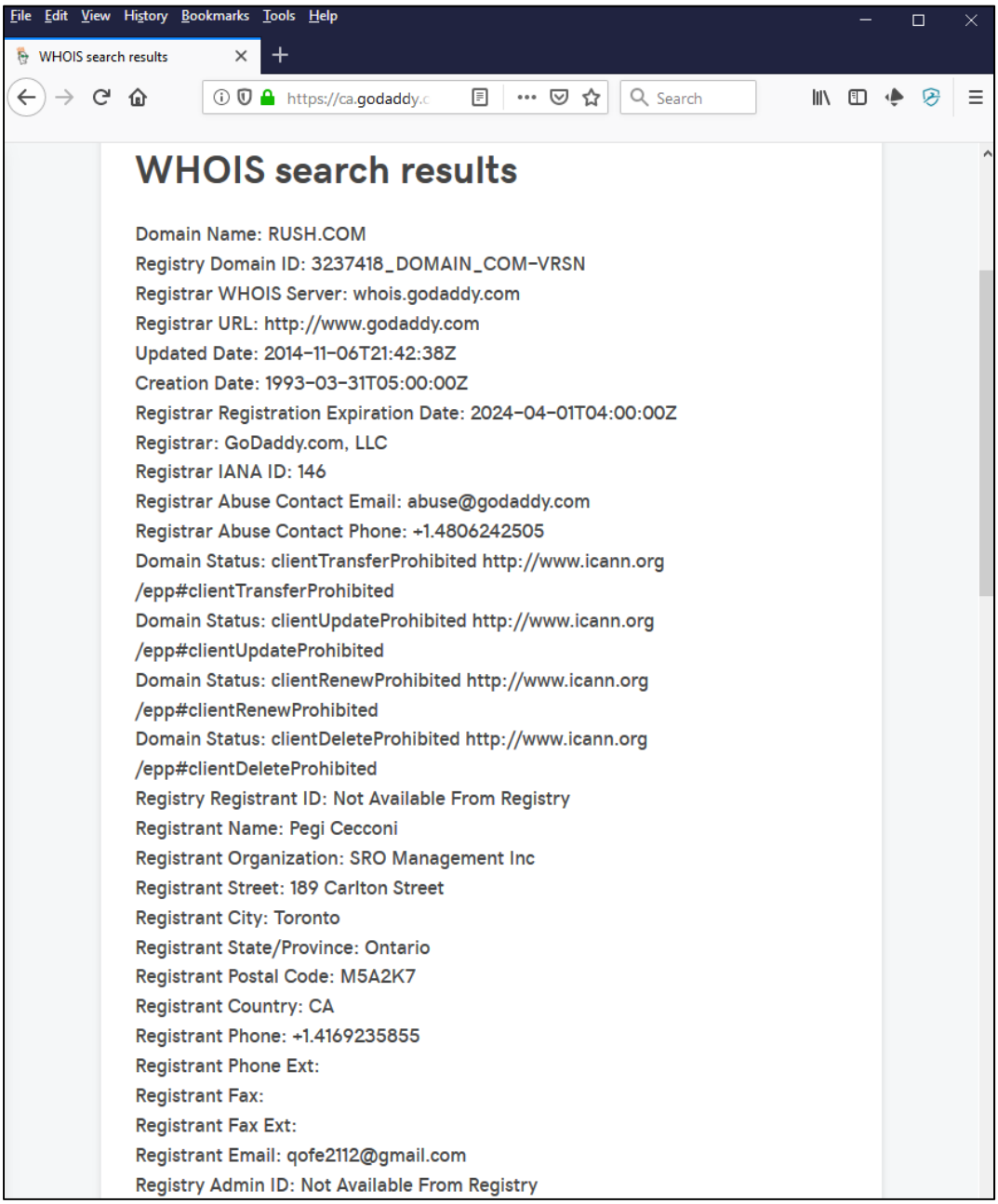
   a. What is the **Registrant Organization** name?

   _____

b. What is the **Registrant Email** listed?

_____

4. We don't actually see the **Registrant Email**, but we do see a URL that may help us gather further information. Highlight the URL from the **Registrant Email** field and press **Ctrl+C** to copy it, then paste it into your browser and review the results. (You may be asked to complete a reCaptcha verification)



a. What is the name, address, telephone, and email address of the **Registrant**?

_____

_____

Bonus Question

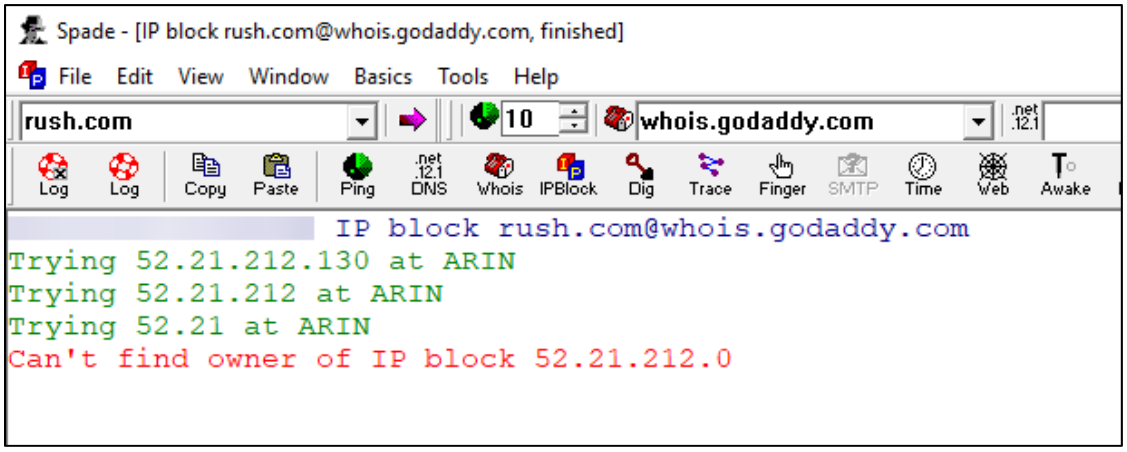
What is the meaning of the email address username? (This will take some research!)

_____

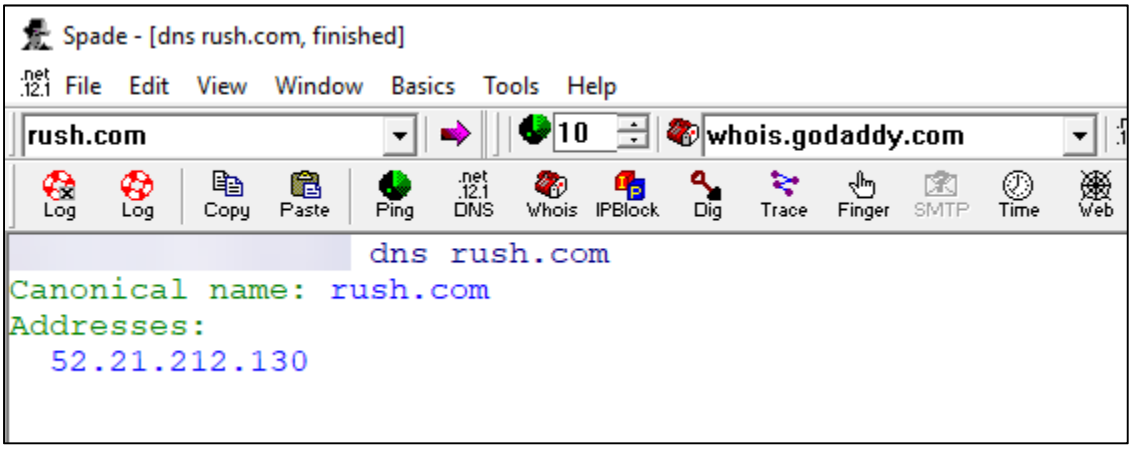
_____

We have now determined who to contact in order to get the information we seek. Or do we? It is possible that "**Pegi Cecconi**" hasn't got a clue how to get the data that we need. She was merely the person that bought the domain name of **rush.com** from GoDaddy back in 1993. An analysis of the website might determine who built the site, and that may be another approach. Here though, we want to determine where the website is hosted, as our plan is to approach them to take down the site or produce a copy of it.

5. In order to take the next step of finding where the site "lives", we need the **IP address** of the domain name. We can do this with **Sam Spade**. Back in the program, we will click on the **IPBlock** action button.



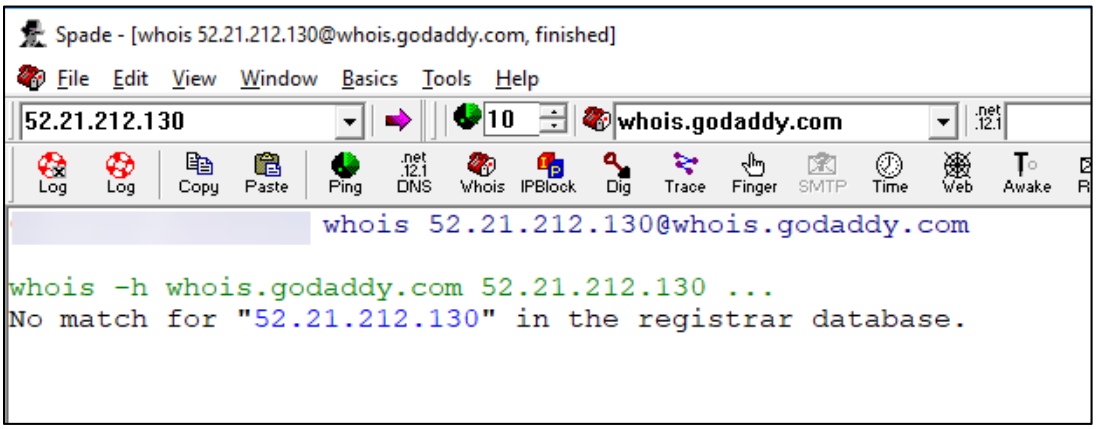
6. Although clicking the **IPBlock** action button was unsuccessful at finding an owner of the block, it did produce an **IP address** for the block. Another way to do this is to click on the **DNS** button, which will perform a **reverse DNS** on whatever is in the search field.



a. What is the **IP address** that corresponds with **rush.com**?

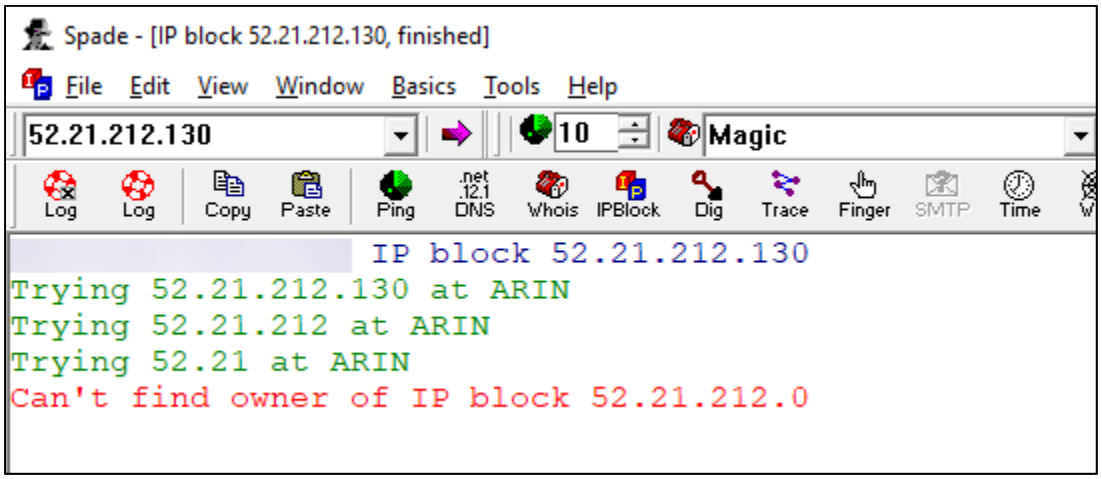
_____

7. In **Sam Spade**, if any data returned regarding your query is in blue text, it is clickable. In other words, clicking on it will place it in the search field. Click on the blue text that is the **IP address**, then click the **Whois** button.



a. Was the search successful? _____

8. We don't really know which server to go to at this point, so let's set the **Whois server** search box back to **Magic** and click on **Whois** again.



a. Was the search successful this time? _____

b. What server is the program trying to get information from? _____

9. Change the **Whois** server to **whois.arin.net** and click the **Whois** button.

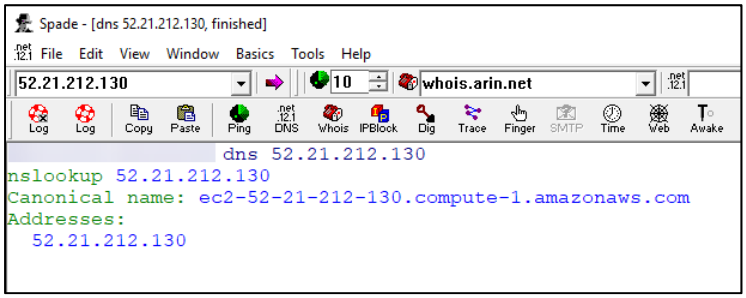
a. What is the IP range that the IP address comes from?

_____

b. Who holds the lease on that IP range?

_____

10. Now that we know the company, how about also identifying the actual computer? Click on the **DNS** button.



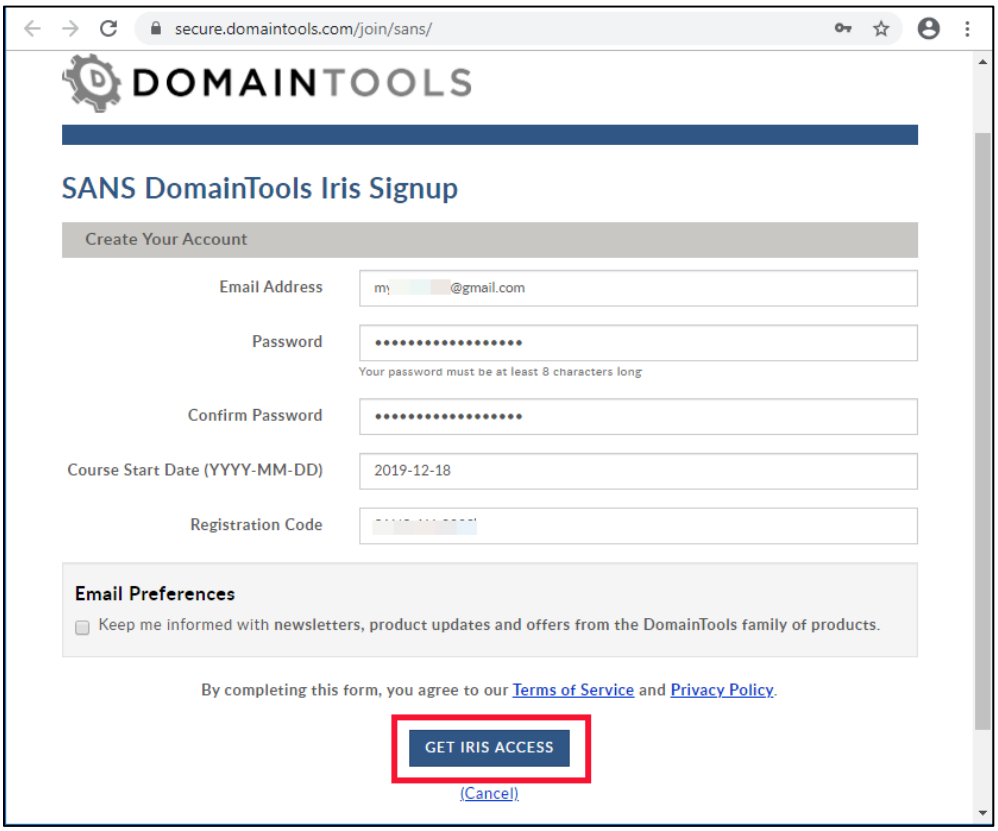
a. What is the name of the computer that the website sits on?

_____

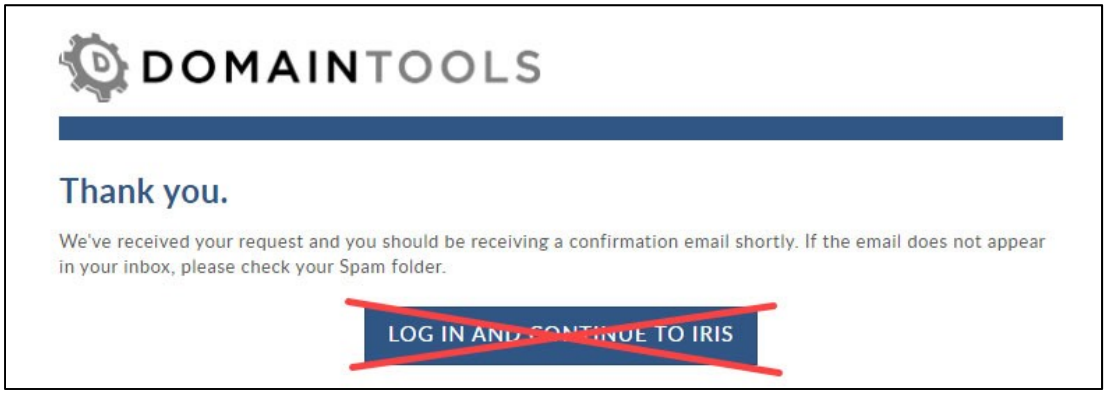
Exercise Preparation – Part 2

1. Although **Sam Spade** is a very capable tool, it always pays to double check your work with another source, both if you did not get the information you were hoping to find, or to verify the information you received.

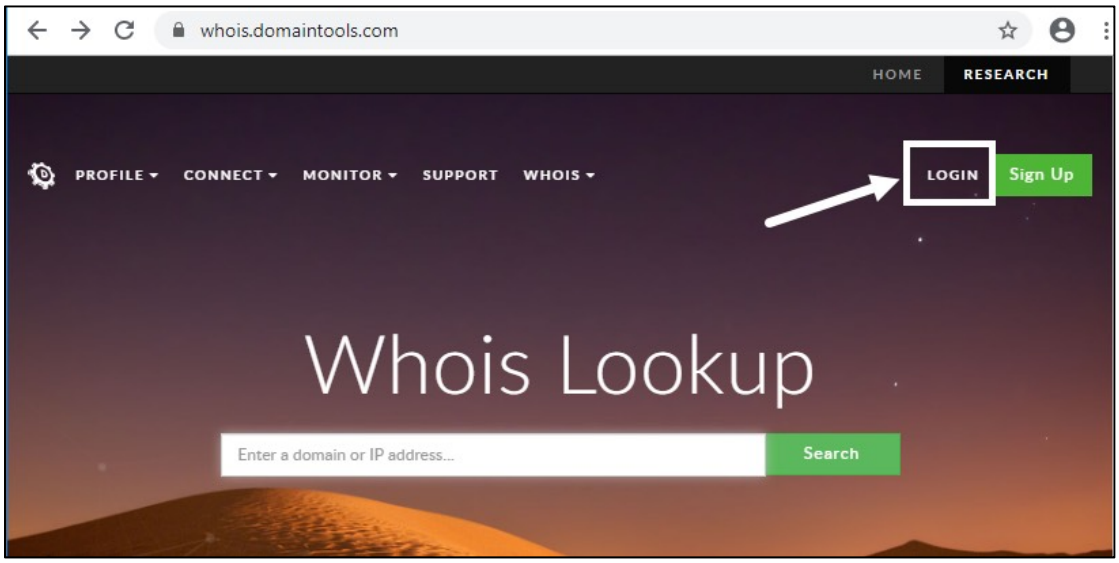
2. Open a browser and go to **https://secure.domaintools.com/join/sans/** Fill in the information to set up an account. You will receive a Registration Code from your instructor, or in the case of OnDemand, your SME. Once done, click on **GET IRIS ACCESS**.



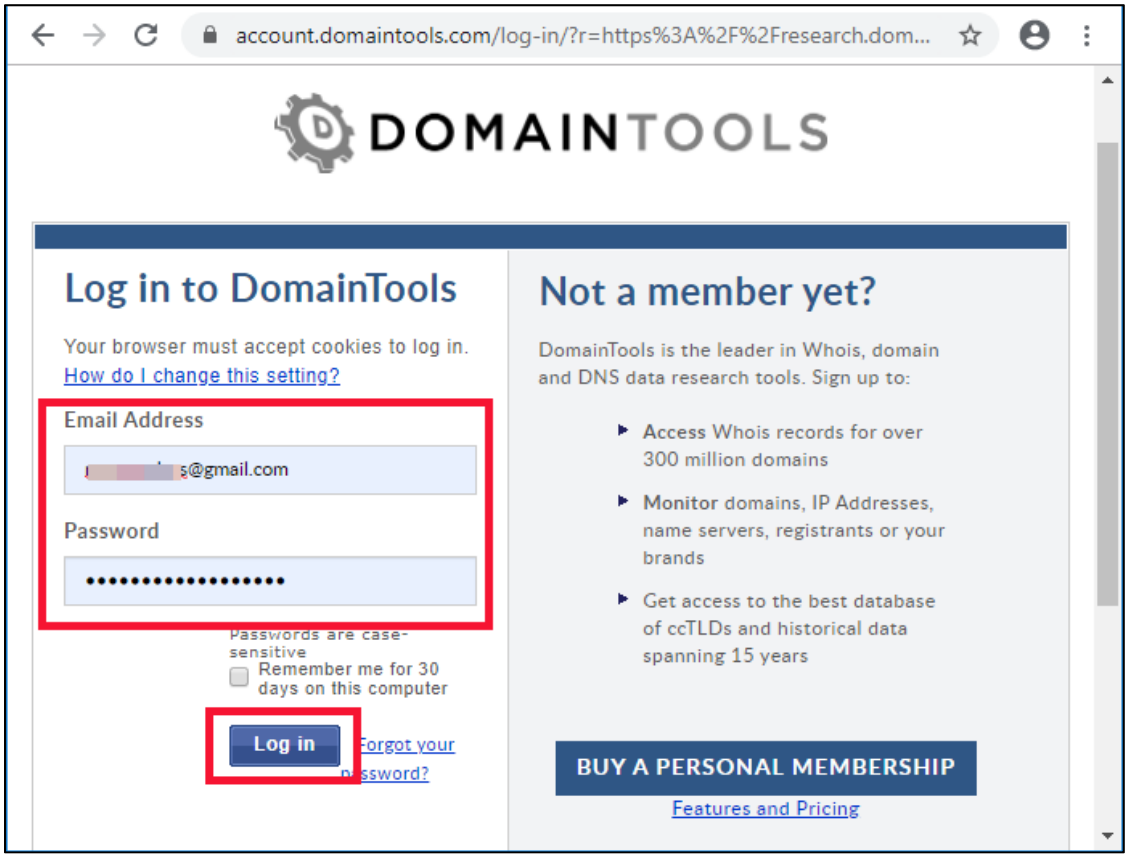
3. You will be taken to the confirmation page, however we will **NOT** click on **LOG IN AND CONTINUE TO IRIS**.



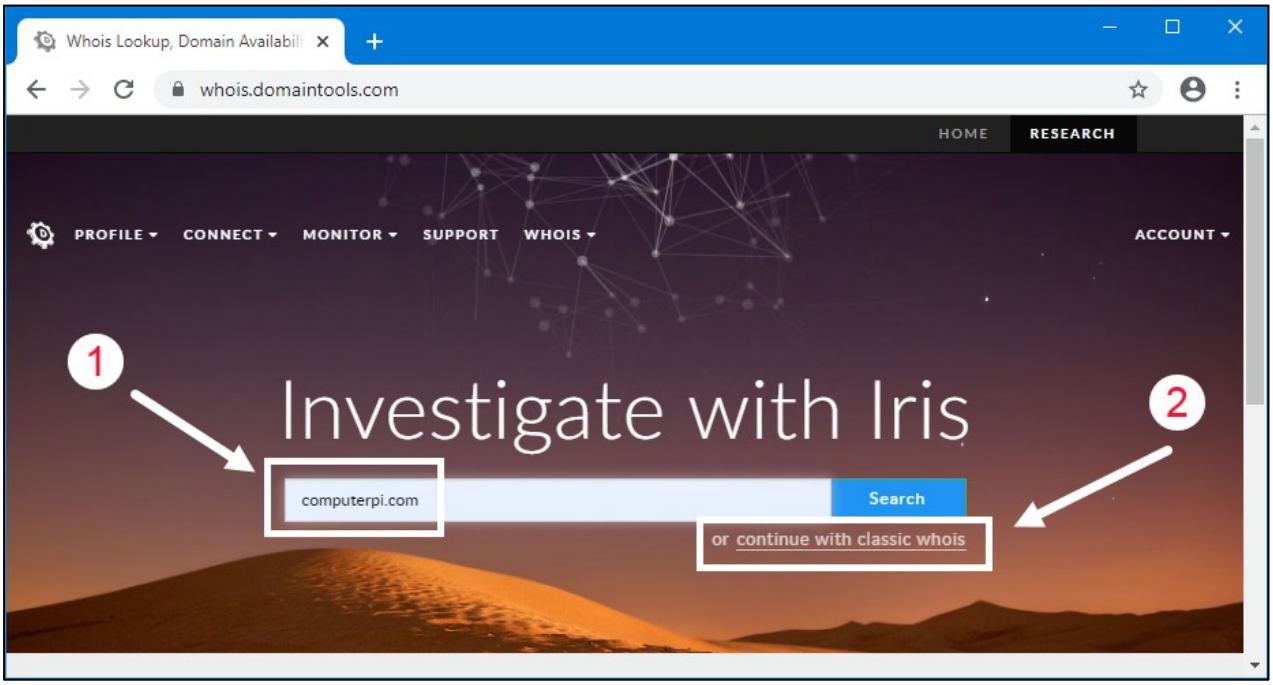
4. Instead, we will navigate to **whois.domaintools.com**. Once there, click on **LOGIN**.



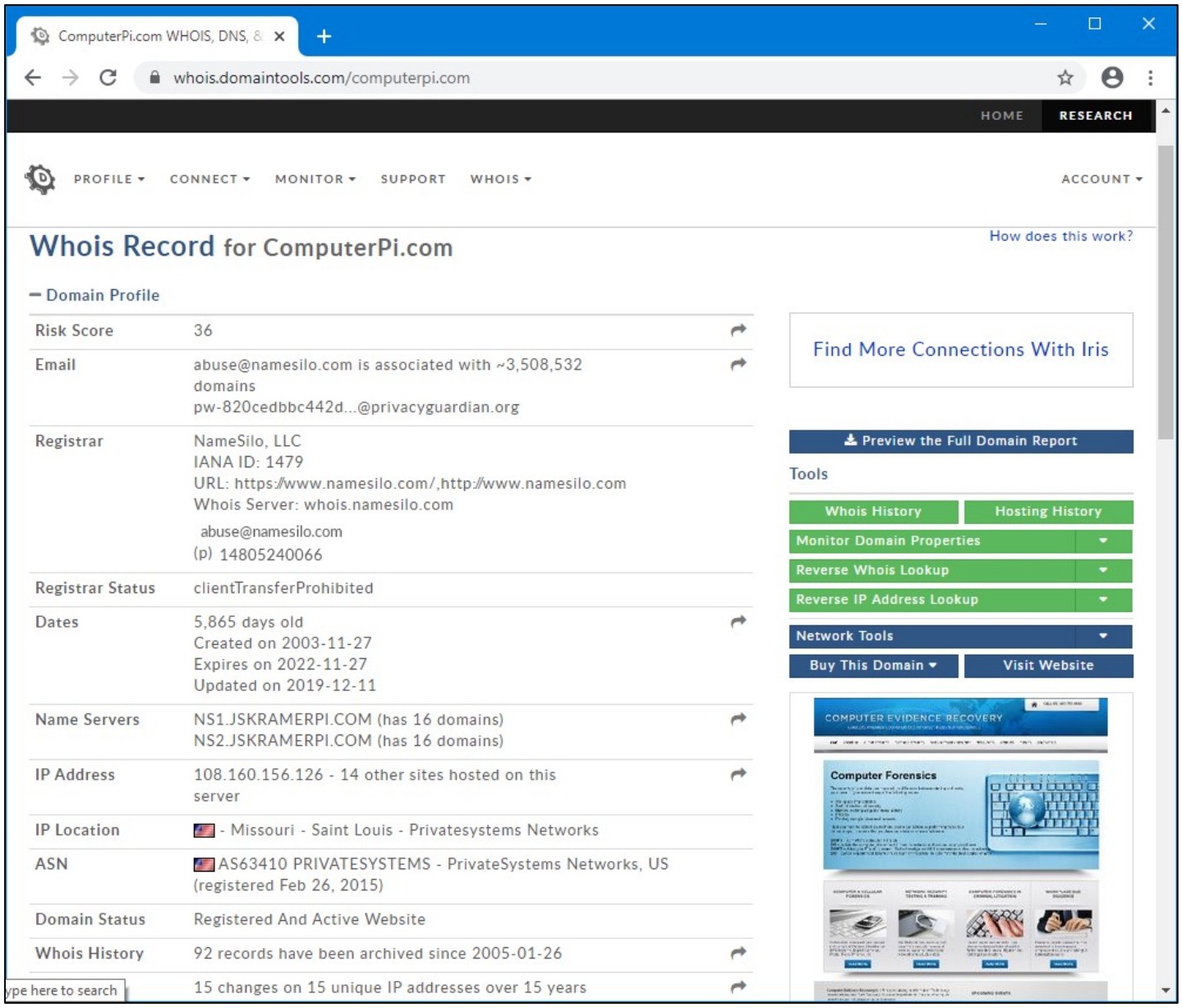
5. The Login page will open. Enter the email address and password you used to set up the account, and then click **Log in**.



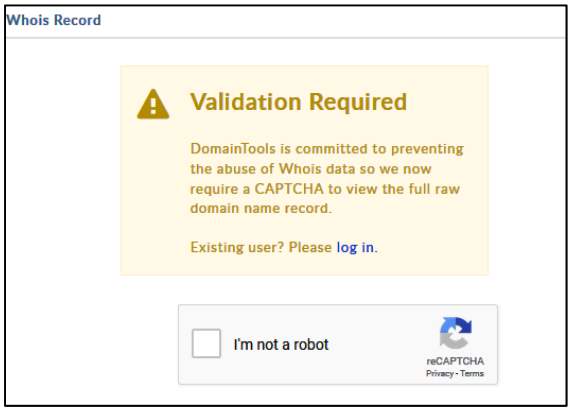
6. The **Investigate with Iris** page will appear. We will not be "investigating with Iris" in this exercise, but it is strongly recommended that you come back to this later and do so, as the features within Iris are incredible. Instead, enter **computerpi.com** in the search bar, and click on **continue with classic whois**.



7. The **Whois Record** appears for **computerpi.com**.



8. If you see a "**Validation Required**" box towards the bottom of the page, fill out the reCAPTCHA verification, and you will then see the full information.



Exercise – Part 2 Questions

1. Spend a bit of time looking around the **DomainTools** web page and answering the following questions.

   a. What country is the registrant from?

   _____

   b. When was the domain name created?

   _____

   c. Who is the contact for the website?

   _____

   d. Where does it appear the website is hosted (physically)?

   _____

   e. What is the IP address of the website?

   _____

2. You will see that there is even a screenshot of the home page. **DomainTools** provides a great deal of other information besides the regular **Whois** data. Scrolling down the page, you also see further information, such as **Hosting History** and other history. For a fee, **DomainTools** can tell you everybody that was ever listed in the **Whois** information, even if they have since been removed.

   a. How many websites are registered on this IP address?

   _____

   b. What is the server type?

   _____

While we are using this information to further our investigation, know that much of this same information is very important to an adversary, who uses services like this to gather intelligence on their targets. Let's move on to the actual **Whois** data that **DomainTools** is reporting.

Scrolling to the end of the page, we see the **Whois** record. Reading through, it is clear the site registrant has used a **Privacy Register** to hide their contact details.

3. Many people give up here. Don't take for granted that you are as far as you can go. Understand what you are looking at. Remember that we were looking at domain name information. But what other information can we gather? You could do things like take the IP address you discovered, and place that in the **DomainTools Whois Lookup** field at the top of the web page and see what information will be found there.

   a. What is the **Organization Name** that handles privacy for this host?

      _____

Bonus Questions

1. Can we narrow down the physical location of the server to a name/address/telephone? If yes, what is it?

   _____

2. What is name/address/telephone for the **Admin Name** at **cispi.net**?

   _____

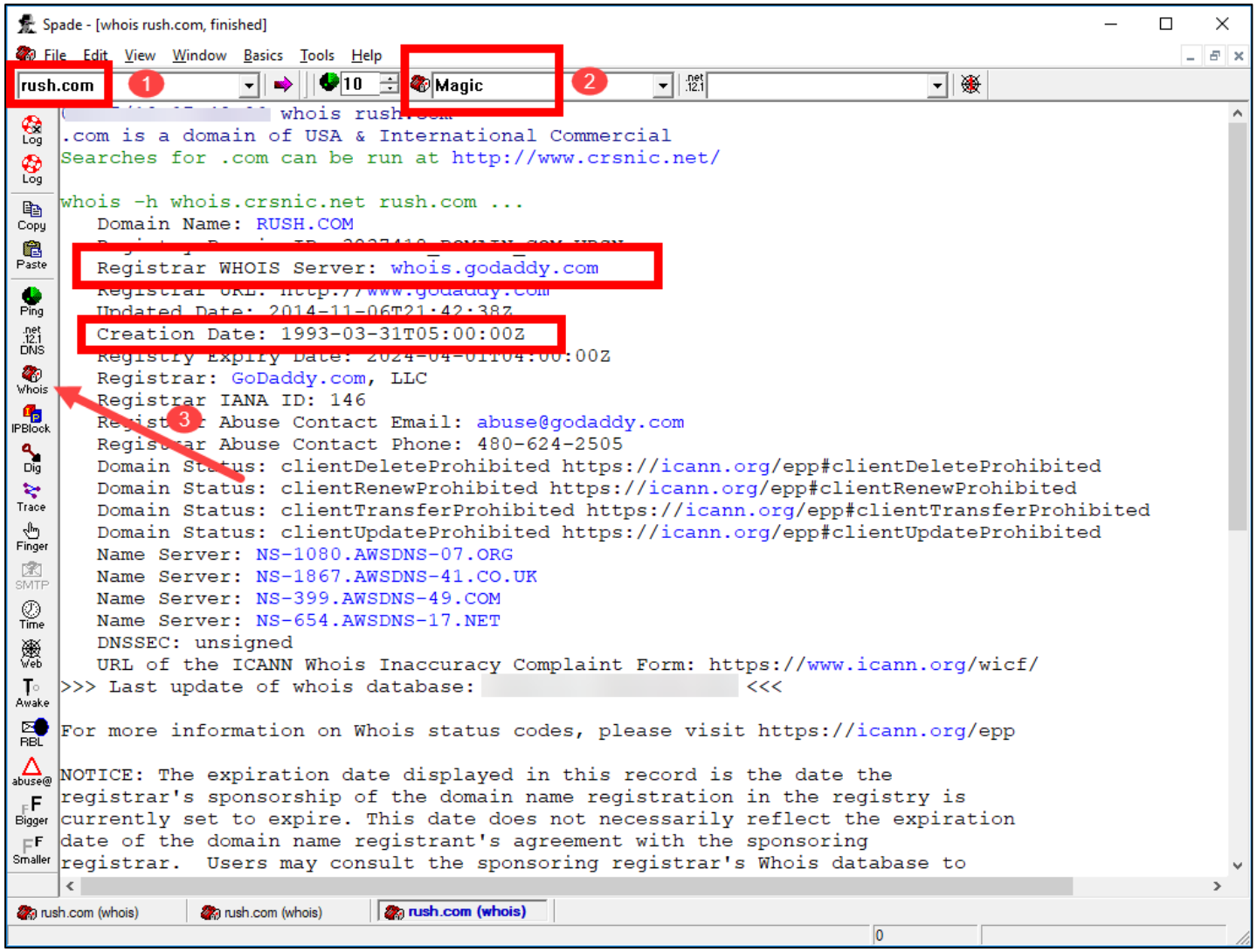
   _____

3. What is the **IP address** of **cispi.net**?

   _____

Exercise - Part 1 Questions with Step-by-Step

1. In the upper left-hand search field, type in **`rush.com.`** For now, we will leave the **Whois** server field on **Magic**. **Magic** is not the last word in searching, but it does try to bring back quick wins. When not successful, it still gives us data that might further our investigation. The action buttons are probably on the left edge of your program window. You can drag and drop it to the top of the screen, if that is your preference. Click on the **Whois** icon. You will see some information appear.

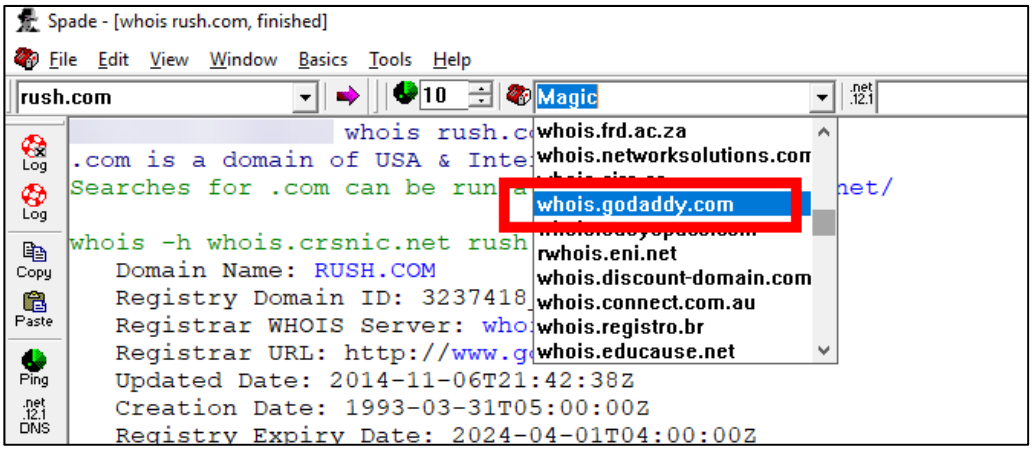


This is certainly not all that is available, nor is this indicative of the information you will always get. But we can extract clues to further our investigation. We already know when the domain name was created, which may be an important part of your investigation.

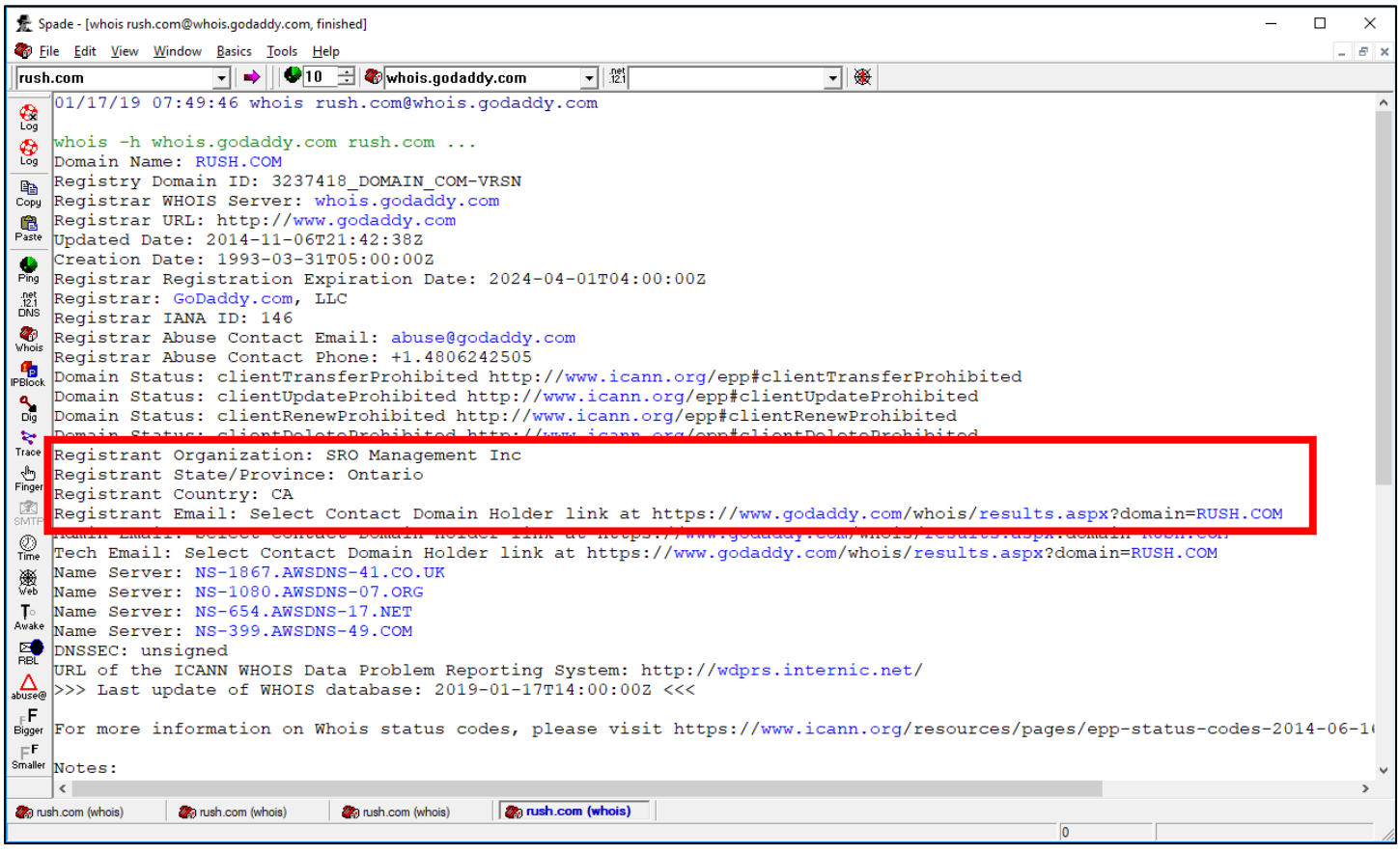
a. What is the **Registrar WHOIS Server**?

Taken from the first boxed line above, we see whois.godaddy.com

2. Noting that **whois.godaddy.com** is listed as the **Registrar WHOIS Server**, and using the **Whois** server drop-down menu, find **whois.godaddy.com** and click on it.



3. It is extremely important to note that in **Sam Spade**, typing a search entry and pressing **Enter** is not the same as typing a search term and clicking an action button. Both return information, but it may be different. For example, clicking the **Whois** button at this point will return the information below, while merely pressing **Enter** would not get you the same data. It is always smart to do both in any situation.



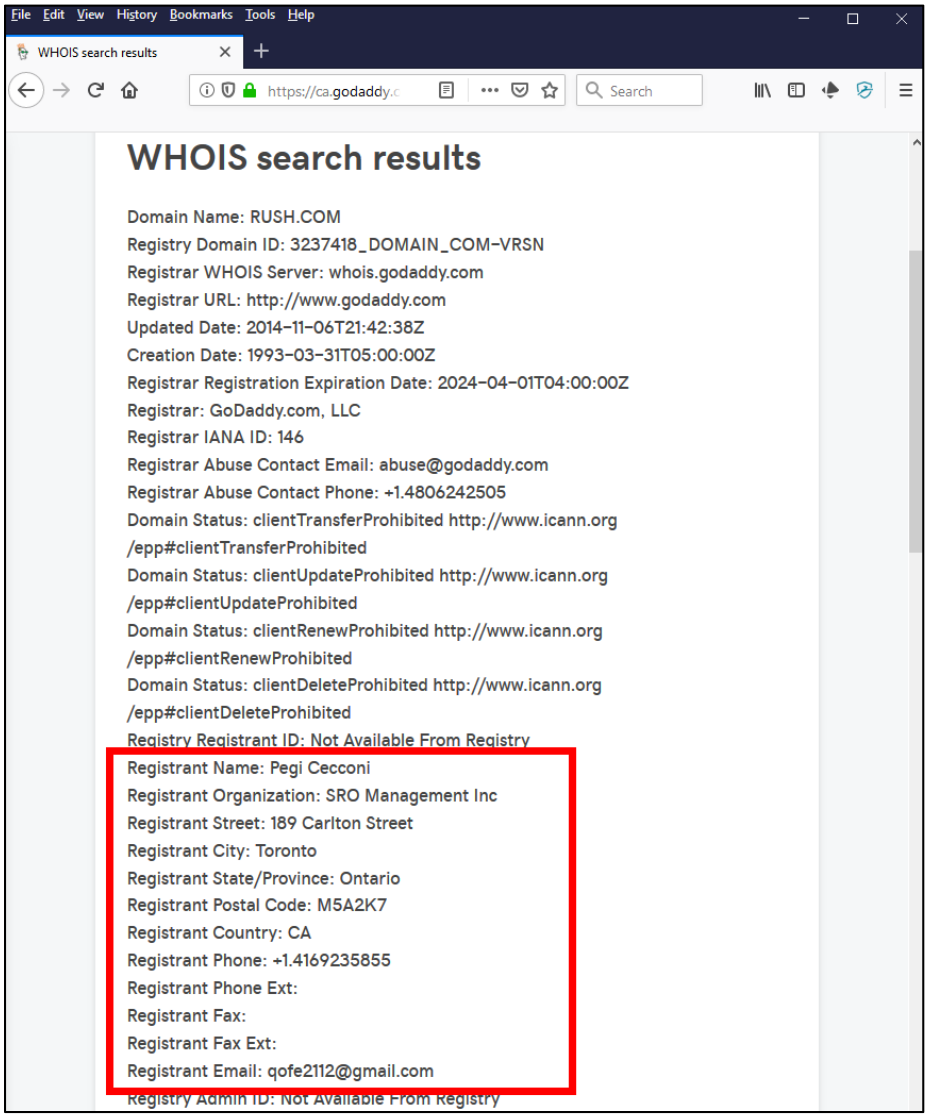
a. What is the **Registrant Organization** name?

Taken from the top line in the boxed area above, we see SRO Management Inc

b. What is the **Registrant Email** listed?

There wasn't one where it was expected. Rather, there was a link that appears as though it may provide that info.

4. We didn't see the **Registrant Email** in the last step, but we did see a URL that could help us gather further information. Highlight the URL from the **Registrant Email** field and press **Ctrl+C** to copy it, then paste it into your browser and review the results. (You may be asked to complete a reCaptcha verification)



a. What is the name, address, telephone, and email address of the Registrant?

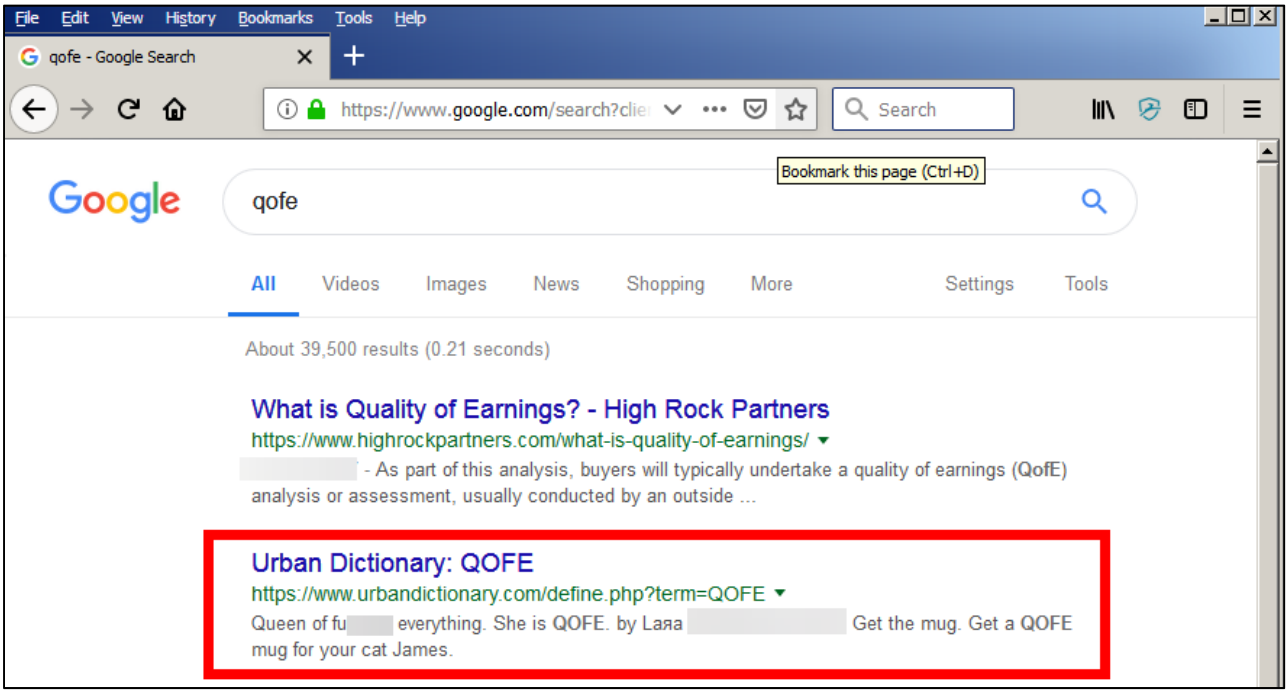
Pegi Cecconi
SRO Management Inc
189 Carlton Street
Toronto, Ontario M5A 2K7
416 923 5855
qofe2112@gmail.com

Bonus Question

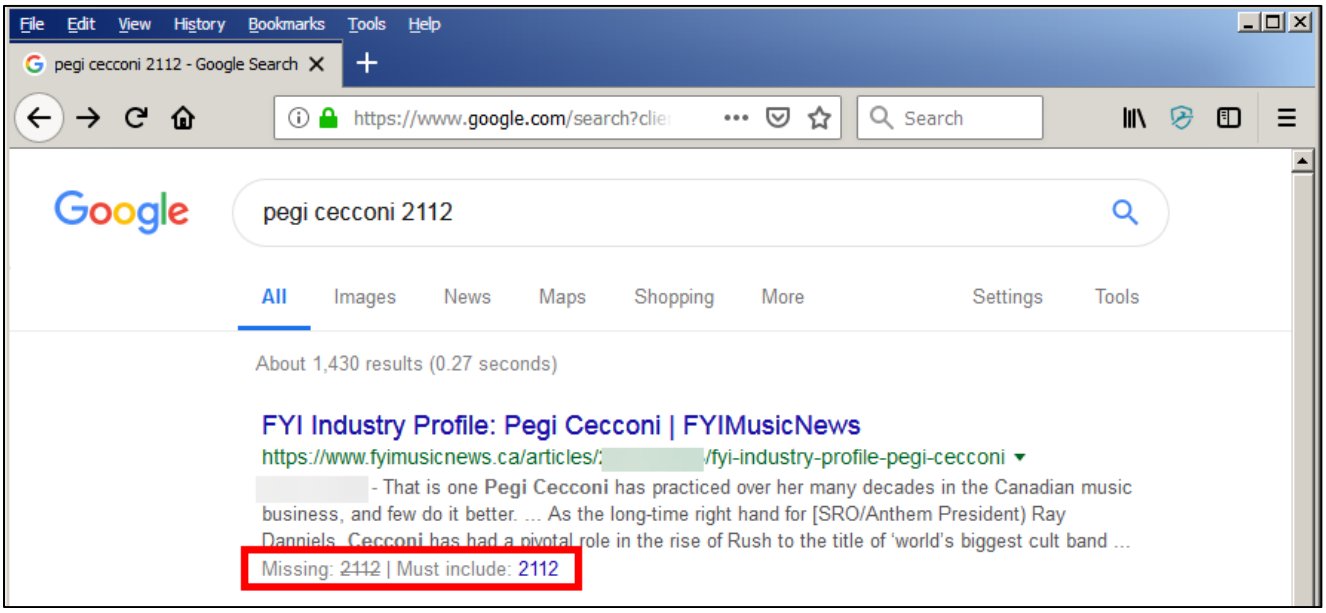
What is the meaning of the email address username? (This will take some research!)

Queen Of F{}^$#@ Everything for the rock group Rush

Googling the email address username of qofe2112 merely brought us back to Pegi Cecconi from a number of sites. If you break it up into qofe and 2112, this might be helpful. Google qofe and you quickly learn from Urban Dictionary that qofe stands for Queen Of F{}^$#@ Everything.



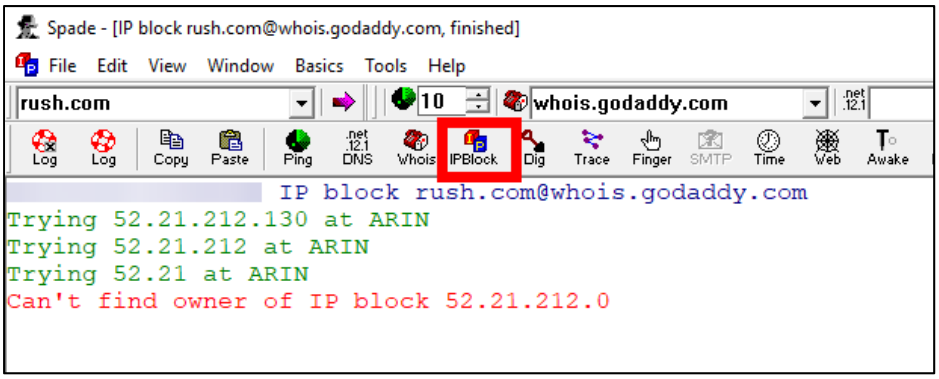
You should also have established by now that Pegi has an association with the rock group Rush. Googling Pegi Cecconi and 2112 would show the Rush association.



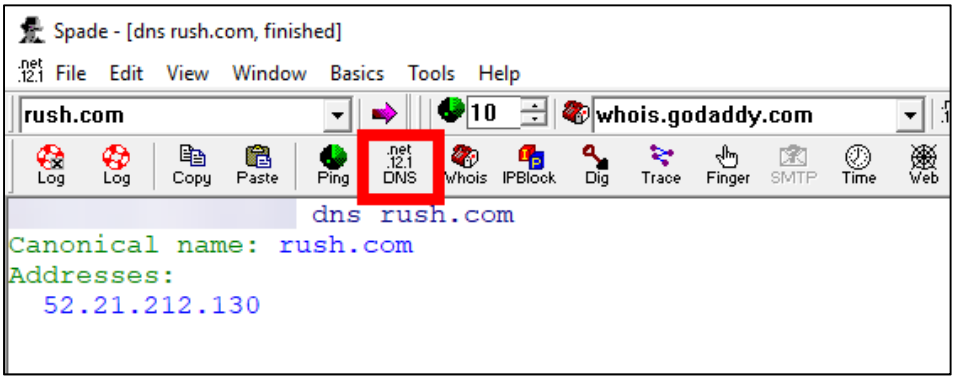
You can see the Rush reference, but that still doesn't explain 2112. At the bottom of the Google search hit, you see the text in the red box stating that the search hit doesn't have 2112 in it. You also have an option to say that the search hit Must include 2112. Click on the 2112 link. Click on the Images heading. You will see a record album from Rush named 2112. One of their most famous albums.

We have now determined who to contact in order to get the information we seek. Or do we? It is possible that "Pegi Cecconi" hasn't got a clue how to get the data that we need? She was merely the person that bought the domain name of **rush.com** from GoDaddy back in 1993. An analysis of the website might determine who built the site, and that may be another approach. Here though, we want to determine where the website is hosted, as our plan is to approach them to take down the site or produce a copy of it.

5. In order to take the next step of finding where the site "lives", we need the **IP address** of the domain name. We can do this with **Sam Spade**. Back in the program, we will click on the **IPBlock** action button.



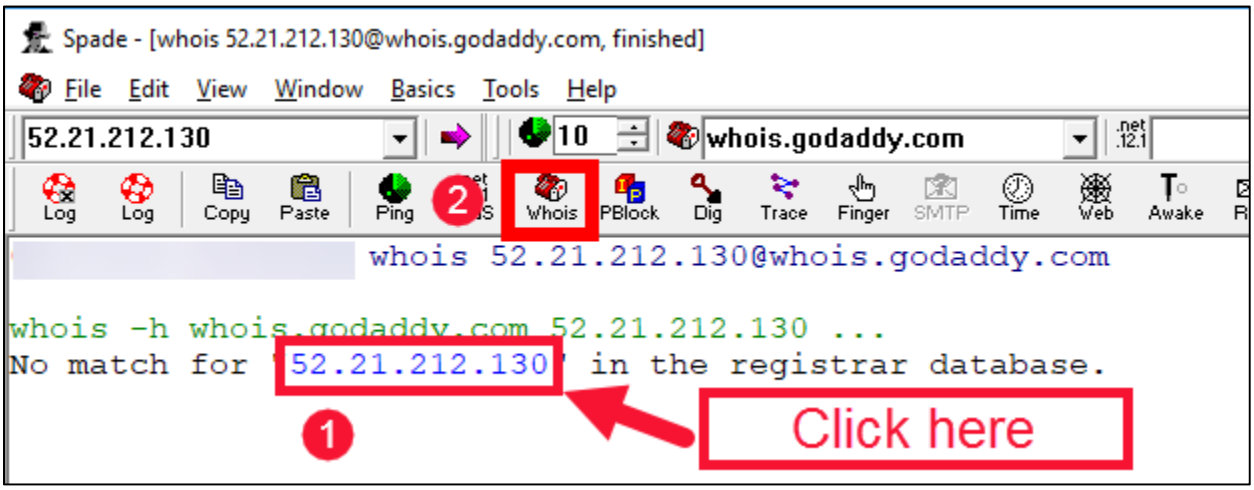
6. Although clicking the **IPBlock** action button was unsuccessful at finding an owner of the block, it did produce an **IP address** for the domain. Another way to do this is to click on the **DNS** button, which will perform a **reverse DNS** on whatever is in the search field.



a. What is the **IP address** that corresponds with rush.com?

52.21.212.130

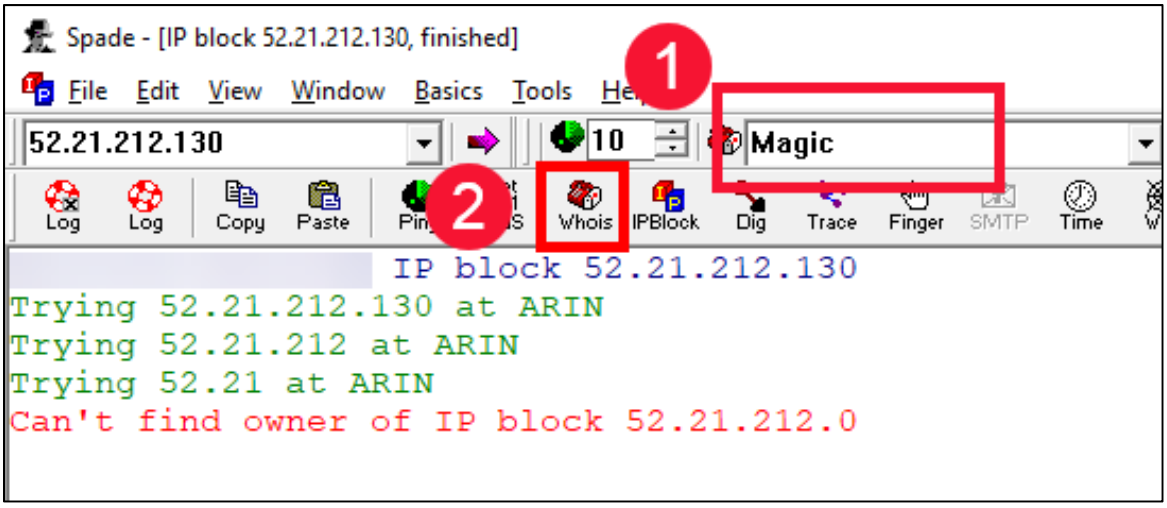
7. In **Sam Spade**, if any data returned regarding your query is in blue text, it is clickable. In other words, clicking on it will place it in the search field. Click on the blue text that is the **IP address**, then click the **Whois** button.



a. Was the search successful in finding out where the computer hosting the website is?

No

8. We don't really know which server to go to at this point, so let's set the **Whois server** back to **Magic** and click on **Whois** again.



a. Was the search successful this time?

No

b. What server is the program trying to get information from? **(Where is Sam Spade "Trying" to find the IP address at?)**

ARIN (American Registry of Internet Numbers)

9. Change the **Whois** server to **whois.arin.net** and click the **Whois** button.

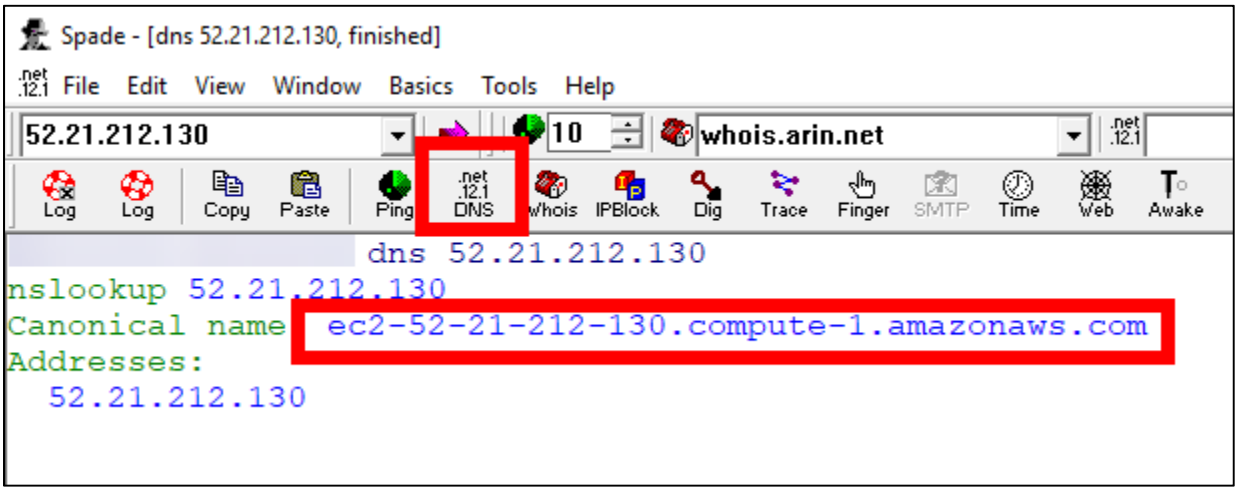
a. What is the IP range that the IP address comes from?

52.0.0.0 – 52.31.255.255 (As seen in top box of screenshot)

b. Who holds the lease on that IP range?

Amazon Technologies Inc. (As seen in second box of screenshot)

10. Now that we know the company, how about also identifying the actual computer? Click on the **DNS** button.



a. What is the name of the computer that the website sits on?

Ec2-52-21-212-130.compute-1.amazonaws.com (Known as the Canonical name)

Exercise – Part 2 Questions Step-by-Step

1. Spend a bit of time looking around the **DomainTools** page and answering the following questions.

   a. What country is the registrant from?

      Appears to be US, but you cannot say for sure with the data available

   b. When was the domain name created? **Look further down the page for this data.**

      November 28, 2003

   c. Who is the contact for the website?

      NameSilo, LLC

   d. Where does it appear the website is hosted (physically)? **Look at IP Location**

      St. Louis, Missouri

   e. What is the IP address of the website? **See the IP Address field**

      108.160.156.126 **Given that this is live data from a live site, it can change without notice!**

2. You will see that there is even a screenshot of the home page. **DomainTools** provides a great deal of other information besides the regular **Whois** data. Scrolling down the page, you also see further information, such as **Hosting History** and other history. For a fee, **DomainTools** can tell you everybody that was ever listed in the **Whois** information, even if they have since been removed.

   a. How many websites are registered on this IP address? **Data is immediately after the IP address**

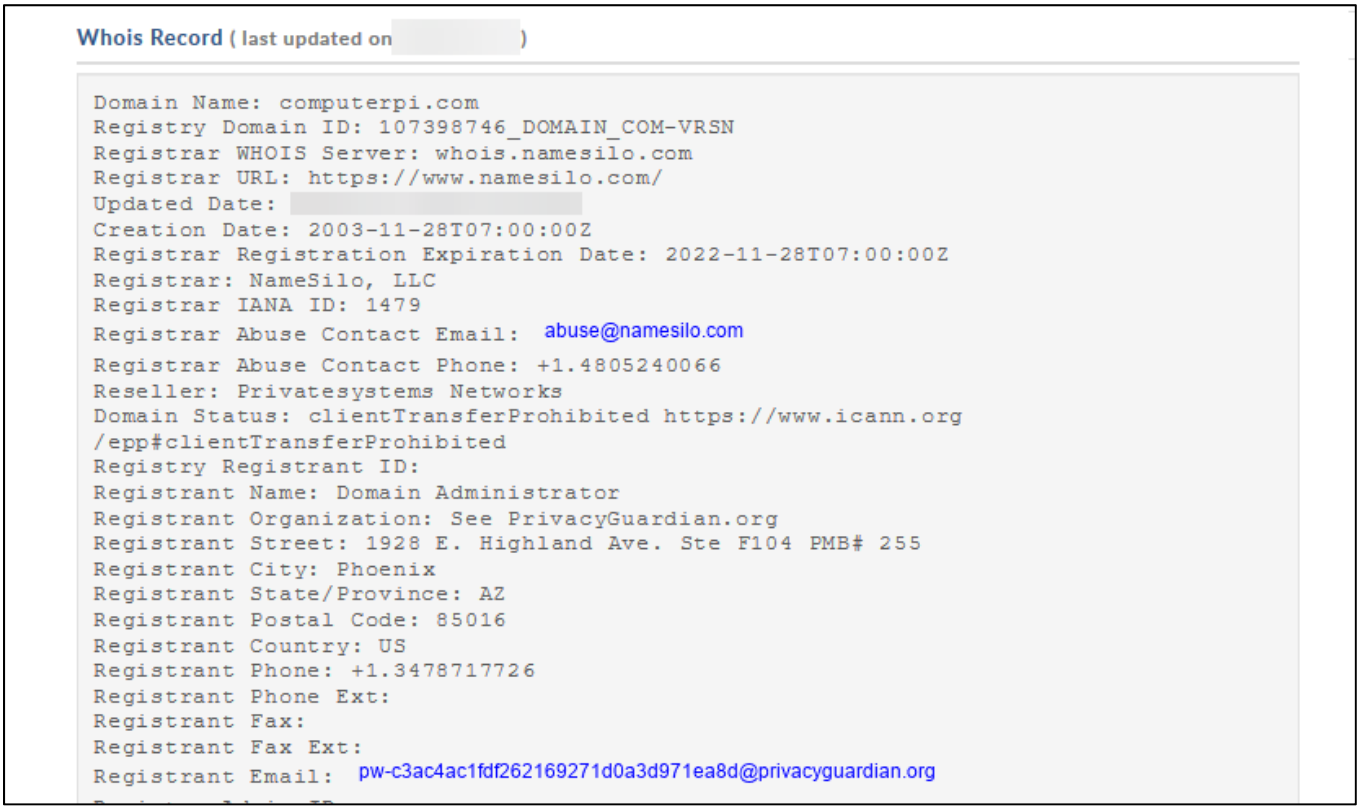
      14 currently (This could change, as this is live data)

   b. What is the server type?

      Apache/2

While we are using this information to further our investigation, know that much of this same information is very important to an adversary, who uses services like this to gather intelligence on their targets. Let's move on to the actual **Whois** data that **DomainTools** is reporting.

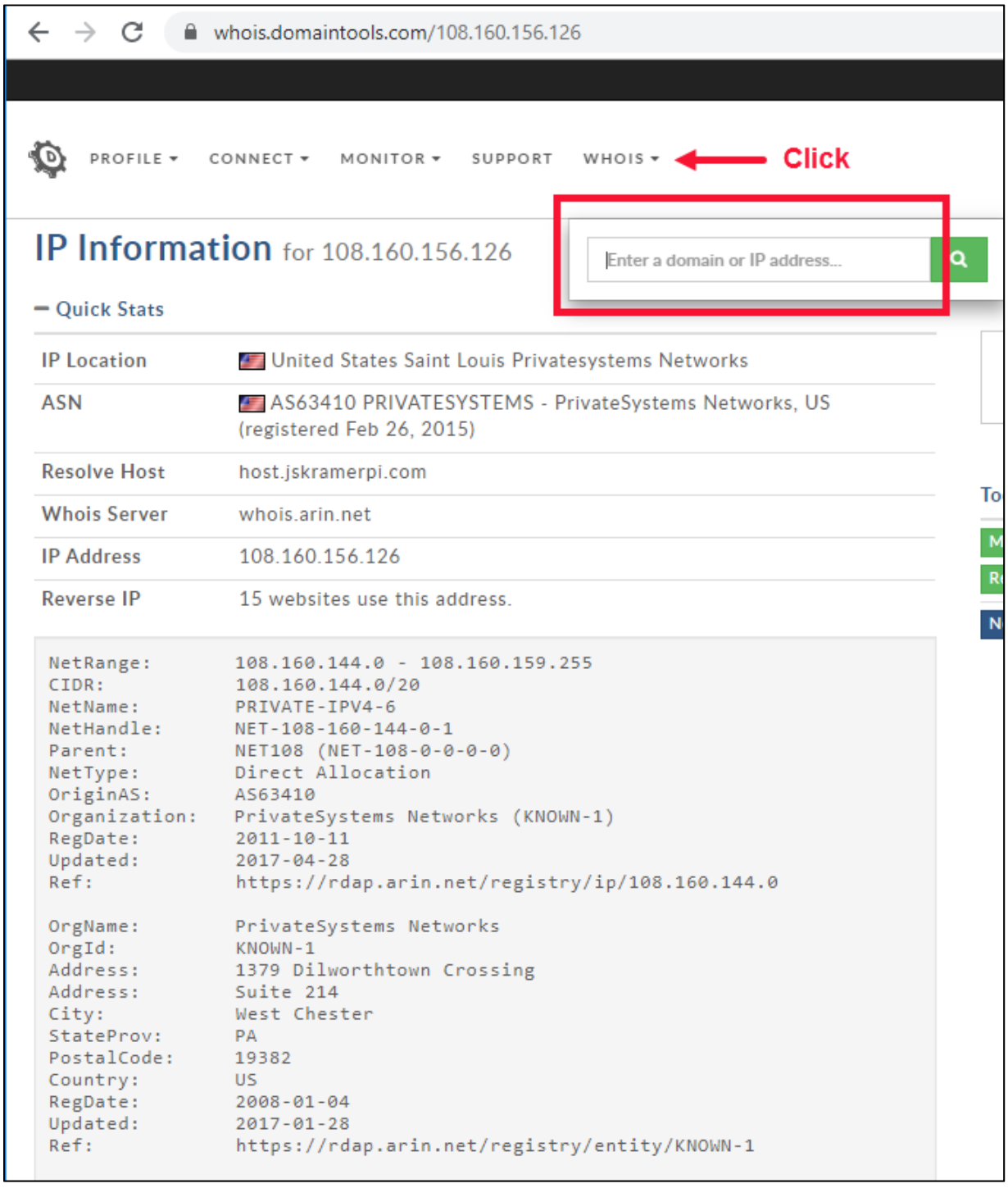
Scrolling to the end of the page, we see the **Whois** record. Reading through, it is clear the site registrant has used a **Privacy Register** to hide their contact details.



Whois Record (last updated on [redacted])

```
Domain Name: computerpi.com
Registry Domain ID: 107398746_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.namesilo.com
Registrar URL: https://www.namesilo.com/
Updated Date: [redacted]
Creation Date: 2003-11-28T07:00:00Z
Registrar Registration Expiration Date: 2022-11-28T07:00:00Z
Registrar: NameSilo, LLC
Registrar IANA ID: 1479
Registrar Abuse Contact Email: abuse@namesilo.com
Registrar Abuse Contact Phone: +1.4805240066
Reseller: Privatesystems Networks
Domain Status: clientTransferProhibited https://www.icann.org
/epp#clientTransferProhibited
Registry Registrant ID:
Registrant Name: Domain Administrator
Registrant Organization: See PrivacyGuardian.org
Registrant Street: 1928 E. Highland Ave. Ste F104 PMB# 255
Registrant City: Phoenix
Registrant State/Province: AZ
Registrant Postal Code: 85016
Registrant Country: US
Registrant Phone: +1.3478717726
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: pw-c3ac4ac1fdf262169271d0a3d971ea8d@privacyguardian.org
```

3. Many people give up here. Don't take for granted that you are as far as you can go. Understand what you are looking at. Remember that we were looking at domain name information. But what other information can we gather? You could do things like take the IP address you discovered, and place that in the **DomainTools Whois Lookup** field at the top of the web page and see what information will be found there.



a. What is the **Organization Name** that handles privacy for this host? **OrgName**

PrivateSystems Networks

Bonus Questions

1. Can we narrow down the physical location of the server to a name/address/telephone? If yes, what is it?

   Yes

Scroll to the bottom of the Whois info, to OrgName. Someone gave out a little too much data! Google the name Tierpoint Seattle, or the address for more information

TierPoint Colocation Data Center
140 4th Ave N., Suite 360
Seattle, Washington 98109

```
OrgName:        PrivateSystems Networks WA
OrgId:          PNW
Address:        Tierpoint Seattle c/o PrivateSystems Networks
Address:        140 4th Ave N. Suite 360
City:           Seattle
StateProv:      WA
PostalCode:     98109
Country:        US
RegDate:        2016-02-28
Updated:        2017-04-28
Ref:            https://rdap.arin.net/registry/entity/PNW
```

2. What is name/address/telephone for the **Admin Name** at **cispi.net**?

Using the whois.domaintools.com web site, do a Whois search on the domain name, then scroll through the returned information.

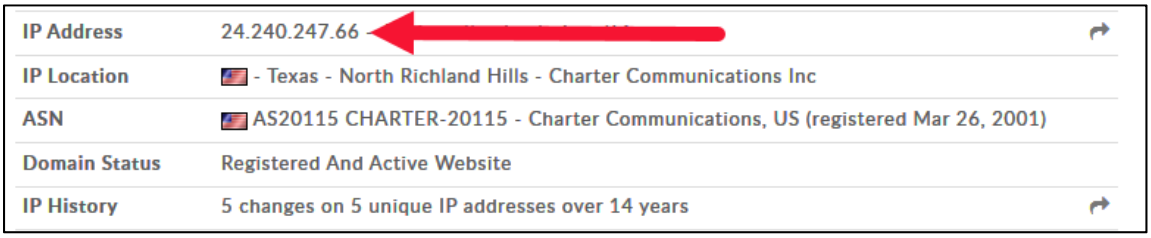
Brian Ingram
Consulting Investigation Services
PO Box 2097
Waxahachie, Texas 75168
972 937 3938

```
Registry Admin ID:
Admin Name: Ingram, Brian
Admin Organization: Consulting Investigation Services
Admin Street: PO Box 2097
Admin City: Waxahachie
Admin State/Province: TX
Admin Postal Code: 75168
Admin Country: US
Admin Phone: +1.9729373938
Admin Phone Ext:
Admin Fax: +1.9999999999
Admin Fax Ext:
Admin Email: info@cispi.net
```

3. What is the **IP address** of **cispi.net**?

Back up the same page as the above information we see the following IP:

24.240.247.66



Exercise—Key Takeaways

- A simple Whois search almost always contains less information than you can find by performing a more in-depth analysis.
- Old tools like Sam Spade can do things that no other tool can do.
- Whois information is public information (for the time being).
- Whois Registrars are mandated to provide a certain amount of information for free.
- DomainTools is one of the most robust tools for IP and domain information. The subscription version allows for a great deal of aggregation and historical information.

This page intentionally left blank.

Exercise 5.1B—Online Attribution – Archive.org

Background

Most everything today has an online component. Hardly a digital forensics investigation goes by that doesn't have evidence pointing off the storage device and into cyberspace. Emails, web history, addresses, URLs and IPs. What can we do to further our investigation?

As a last resort, we always have the option of subpoena or warrant, but given the inherent difficulties (not to mention cost) in pursuing either, this might not be the avenue we wish to pursue initially, if at all. Surely there must be a mechanism we can use in at least some cases to get us closer to our goal.

From online posts on social media, to comments on blogs or websites, people are "saying" more and more than ever before. In some cases, these comments cause real damage, and the perpetrators need to be tracked down.

Objectives

- Analyze historical website data using archive.org
- Use Search.org to identify the Custodian of Record for an ISP

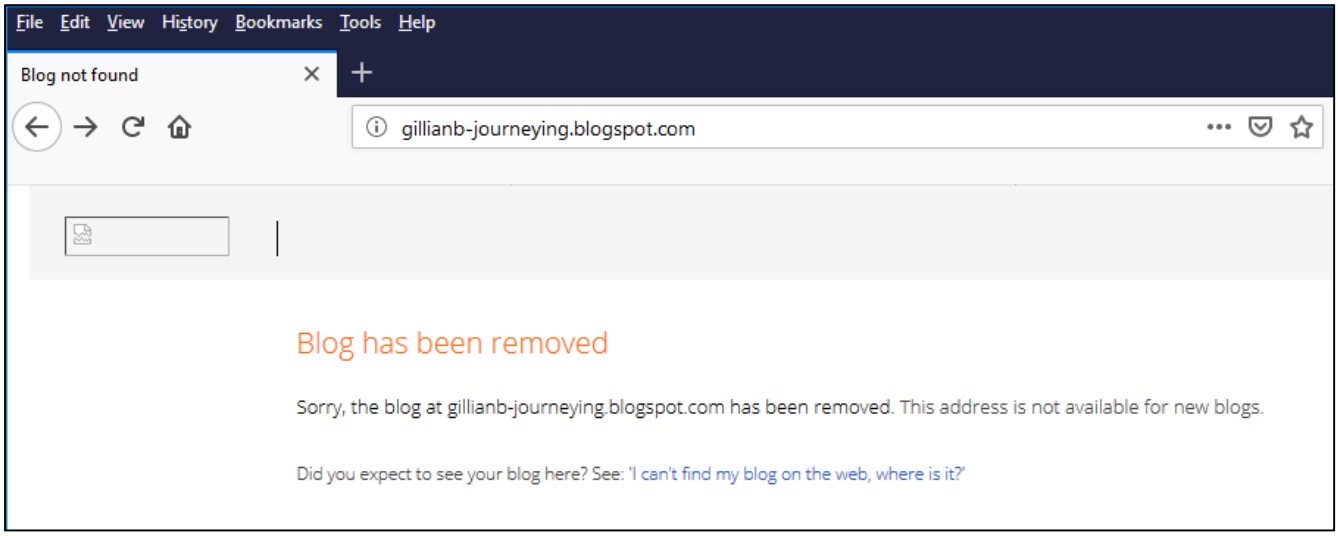
Exercise Preparation

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

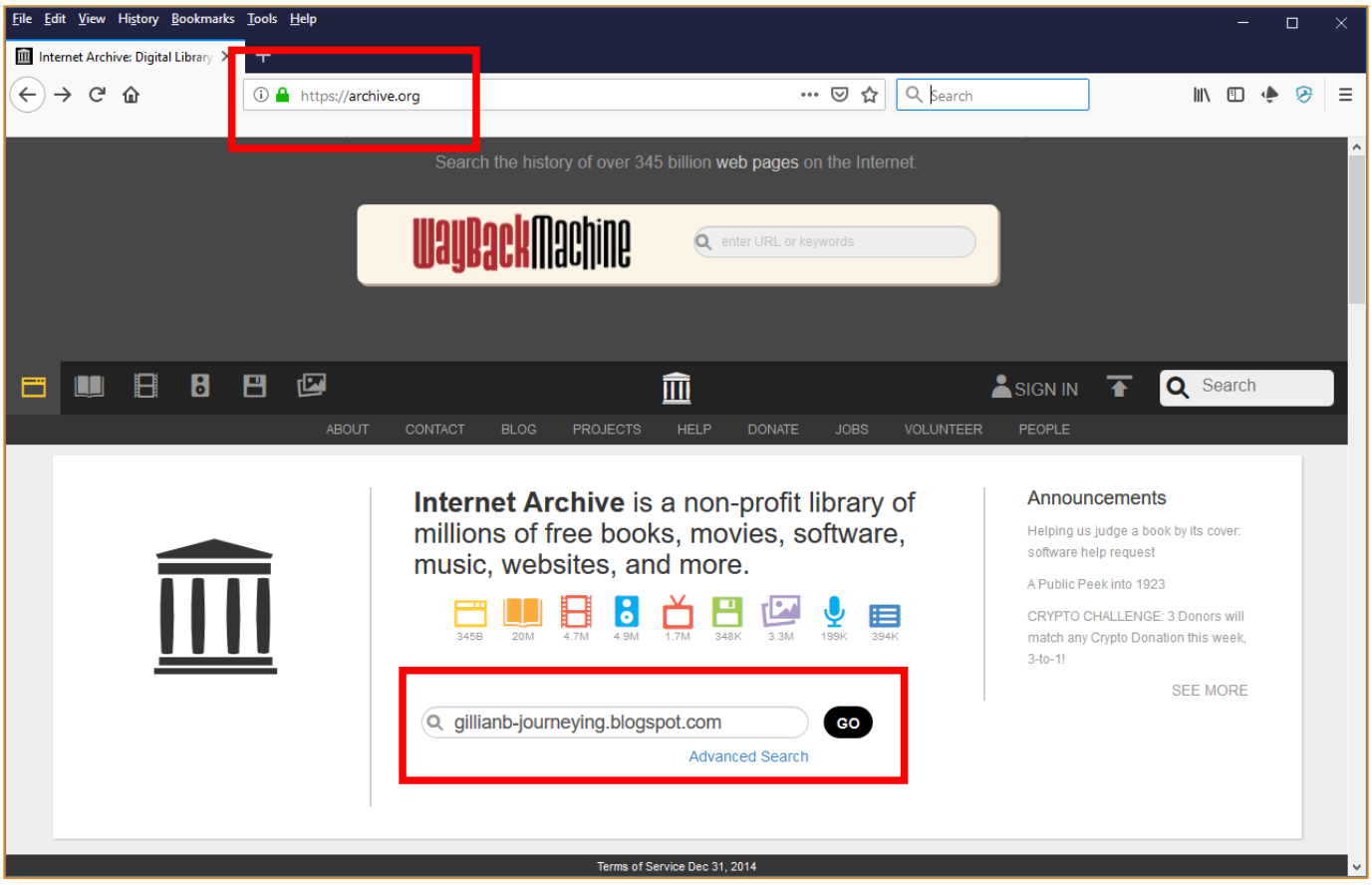
You are involved in an investigation where a client who used to have a blog, lost access to the blog due to a change of email address and deletion of the address that was used to set up and access the account. The blogging address provided to you is "**gillianb-journeying.blogspot.com**". The client has tasked you with recovering the blog posts that were written.

3. Armed with the address of the blog, go to a browser and see what comes up, and if there are any clues to follow.



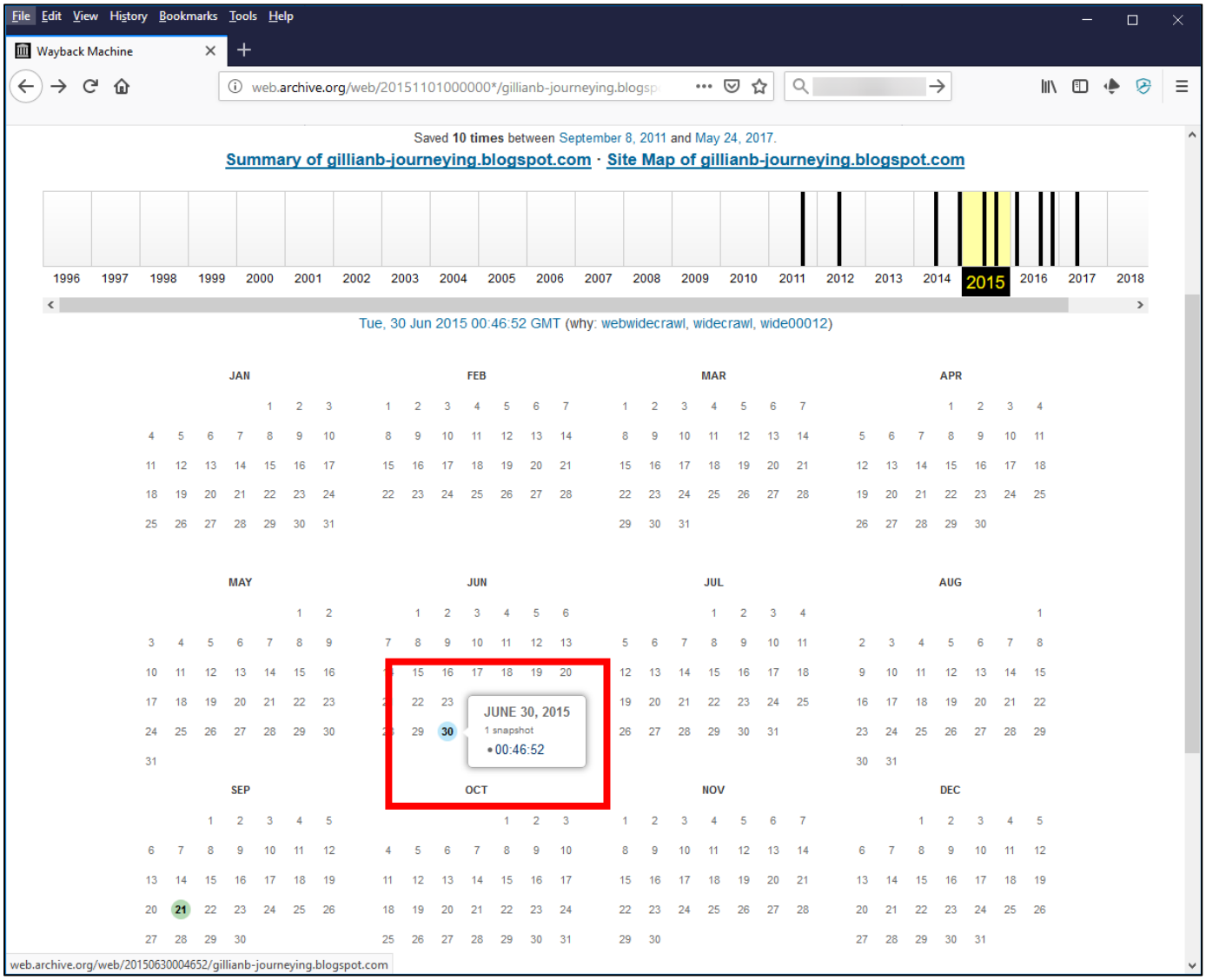
4. We note that indeed the blog is not present any longer, and the suggestions when you follow the included link are unproductive. Time to visit the **Wayback Machine**.

5. In your browser, go to **archive.org**. In the search field on that page, enter the blog address.

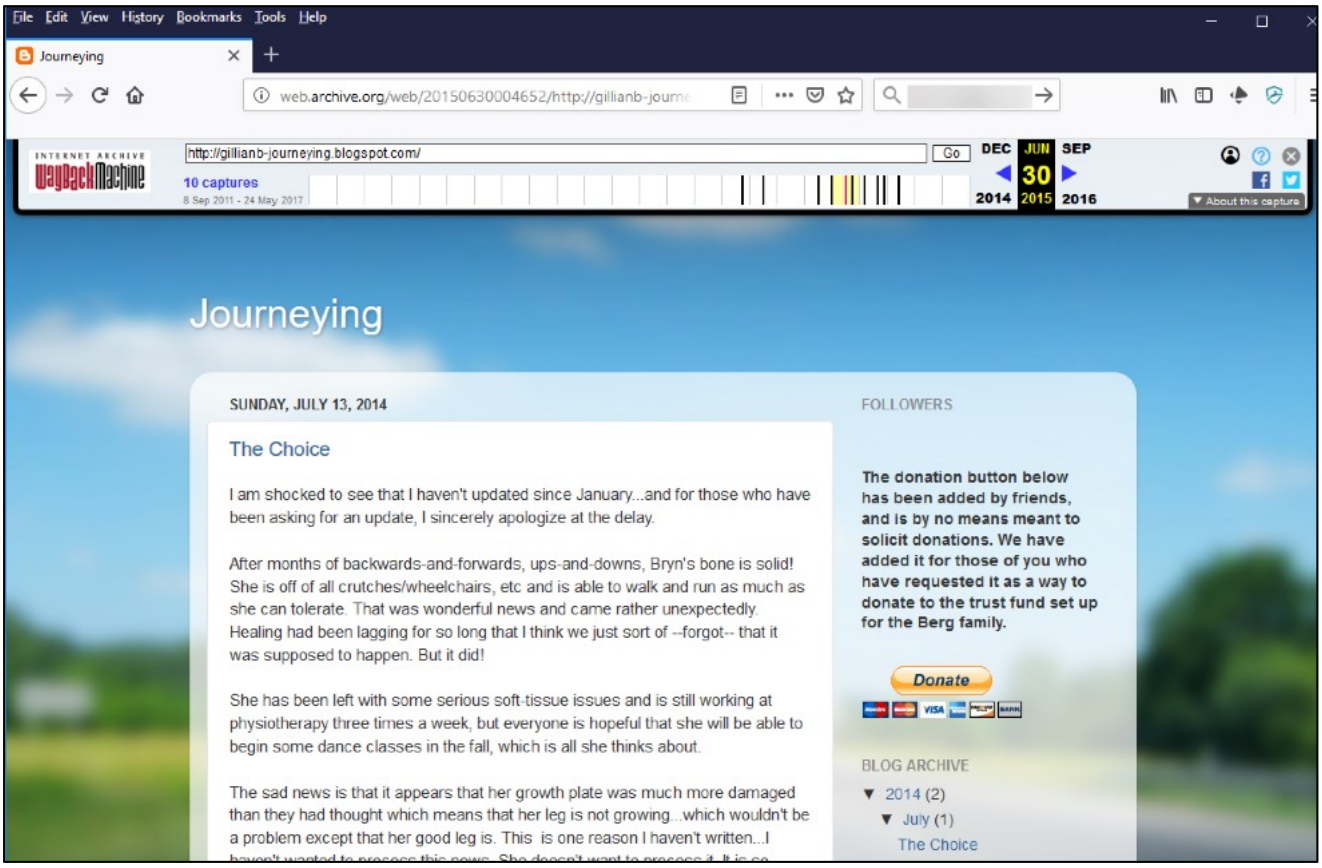


6. You can place the address in either search field. We have used the lower one. Click **GO**.

7. You will see a calendar. The bar at the top of the page will delimit the calendar by years. Selecting a year will show that year's calendar, and you can then select any date with a blue highlight. Hovering your mouse over the blue date will show the snapshots of the website at that date and time. Click on the snapshot for **JUNE 30, 2015**.



8. You will be presented with the webpage as it existed on that day.



Exercise - Questions

1. This is not everything that is available of course. It is only whatever was captured on that date and time when the web spiders crawled the site. You can continue to go back and forth checking other dates for older or newer content. Understand that websites can place directives in their **robots.txt** file at the root of the website that will tell spiders to specifically NOT archive the site to locations such as **archive.org**.

   a. What date is the oldest blog post available for this site?

      _____

   b. Is it the first blog? If not, how do you know?

      _____

As we attempted to recover the blog material for our client, we discovered that for about a year, the blog site had not been crawled, so there is a year worth of posts that are missing. Our client has directed us to exhaust all available avenues on their behalf. Our next step then is to determine who owns the site where the blog was created. We can do that by using a **Whois** server.

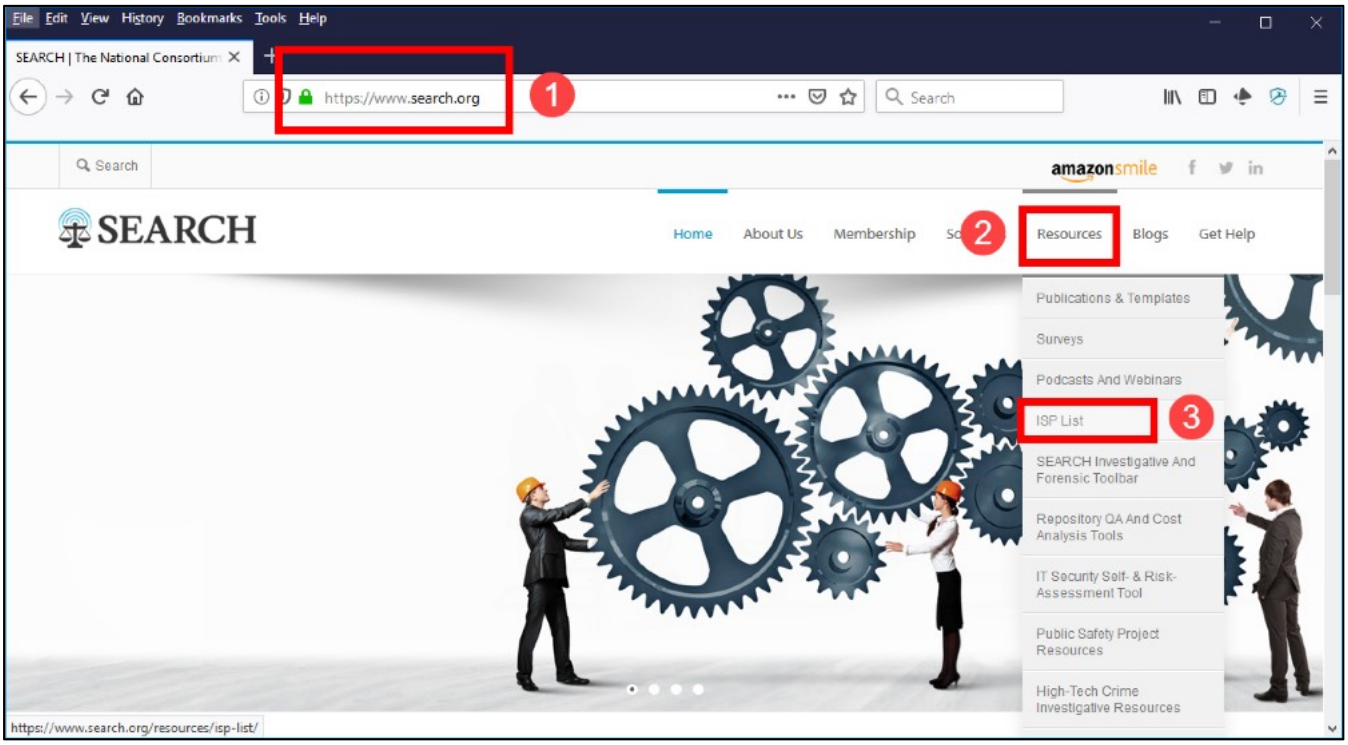
2. We will start at **whois.domaintools.com** and enter the name of **blogspot.com.**

   a. Who appears to be the owner of **blogspot.com**?

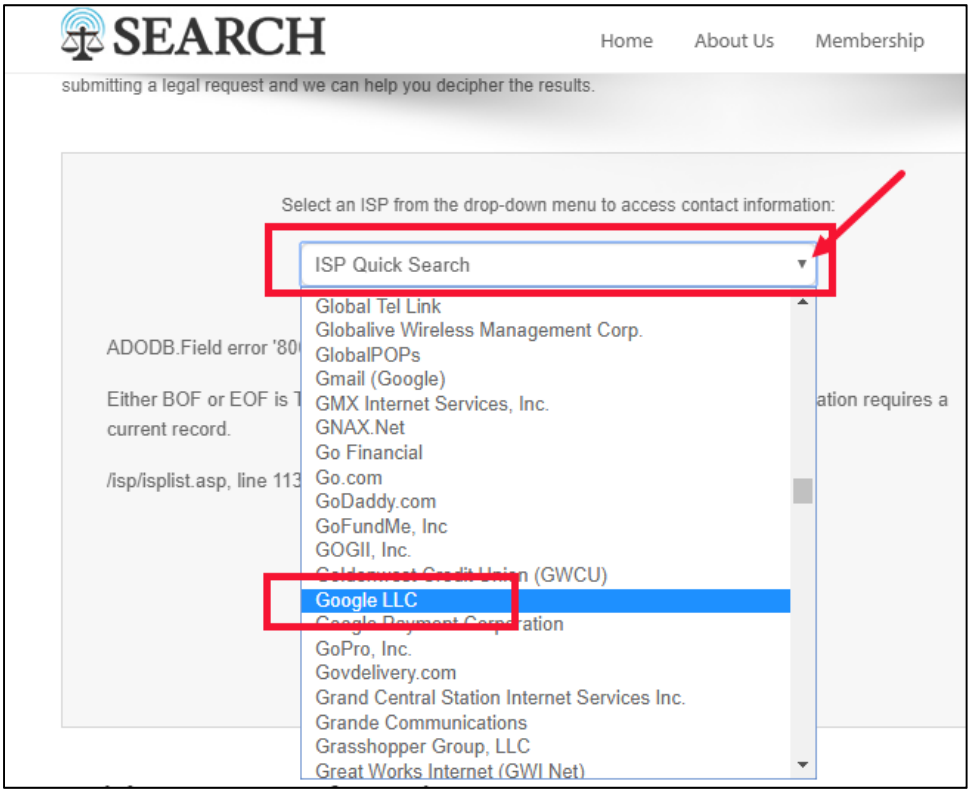
   _____

The client has directed their counsel to serve process on the entity you identified in the previous step, in order to compel them to provide the contents of the site. Counsel has requested we find out who exactly to serve process on.

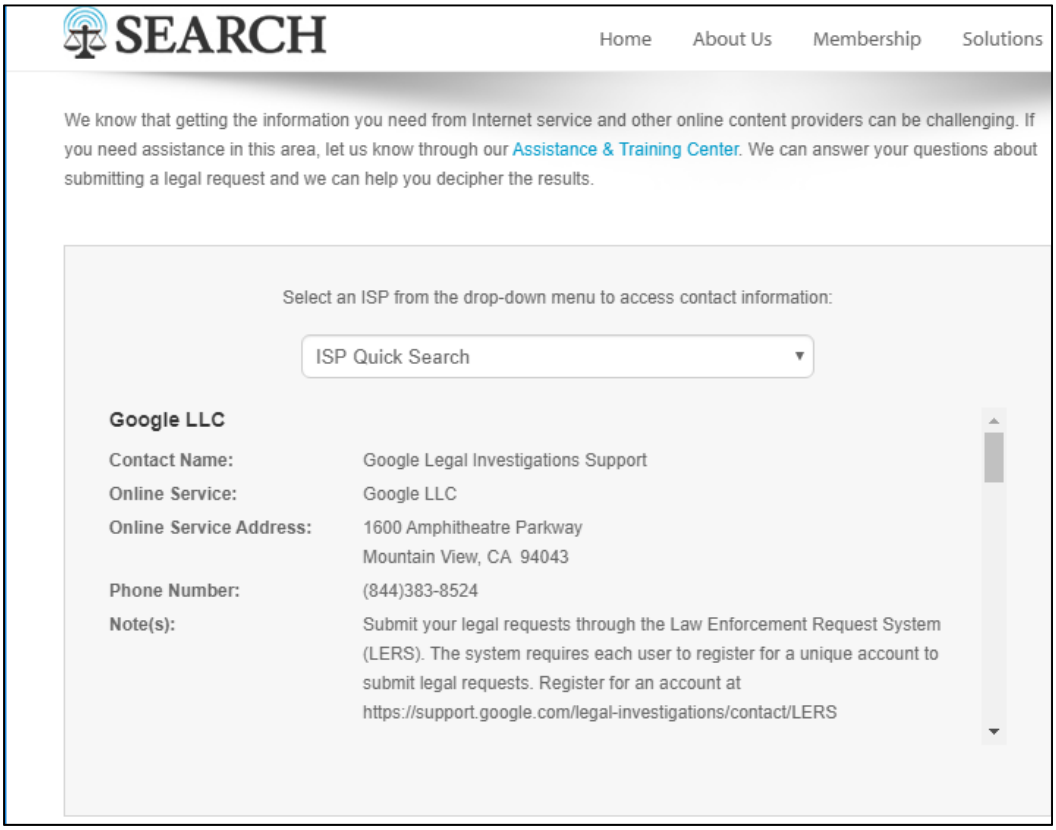
3. Open a browser, go to **search.org,** and select **Resources -> ISP List**.



4. Click on the arrow in the **ISP Quick Search** field, scroll through the impossibly long list until you find the entity you identified in a previous step, and click on it.



5. The legal contact for the entity will be shown.



a. Who is the Contact Name of the entity you identified?

_____

Do NOT assume that this is the end of your trail! This list was designed primarily for law enforcement. If you send documents unannounced, do not be surprised if they are rejected. Most every entry at search.org will have a telephone number. If your matter is civil in nature, call them first and determine if there are any special considerations for you to follow.

Bonus Questions

6. Using what you have learned regarding **archive.org**, try to answer some questions regarding an earlier section of this module.

   a. When is the first noted material for the website **rush.com**?

   _____

   b. Who was the owner at that time?

   _____

   c. When is the first noted date that the band Rush had ownership/control of **rush.com**?

   _____

   d. How can you tell it belonged to the band Rush at that time?

   _____

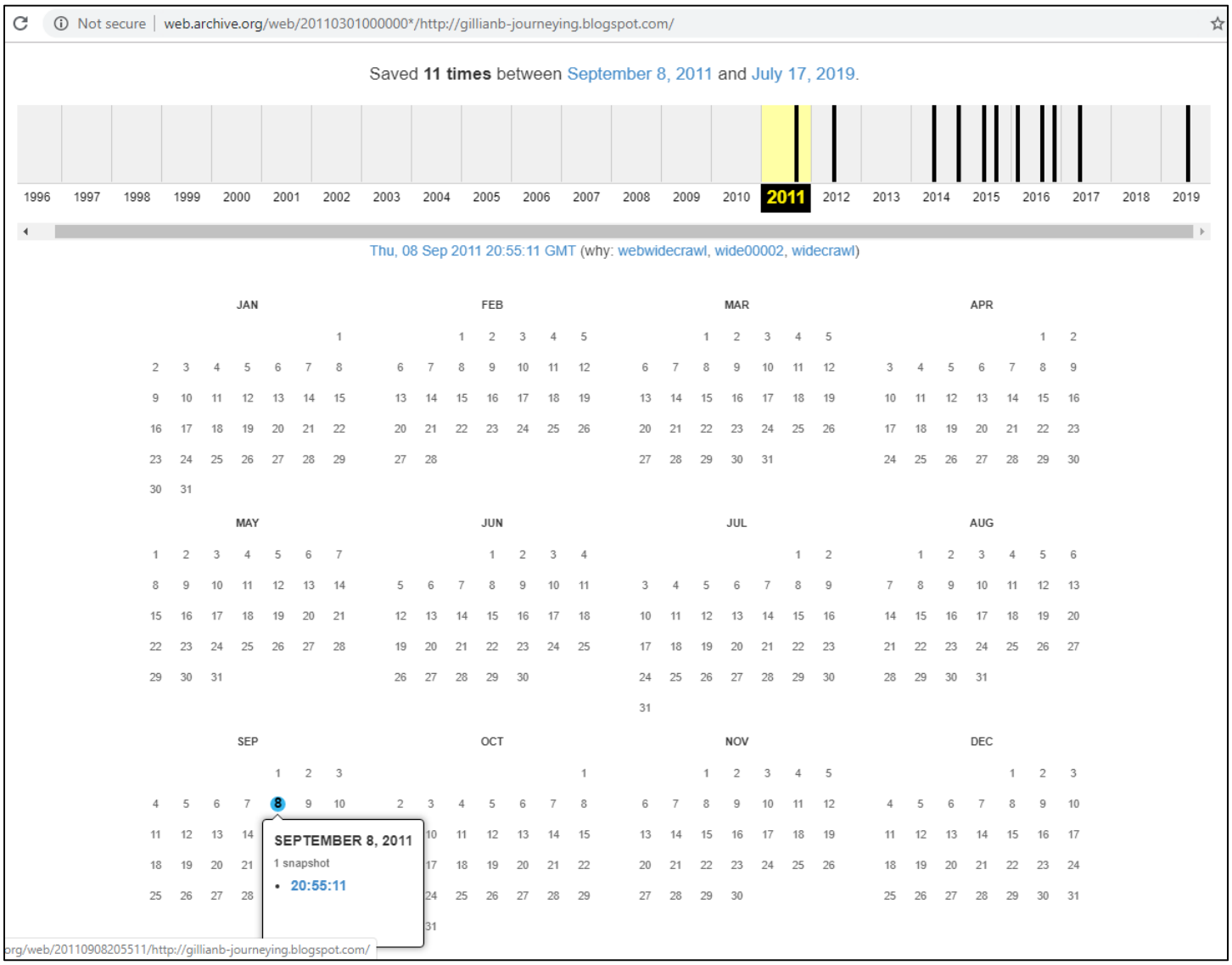
Exercise – Questions with Step-by-Step

1. This is not everything that is available of course. It is only whatever was captured on that date and time when the web spiders crawled the site. You can continue to go back and forth checking other dates for older or newer content. Understand that websites can place directives in their **robots.txt** file at the root of the website that will tell spiders to specifically NOT archive the site to locations such as **archive.org**.

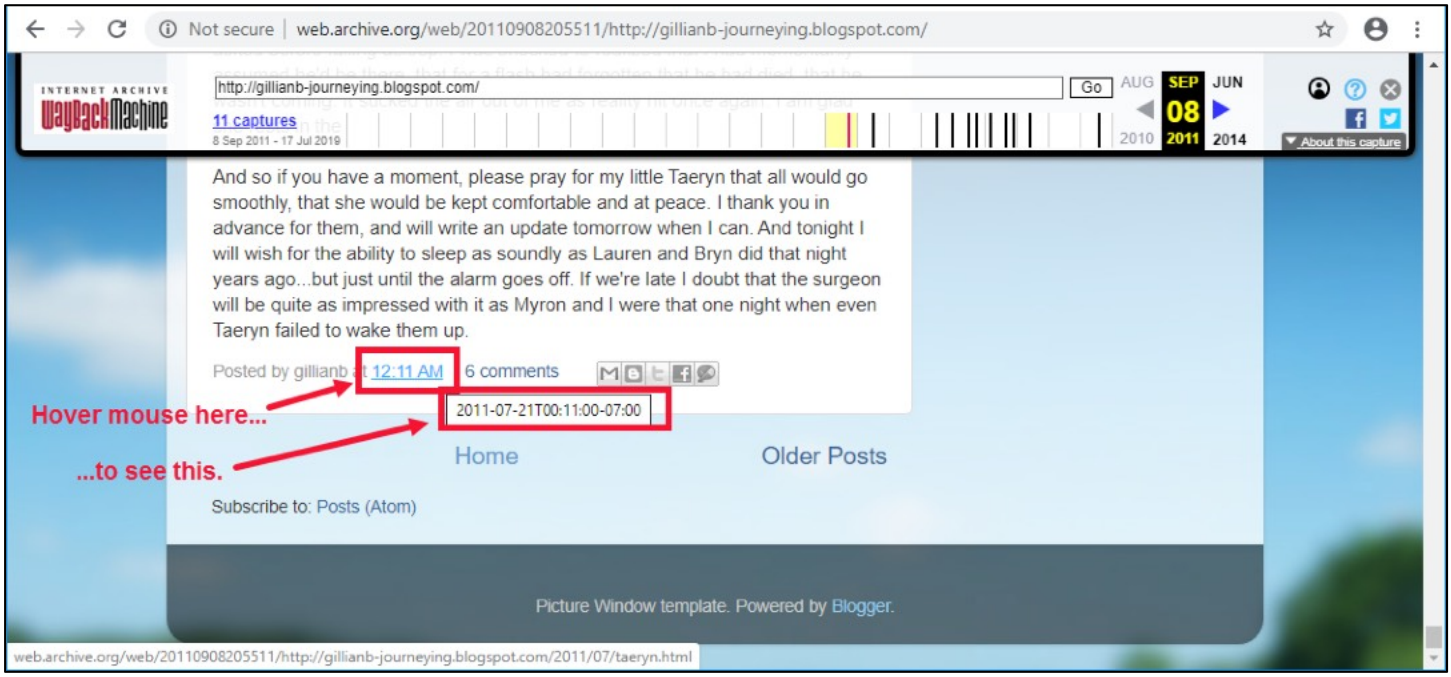
   a. What date is the oldest available blog post for this site?

      July 21, 2011

      In order to get our answer, we must first go back to the calendar, and back to the earliest date of any archive material. In this case, it was September 8, 2011.

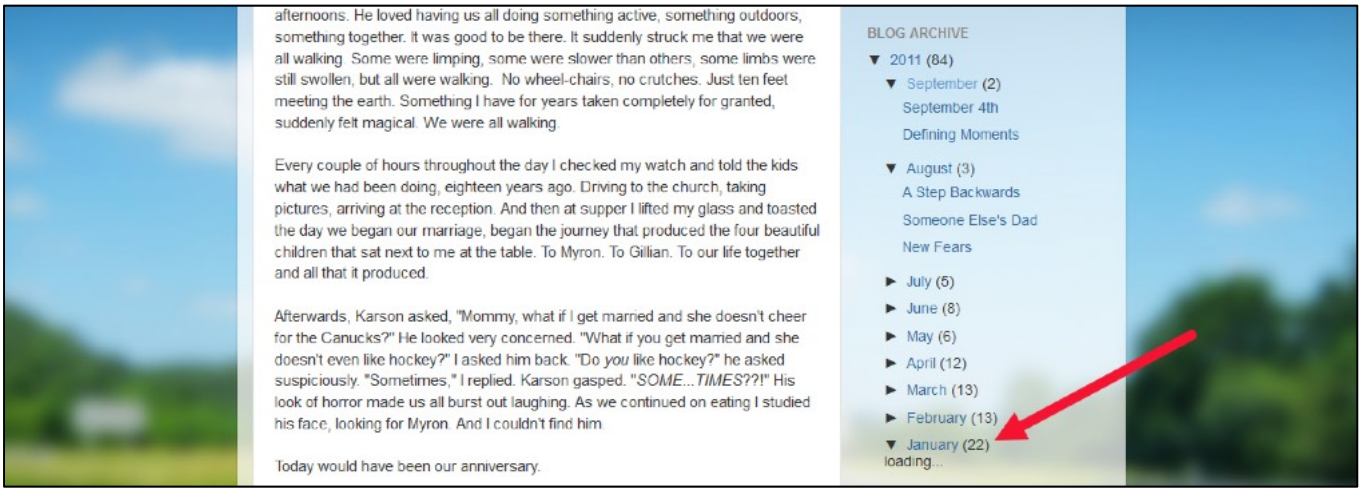


Click on the only snapshot from that date, and go to the resulting web page. Scroll to the bottom, and hover the mouse over the time link. The date will be displayed.



b. Is it the first blog post? If not, how do you know?

No. We can see by the BLOG ARCHIVE listing on the right of the blog post how may posts have been created by month. In this case, January of 2011 had 22 messages. The first of those would be the first blog post.



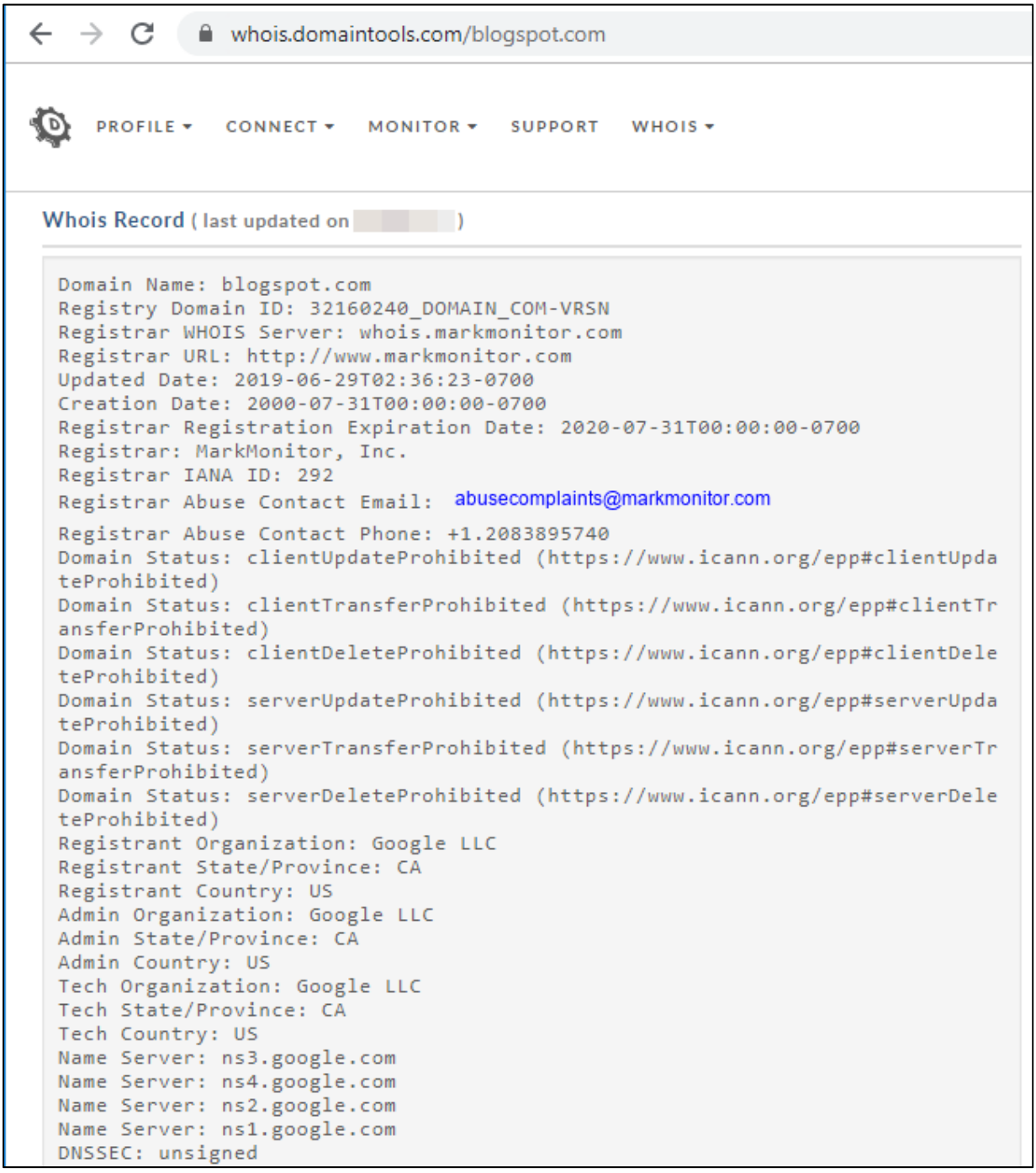
As we attempted to recover the blog material for our client, we discovered that for about a year, the blog site had not been crawled, so there is a year worth of posts that are missing. Our client has directed us to exhaust all available avenues on their behalf. Our next step then is to determine who owns the site where the blog was created. We can do that by using a **Whois** server.

2. We will start at **whois.domaintools.com** and enter the name of **blogspot.com.**

   a. Who appears to be the owner of **blogspot.com**?

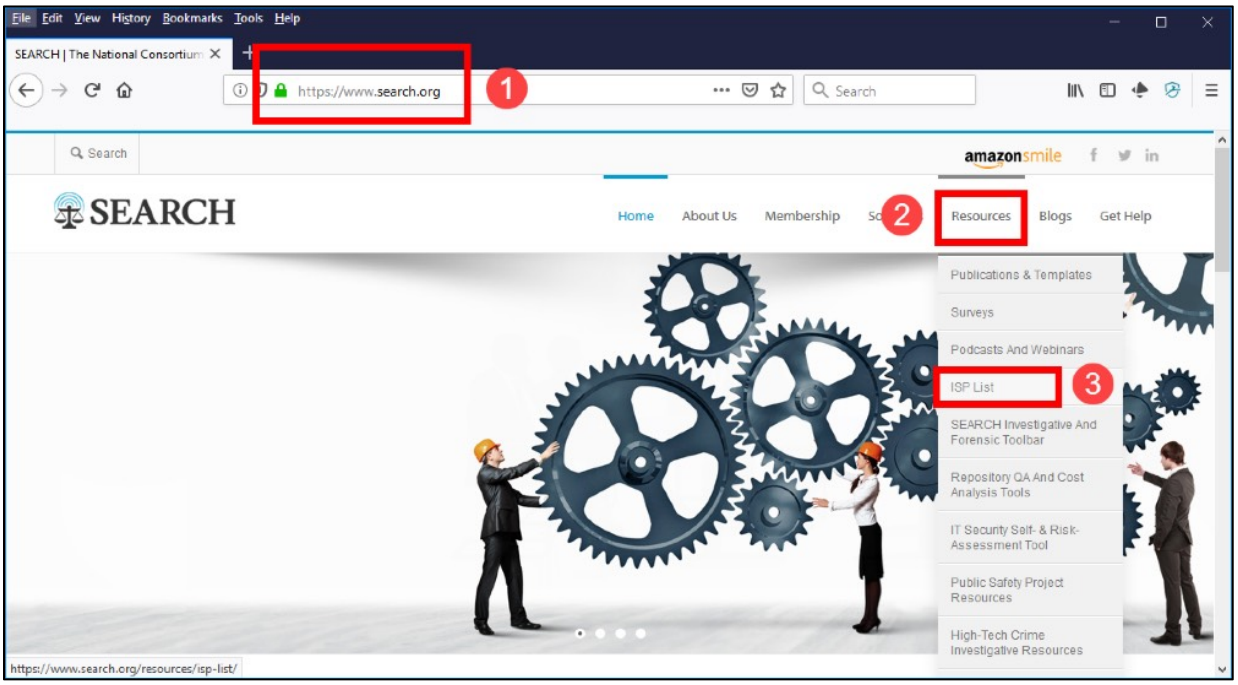
   The Registrant Organization information shows:

   Google LLC

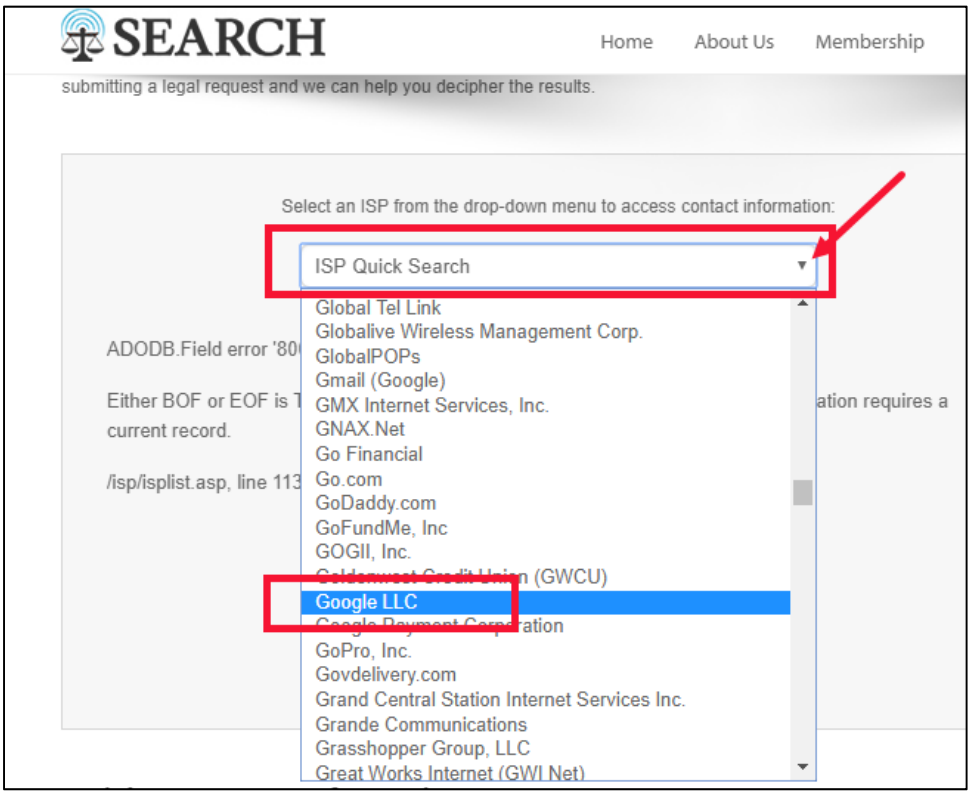


The client has directed their counsel to serve process on the entity you identified in the previous step, in order to compel them to provide the contents of the site. Counsel has requested we find out who exactly to serve process on.

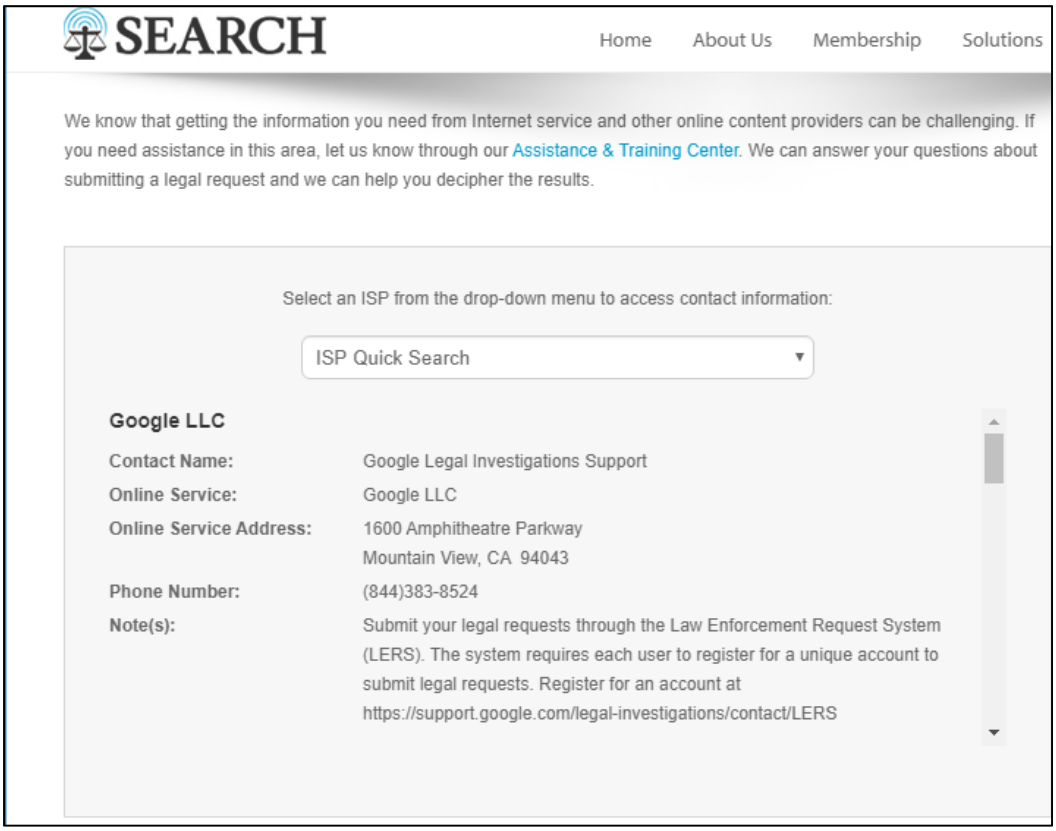
3. Open a browser, go to **search.org,** and select **Resources -> ISP List**.



4. Click on the arrow in the ISP Quick Search field, scroll through the impossibly long list until you find the entity you identified in a previous step, and click on it.



5. The legal contact for the entity will be shown.



a. Who is the **Contact Name** of the entity you identified?

Google Legal Investigations Support

Do NOT assume that this is the end of your trail! This list was designed primarily for law enforcement. If you send documents unannounced, do not be surprised if they are rejected. Most every entry at search.org will have a telephone number. If your matter is civil in nature, call them first and determine if there are any special considerations for you to follow.

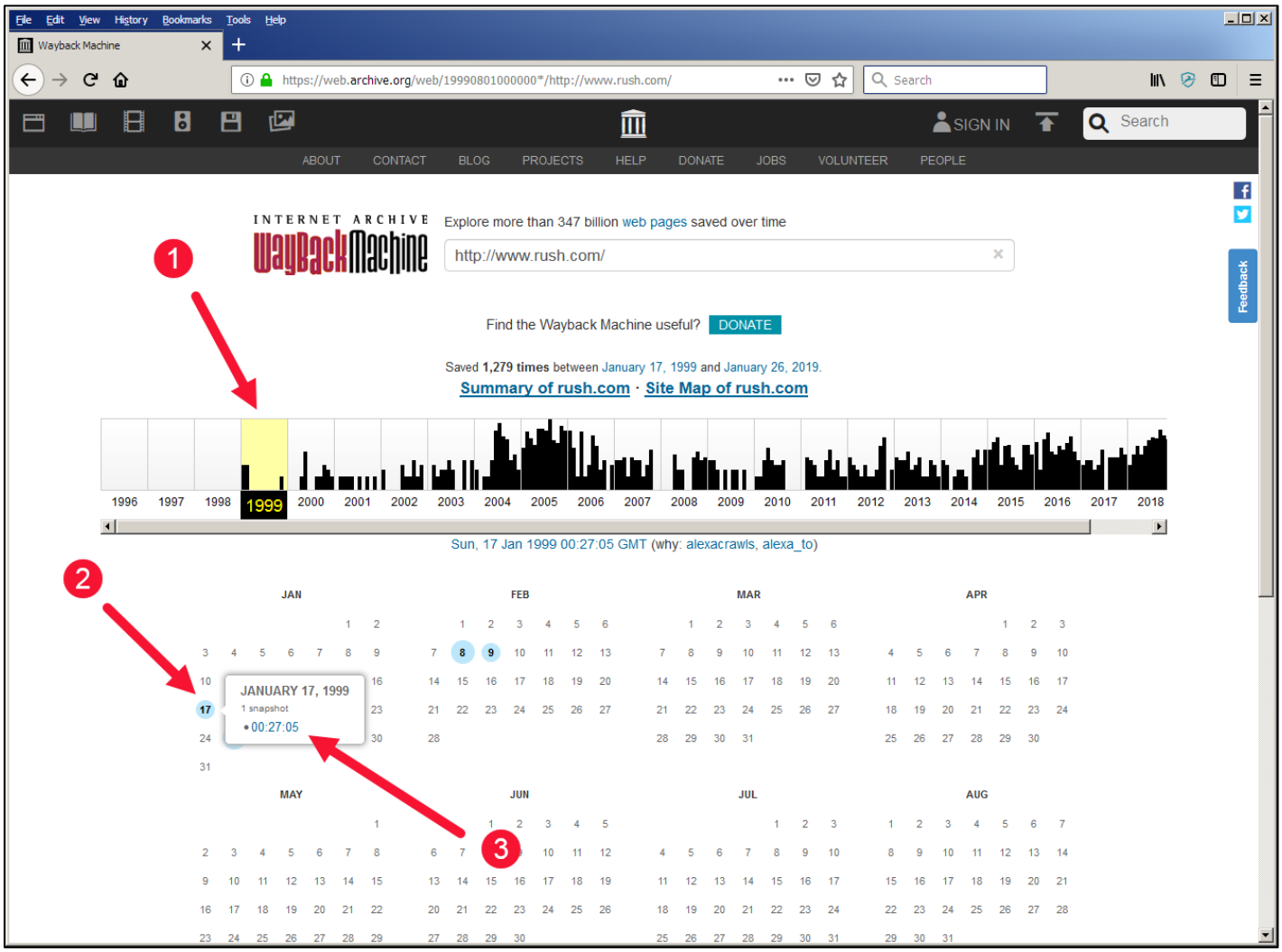
Bonus Questions

6. Using what you have just learned, try to answer some questions regarding an earlier section of this module.

a. When is the first noted material for the website **rush.com**?

January 17, 1999

Using the archive.org website, research the URL of rush.com. Note the calendar bar that is displayed by years. Select the year of first activity, and then the first date in the calendar that has a blue dot on it.



b. Who was the owner at that time? **Once you have clicked on the time above, you see the website as is was at that moment.**

Bob Ames

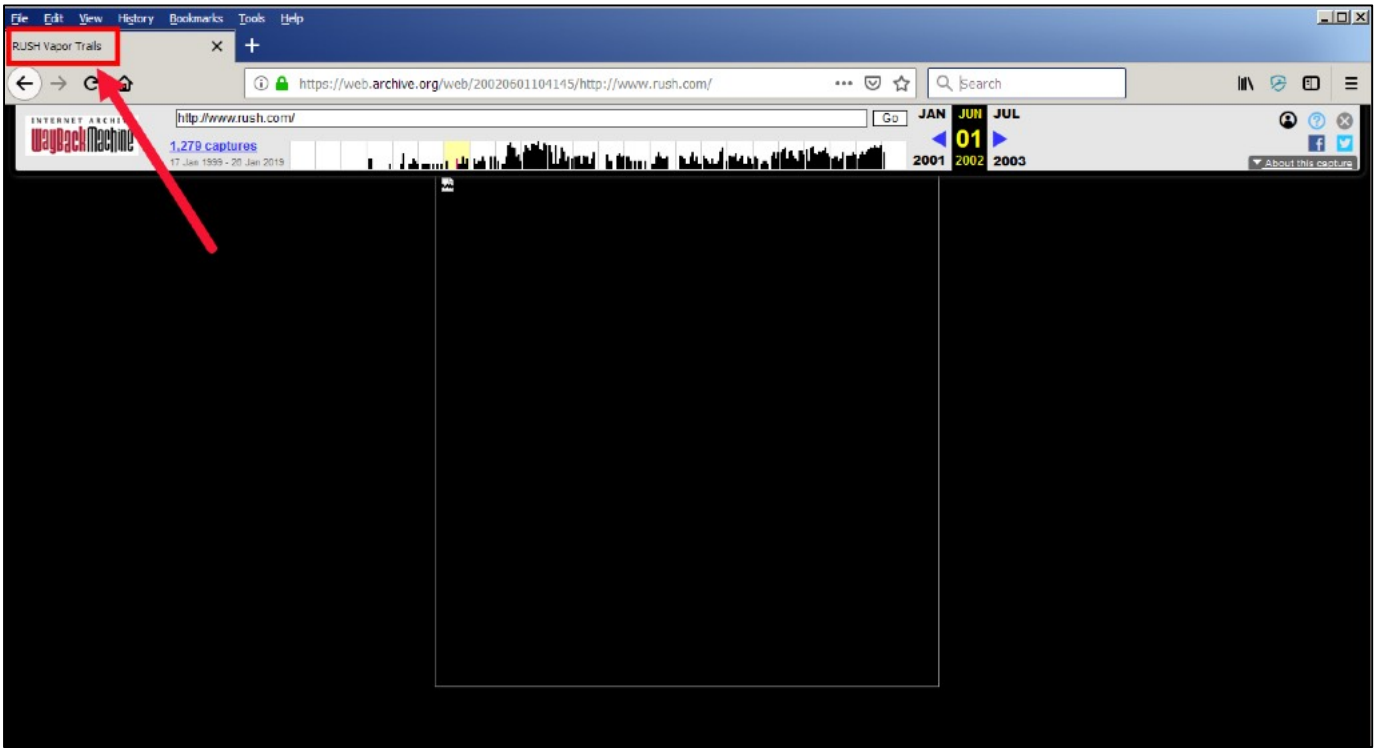
c. When is the first noted date that the **band** Rush had ownership/control of **rush.com**?

June 1, 2002

Keep checking subsequent years/dates until you find a web page that appears to belong to the band, rather than Bob Ames.

d. How can you tell it belonged to the band Rush at that time?

The browser tab contains the name RUSH Vapor Trails. A quick Google search will show this is the name of one of the band's albums.



Exercise—Key Takeaways

- Half the battle in gathering information is knowing where to look and how to ask.
- Archive.org will maintain information, including websites, long after information has been changed or removed.
- Websites like search.org assist in finding the entities responsible for Internet infrastructure.

Exercise 5.2A—PCAP Collection

Background

Network traffic is a rich source of forensic information that allows for discovering things such as computers that are engaged in conversations, files being transferred, hostname lookups, and more. It is important for forensic practitioners to know how to interpret network traffic in order to be able to extract this kind of detail.

Before you can analyze network traffic, however, you must know how to capture it. In this exercise, we will see how to capture network traffic to pcap files which can be analyzed in an offline fashion.

Once we begin analysis, we will use precooked pcap files to ensure all students are able to achieve the same findings. We will use a collection of tools such as Wireshark, tshark, passivedns, and NetworkMiner to find and extract rich amounts of information from pcap.

Objectives

- Understand how to use dumpcap to collect network traffic to pcap files
- Use filters to limit the traffic collected by dumpcap

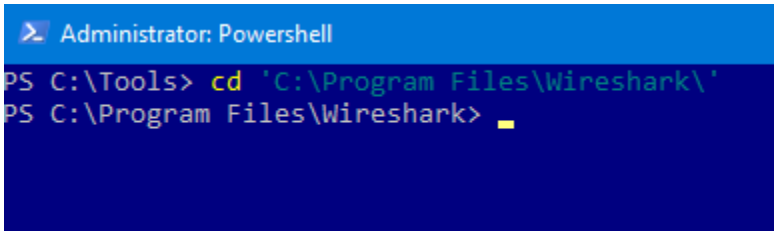
Exercise Preparation

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

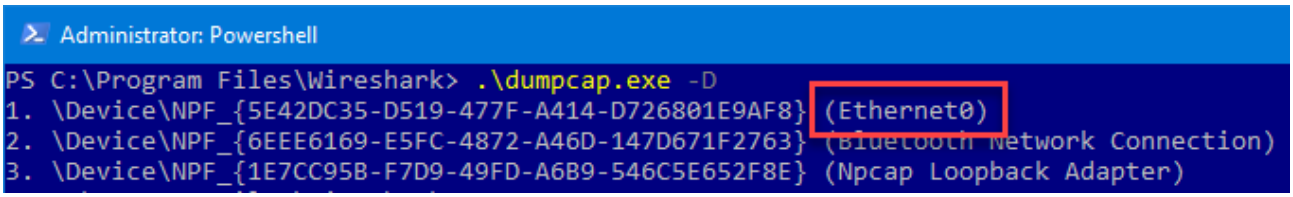
NOTE: If you are in an air gapped environment, or find yourself in another situation without Internet access, you can still perform the steps in this exercise to practice collecting network traffic. You may not actually record as much as you would if you did have Internet access, but the takeaway is collection. We will switch to a precooked pcap file for our analysis work in later exercises.

3. Open **PowerShell** using the shortcut on the **Desktop** and navigate to **`C:\Program Files\Wireshark`**



4. Get a list of available interfaces with the **-D** switch:

```
.\dumpcap.exe -D
```



Since we are on a virtual machine that is connected to the host via an ethernet adapter, look for an entry like what is shown above. The number at the left is the interface number.

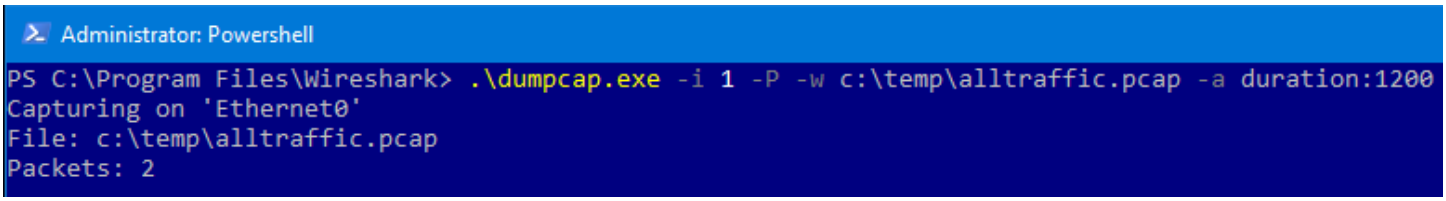
Record your interface number: _____

5. Now that we know which interface to use, we can start collecting traffic and saving the packets to a file. We will also limit the collection to 20 minutes, but you can always stop the collection via **`CTRL-C`**.

```
.\dumpcap.exe -i 1 -P -w C:\Temp\alltraffic.pcap -a duration:1200
```

The **`i`** option specifies which interface number to use (most likely to be interface number 1).
The **`P`** option specifies to use pcap format vs pcapng, for greater compatibility.
The **`w`** option specifies the file to save packets to. We recommend always using an absolute path.
The **`a`** option lets us specify a stop condition. In this case, we want to stop after 1200 seconds, or 20 minutes.

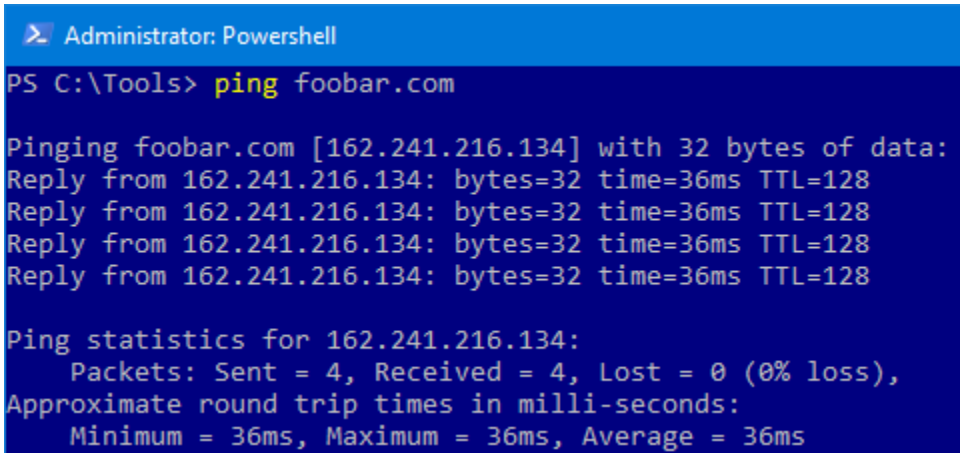
While you do not need to specify a stop condition, it can be useful to prevent massive pcap files from being created. There are also other options that let you create new files (**`-b`**) should you want to create several pcap files during collection.



NOTE: Be sure to complete the next few steps below within 20 minutes so all your traffic is recorded. This should be plenty of time, and you will be instructed when to stop the capture below.

6. With collection underway, open another **PowerShell** window and ping the following:

```
ping foobar.com
ping dslreports.com
ping rushisaband.com
```

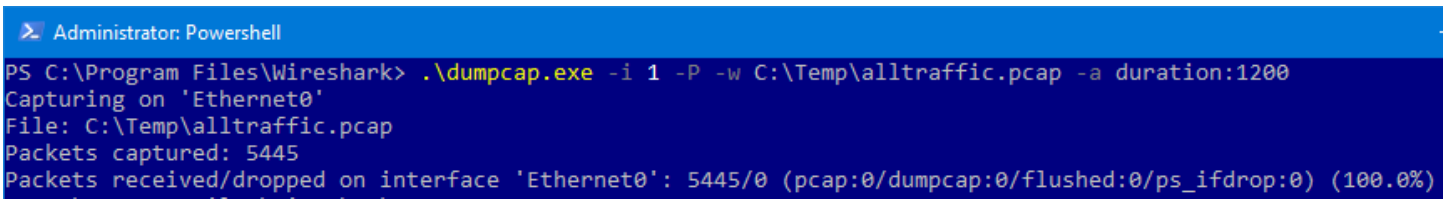


7. Open a web browser (it does not matter which one) and visit the following URLs. **NOTE**: Be sure they all start with '**http://**' and not '**https://**' (If it redirects to **https://**, do not worry about it)

```
http://www.dslreports.com
http://www.msn.com
http://www.espn.com
http://www.bbc.com
http://go.com
```

The order does not matter, nor do you have to browse around on the site after the initial page loads for more than a few clicks.

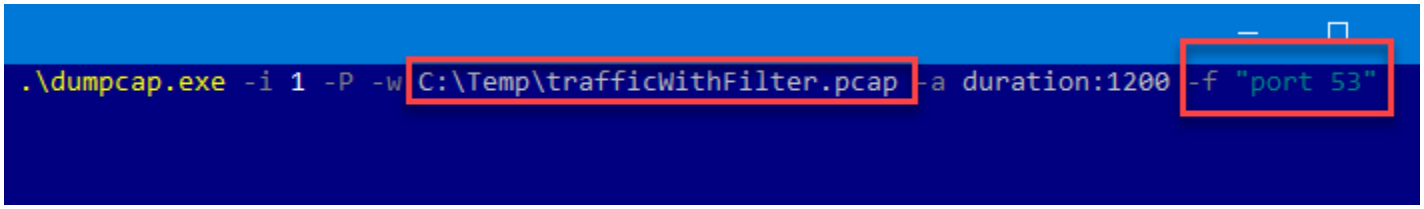
8. Find the PowerShell window where **dumpcap** is running and stop the packet capture by pressing **CTRL-C**.



9. Next, we are going to create another network capture, but this time, we will limit the traffic that is collected with a filter rule.

```
.\dumpcap.exe -i 1 -P -w C:\Temp\trafficWithFilter.pcap -a
duration:1200 -f "port 53"
```

This is the same command as before but notice we have changed the filename to **`C:\Temp\trafficWithFilter.pcap`** and added the **`-f`** switch. This switch allows for specifying a filter (in double quotes) for traffic collection. Here, we are only interested in collecting traffic that involves **port 53**, or **DNS**.



10. With collection underway, open or reuse an existing **PowerShell** window and ping the following:

```
ping foobar.com
ping for498.com
ping rushisaband.com
```

11. Open or reuse an existing web browser window and visit a few of the same URLs from above. Visiting two or three is enough.

```
http://www.dslreports.com
http://www.msn.com
http://www.espn.com
http://www.bbc.com
http://go.com
```

12. Find the **PowerShell** window where **dumpcap** is running and stop the packet capture by pressing **CTRL-C**.

BONUS: Run **capinfos.exe** against your collected captures:

```
.\capinfos.exe C:\Temp\alltraffic.pcap
```

Exercise – Questions

1. In the **PowerShell** window where **dumpcap** was running, review the output and answer the following questions:

   a. For the **alltraffic.pcap** collection, how many packets were *captured* by **dumpcap**?

   _____

   b. For the **alltraffic.pcap** collection, how many packets were *dropped* by **dumpcap**?

   _____

   c. For the **trafficWithFilter.pcap** collection, how many packets were *captured* by **dumpcap**?

   _____

   d. For the **trafficWithFilter.pcap** collection, how many packets were *dropped* by **dumpcap**?

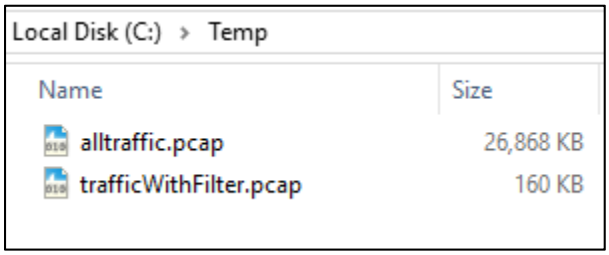
   _____

   Look for the string:

   Packets received/dropped on interface 'Ethernet0': xxxx/yyyy

   The first number after the colon (**xxxx** in the example above) is the packets that were captured and the second (after the forward slash, **yyyy** in the example above) is the number of packets dropped.

2. Open **File Explorer** and navigate to **C:\Temp.** There should be two files that end in .pcap which are the two network captures from above.



   NOTE: Your file sizes will be different from what is shown above. When answering the following questions, use the values from **YOUR** captures.

   a. What is the size of **alltraffic.pcap**?

   _____

b. What is the size of **trafficWithFilter.pcap**?

_____

c. Why is one so much bigger than the other?

_____

_____

Exercise –Questions with Step-by-Step

Your answer to the questions about the number of packets captures will most likely never match anyone else's, or even your own, should you repeat the steps above. This is due to a wide range of factors, such as other software running on your machine at the time of collection, web page contents changing, and so on.

Because of the dynamic nature of packet capture, step-by-step answers are not possible.

Exercise—Key Takeaways

- Tools like dumpcap and tcpdump allow for the collection of network traffic for later analysis.
- The collection of traffic can be filtered based on a wide range of criteria which makes it very flexible.
- Encryption of data on the wire may prevent you from getting all the answers you would like to get.

Exercise 5.2B—PCAP Analysis – Wireshark

Background

Network traffic is a rich source of forensic information that allows for the discovering of things such as computers that are engaged in conversations, files being transferred, host name lookups, and more. It is important for forensic practitioners to know how to interpret network traffic in order to be able to extract this kind of detail.

Objectives

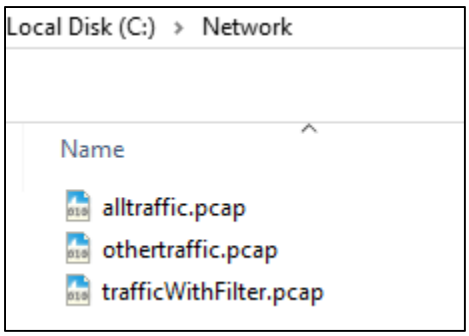
- Use Wireshark to inspect pcap data

Exercise Preparation

1. Boot your **FOR498 Windows VM**

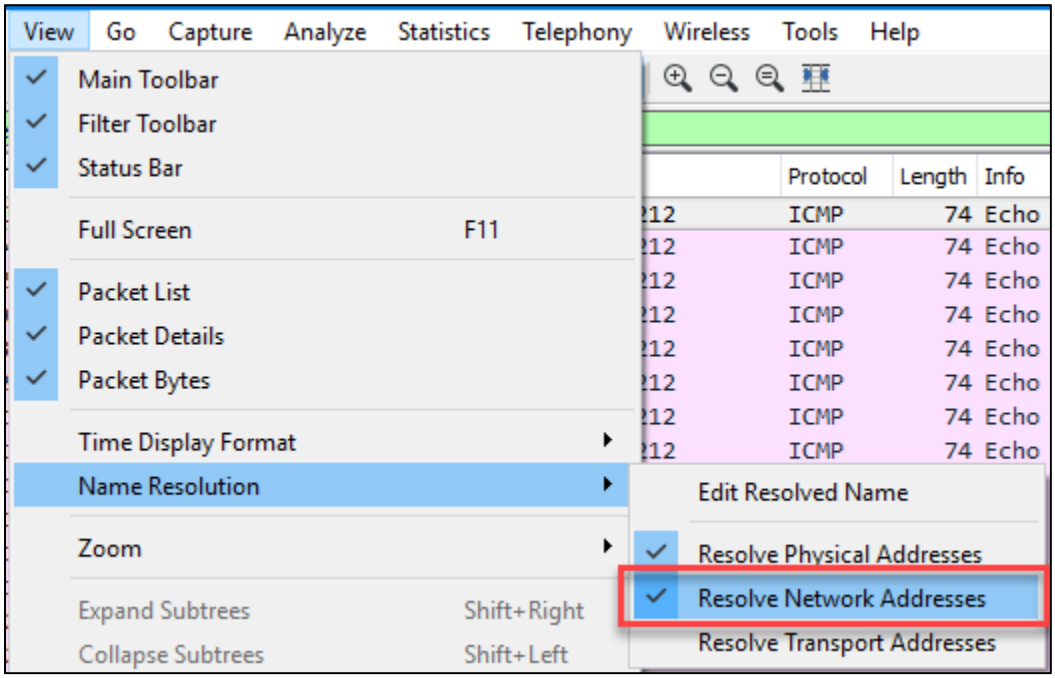
2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Verify you have the three pcap files as shown below in the **C:\Network** folder.

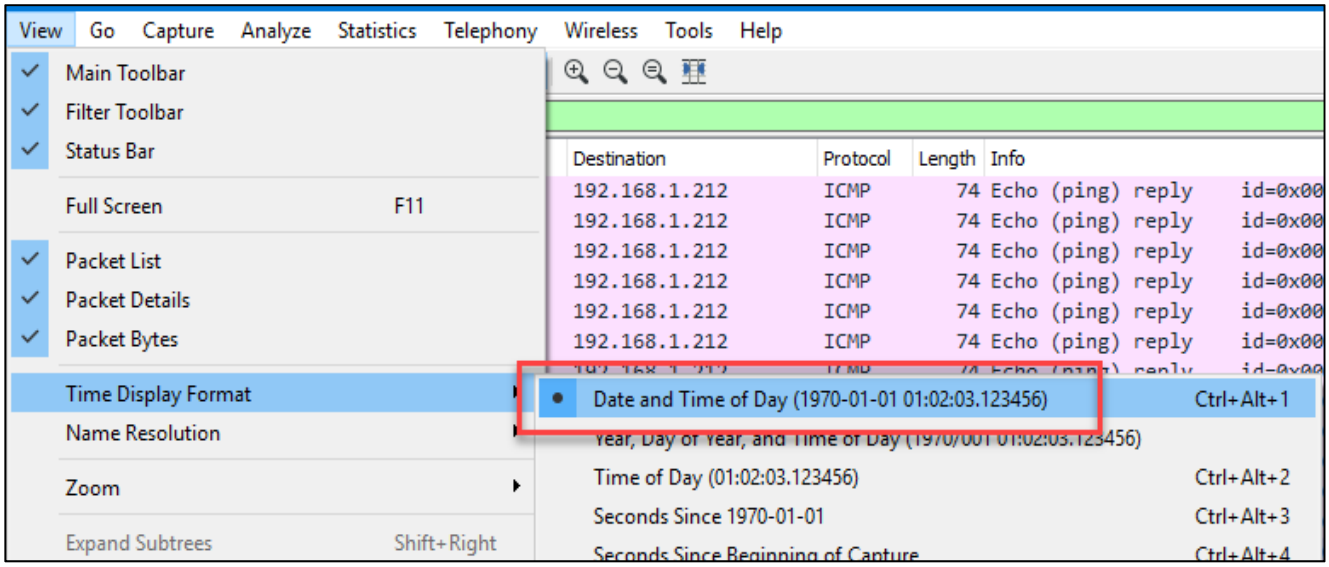


4. Start **Wireshark** using the shortcut in the **Network Tools** fence on the **Desktop**.

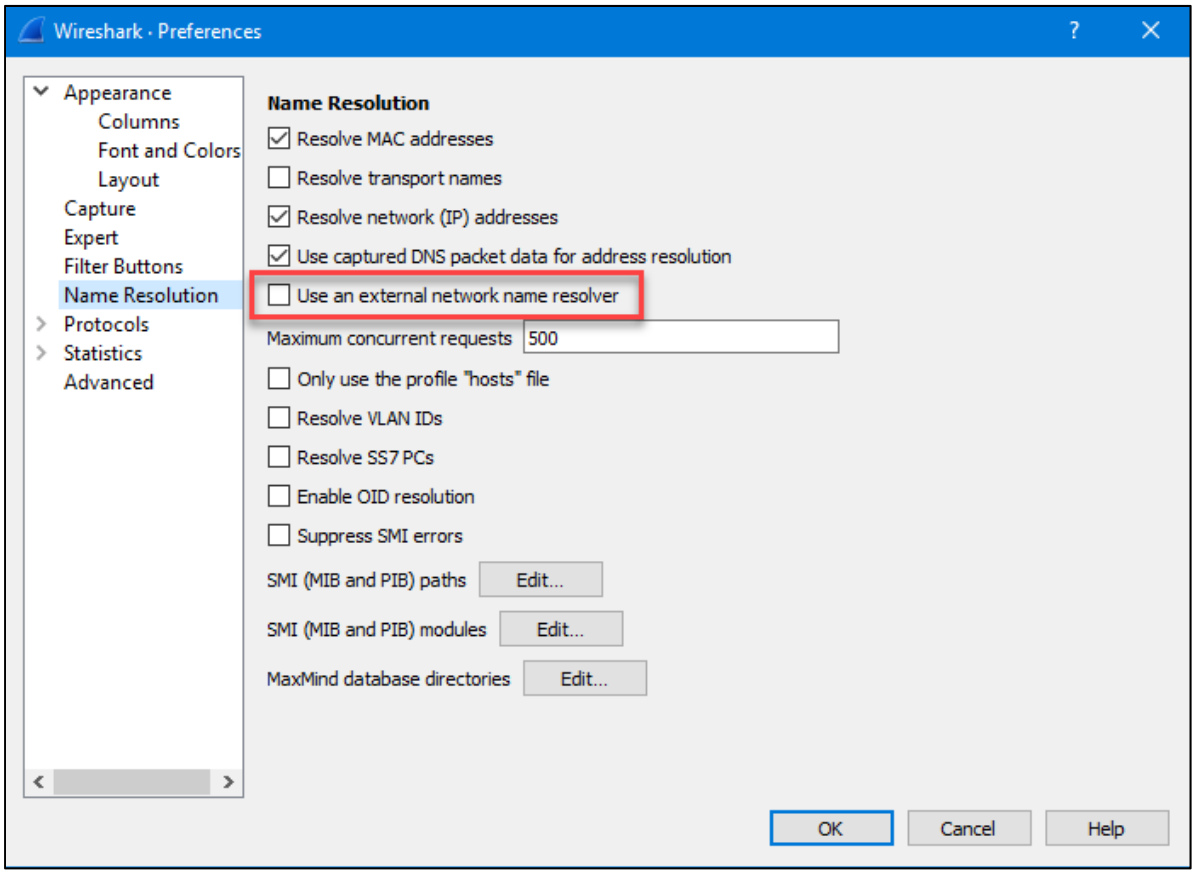
5. Use the **View → Name Resolution** menu to enable the **Resolve Network Addresses** option. This will resolve IP addresses to hostnames based on data in the pcap file.



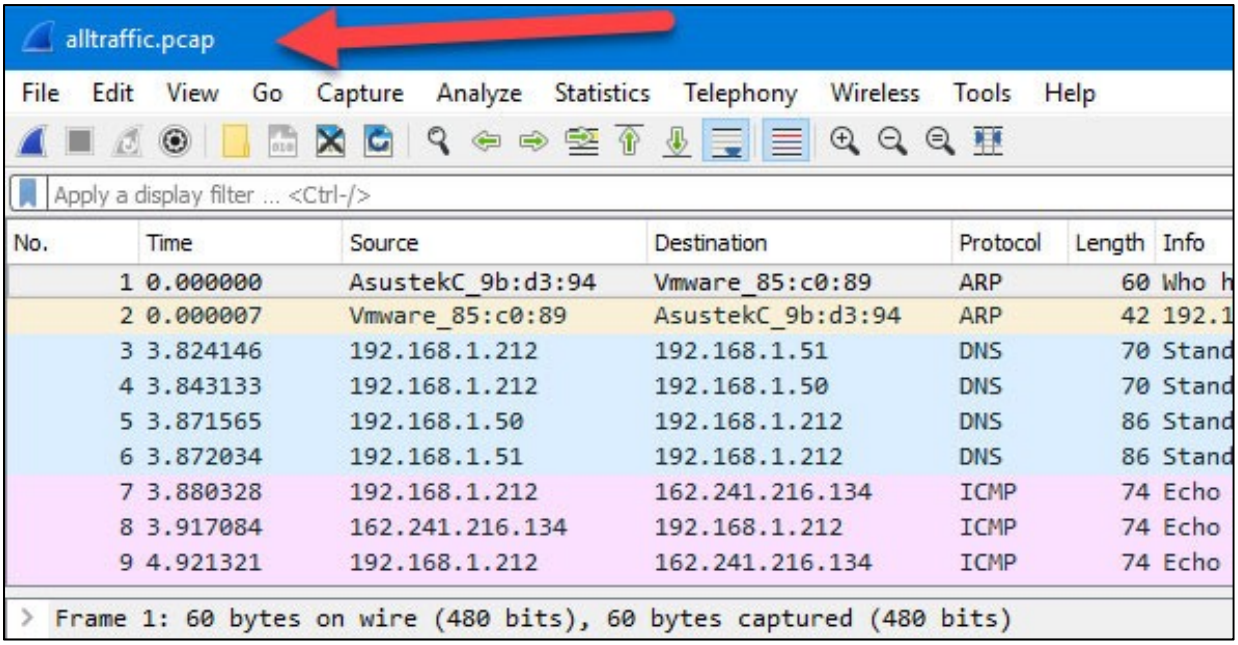
6. Use the **View → Time Display Format** menu and select **Date and Time of Day** at the top. This makes the time easier to read.



7. Finally, click on **Edit → Preferences**, click on **Name Resolution**, and uncheck **Use an external network name resolver**.

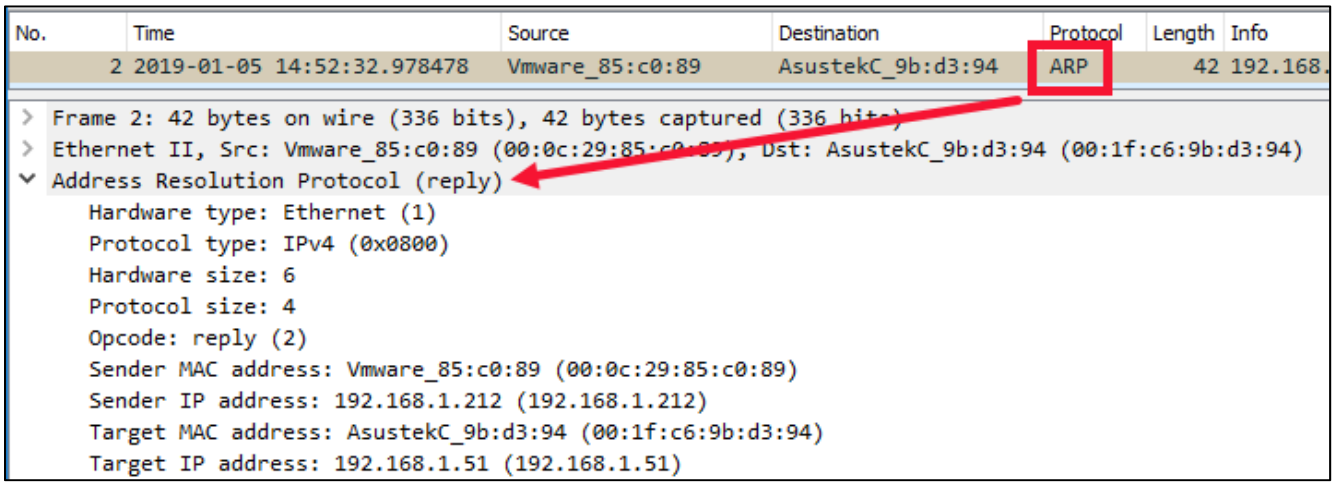


8. Load **C:\Network \alltraffic.pcap** via **File → Open.**



9. Take a few minutes (5 minutes max) to browse the data in the capture. Pay attention to the **Source** and **Destination** columns as well as the **Protocol** and **Info** columns.

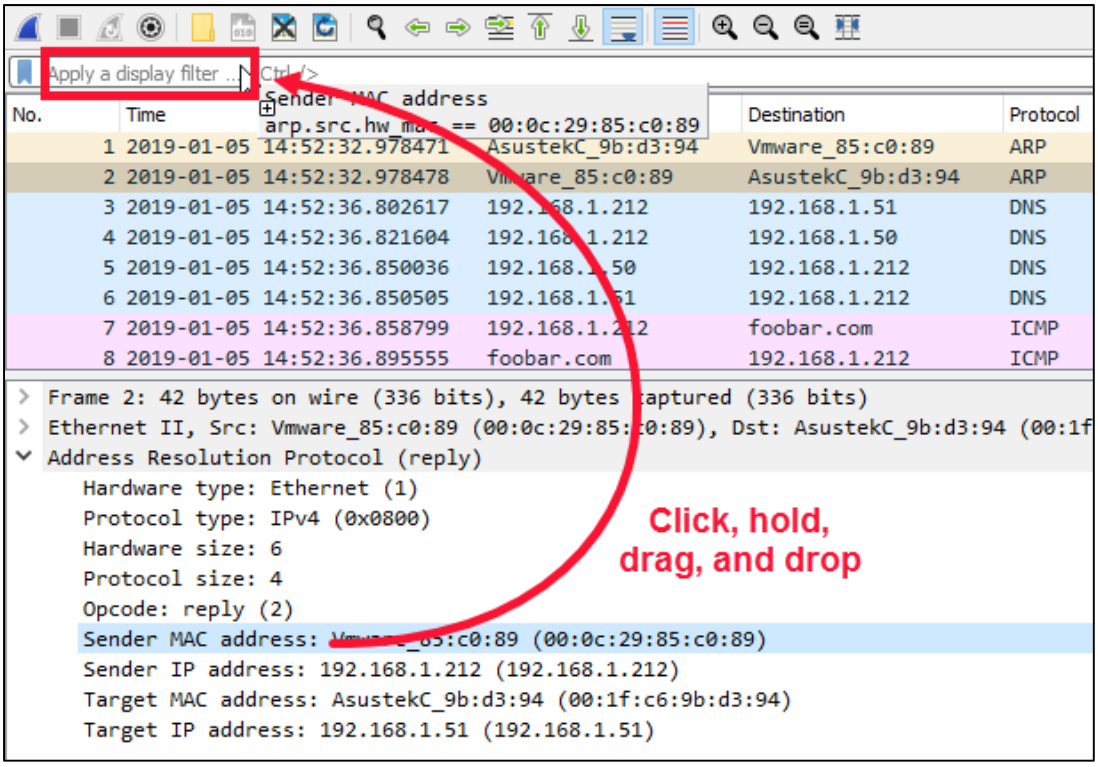
   Also note that as you select different packets, the bottom view changes to represent the data that is contained in one of the packets.



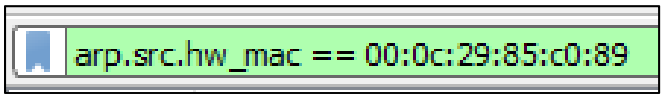
   In the example above, notice that there are three "sections" of data. The first two (**Frame** and **Ethernet**) are collapsed, and the third, **Address Resolution Protocol**, has been expanded.

   Each of the bytes in the packet is broken down into its corresponding properties which makes reviewing things much easier.

   PROTIP: If you find something interesting and you want to create a filter for it, select a property, left click and hold the mouse button, then drag the value to the display filter bar.



This builds the filter and value automatically:



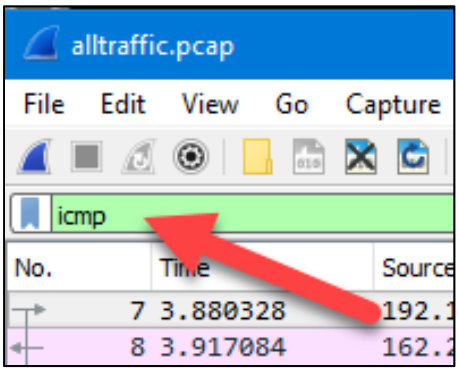
This capability makes learning filter criteria significantly easier!

Exercise – Questions

NOTE: A great reference for all the possible fields is available at **http://for498.com/ovepr**

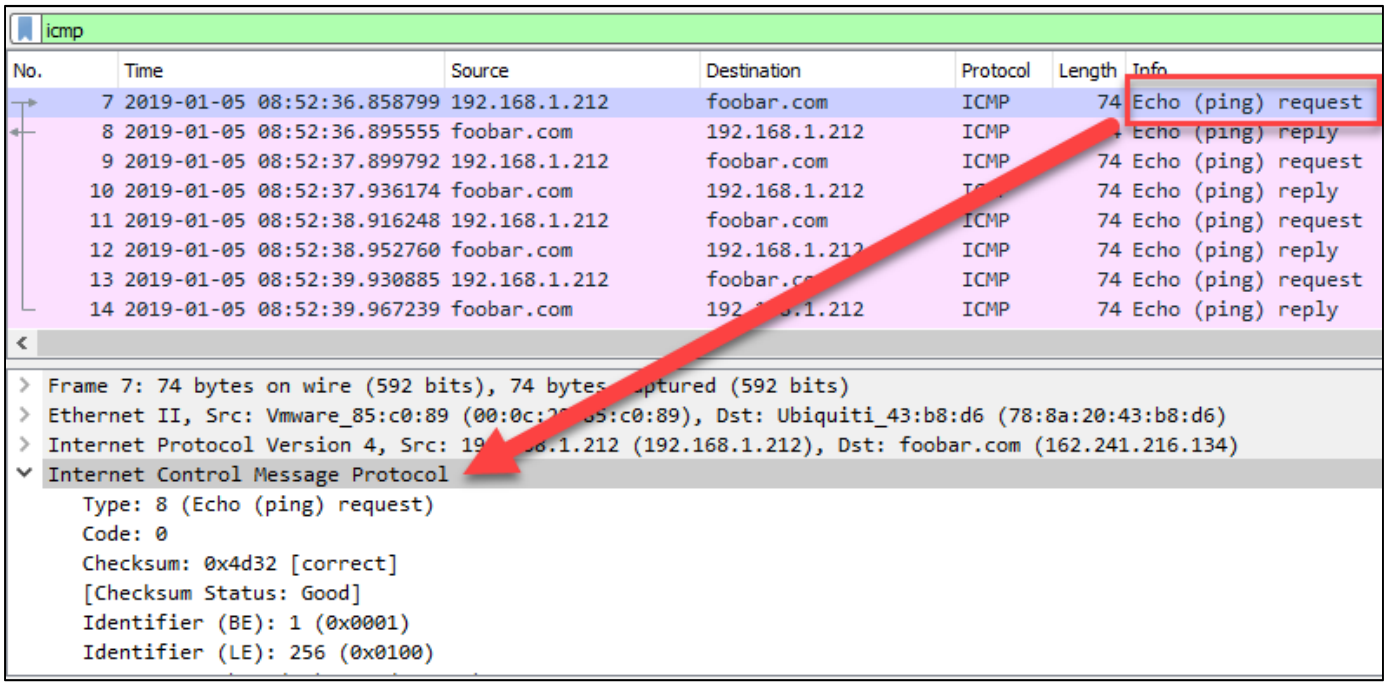
1. Recall in the previous exercise we pinged several domains. The **ping** command uses the **Internet Control Message Protocol (ICMP)** to do its work. We can find these ping requests by using a display filter in **Wireshark**, specifically, the **icmp** expression. In the display filter box, enter the filter below and then press **Enter**.

```
icmp
```



Notice what happened here. All the packets were filtered for only those that match the filter.

2. With the packets now only showing **icmp**, select the first packet in the list and, if necessary, expand the **Internet Control Message Protocol** section.

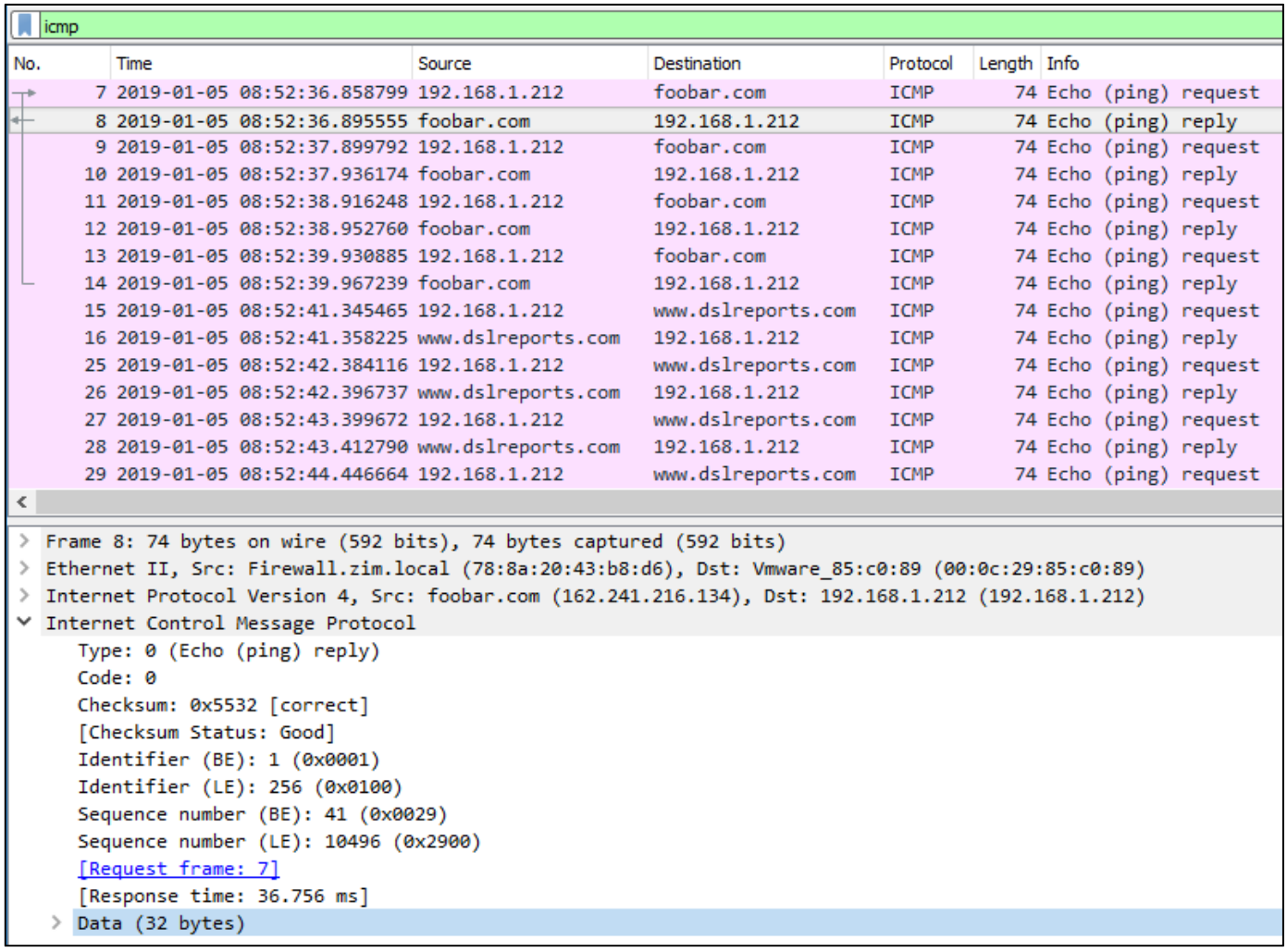


In the packet list, we can see the **Source** of the **ping** (where the request came from), and the **Destination** (the value that was pinged). The **Info** column shows that the first packet is a request, which can also be seen in the details at the bottom.

a. What is the **ICMP** type for a **ping** request?

_____

3. Next, select the **Echo (ping) reply** (it should be the next packet in the list) for the request we just looked at. When looking for the **reply** (assuming there is one that is), you can find it easily by looking for the packet where the **Source** and **Destination** are reversed, as seen below.



a. What is the **ICMP** type for a **ping** reply?

_____

b. In frame 8, how long did it take for the host that was pinged to respond? (Hint: Look for the **Response time** property)

_____

Also note that, once you are looking at a **reply**, you can see the **Request frame** number of the request. This works the other way too, in that the **request** packet would reference the **reply** frame. This lets you verify you are working on the same "pair" of **request** and **reply**.

4. Replace the display filter with the following and press **Enter**.

```
icmp.type == 0
```

a. What did this filter do?

_____

_____

b. How many packets match the above filter? (Hint: Look in the lower right for the **Displayed** counter)

_____

c. Can you find another host that was pinged in addition to the ones specifically mentioned in the **exercise 5.2A**? If so, what is the hostname?

_____

NOTE: We initially pinged the following hosts: **foobar.com**, **dslreports.com**, and **rushisaband.com**

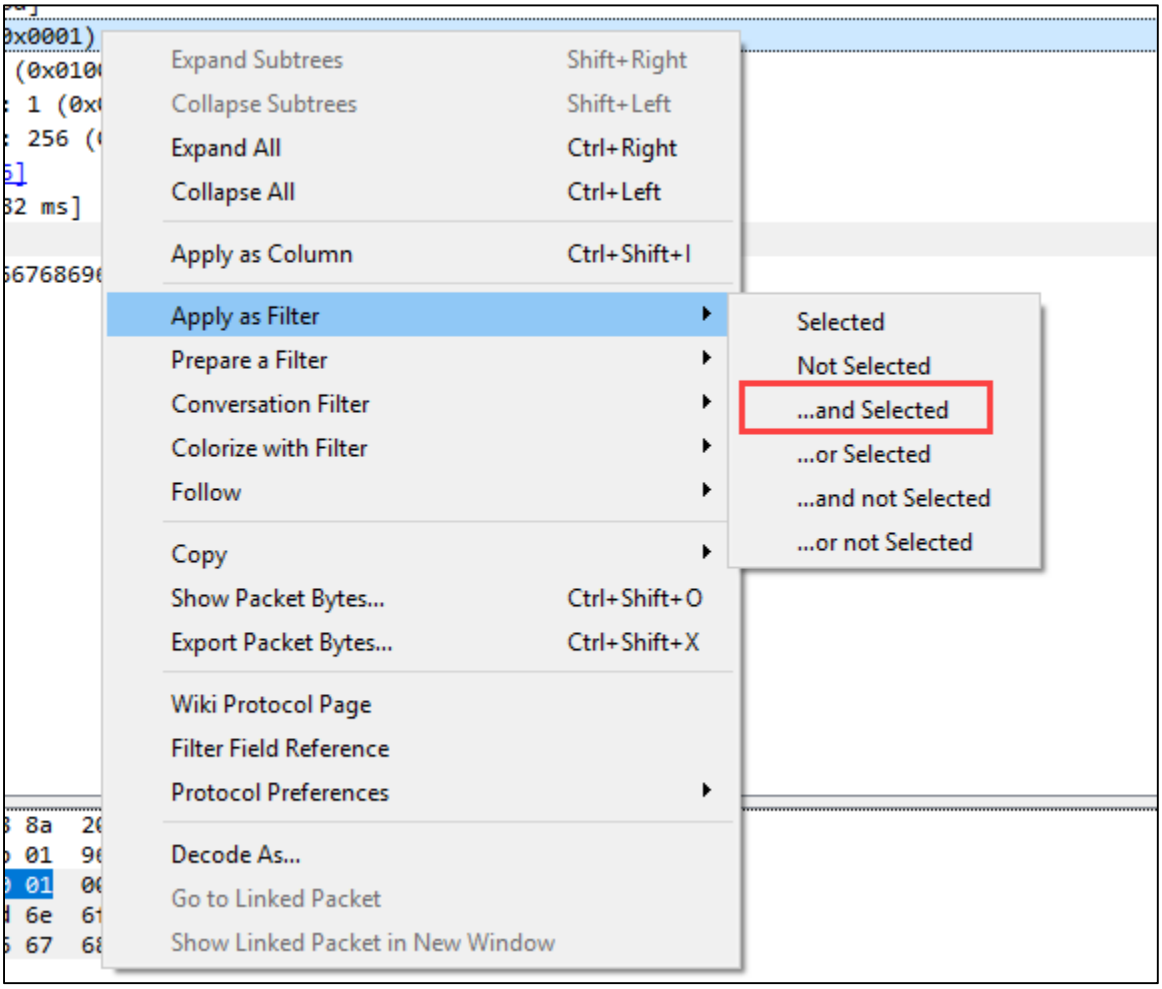
d. If the host responded to the ping, how long did it take?

_____

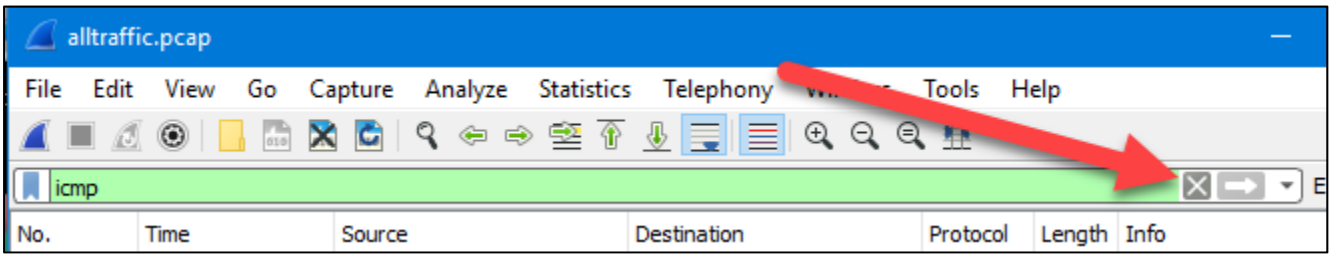
Remember, you can drag and drop a value from the information displayed at the bottom to quickly filter on anything you like.

PROTIP: If you want to *add* additional filter criteria, select the property to add, then right click and choose **Apply as Filter | …and Selected**

Here is an example of this:

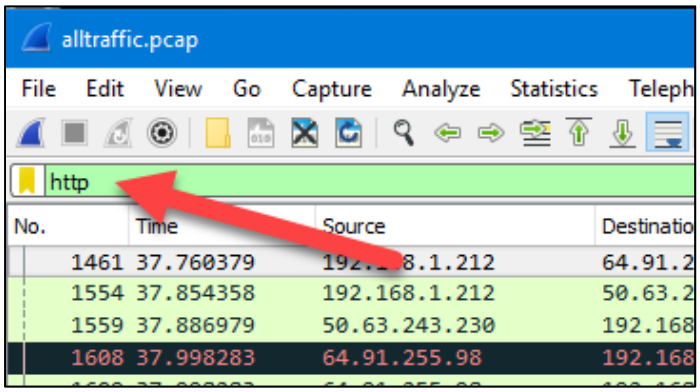


5. Clear any display filters in **Wireshark** by clicking the **X** as shown below.



6. In the display filter dialog, enter the following filter and press **Enter**. Notice what the **Protocol** column now shows.

```
http
```



   a. How many packets were found? (Look in the status bar on the bottom)

      _____

7. Spend a few minutes reviewing the kinds of information available in various **HTTP** requests.

8. For most of the **HTTP** packets, the **Info** pane starts with something like "**GET**" or "**POST**". These are **HTTP** "verbs" that describe interaction with a web server. A "**GET**" would be a request to get data from a web site, whereas a "**POST**" would be sending data to a web site.

   To find all the **GET** requests, change the display filter to the following and press **Enter**.

```
http.request.method == "GET"
```

   a. How many **GET** requests were found? (See the **Displayed** counter in lower right again)

      _____

9. Change the display filter to find all the "**POST**" methods using the following filter:

```
http.request.method == "POST"
```

   a. How many **POST** requests were found? (See the **Displayed** value in lower right again)

      _____

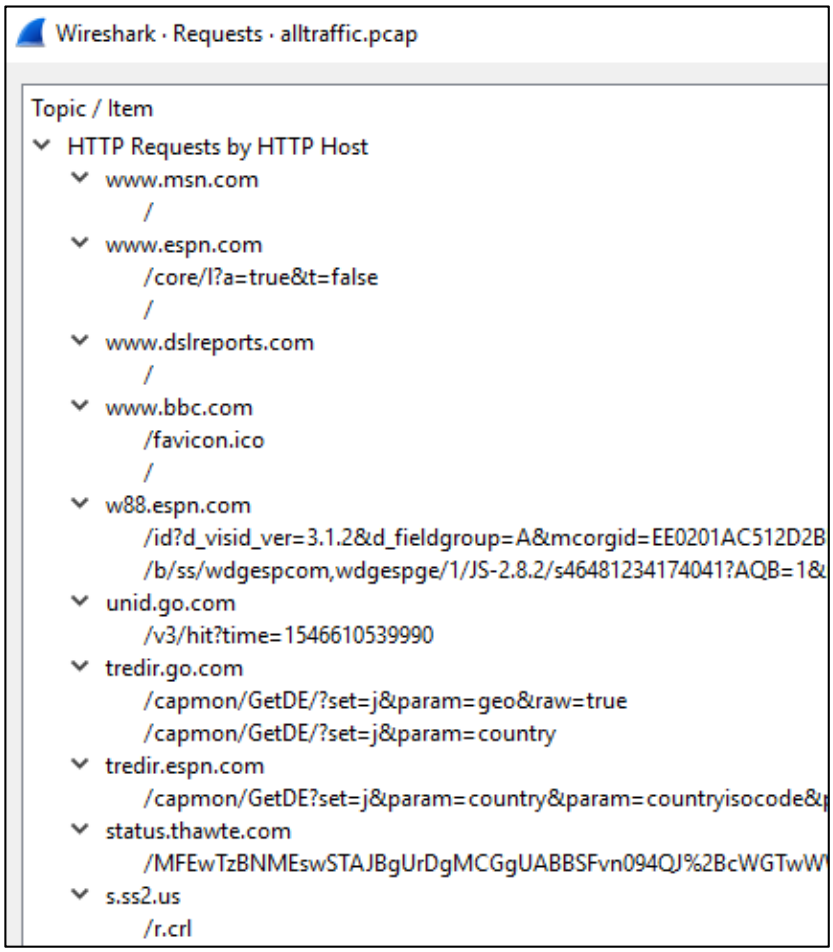
b. Why would it be useful in an investigation to see where data is being **POST**ed to?

_____

_____

_____

10. **HTTP** conversations can be very useful in a wide range of investigations. They can also be voluminous, and reviewing potentially thousands of requests can become overwhelming. **Wireshark** can help in this regard by providing a nice summary of all **HTTP** requests. Under the **Statistics** menu, click **HTTP | Requests.**



At the bottom of the report is a Display filter that allows for entering filters just like you did in the main Wireshark dialog. For example, entering **`http.host == "www.dslreports.com"`** limits the results to only requests where the provided host was found.

There is also a handy button for saving out the report to a text file.

11. In the **Requests** dialog, use the **Display filter** below and answer the following question:

```
http.host == www.dslreports.com
```

    a. What is the **URL** related to a comment that was requested from **www.dslreports.com**?

    _____

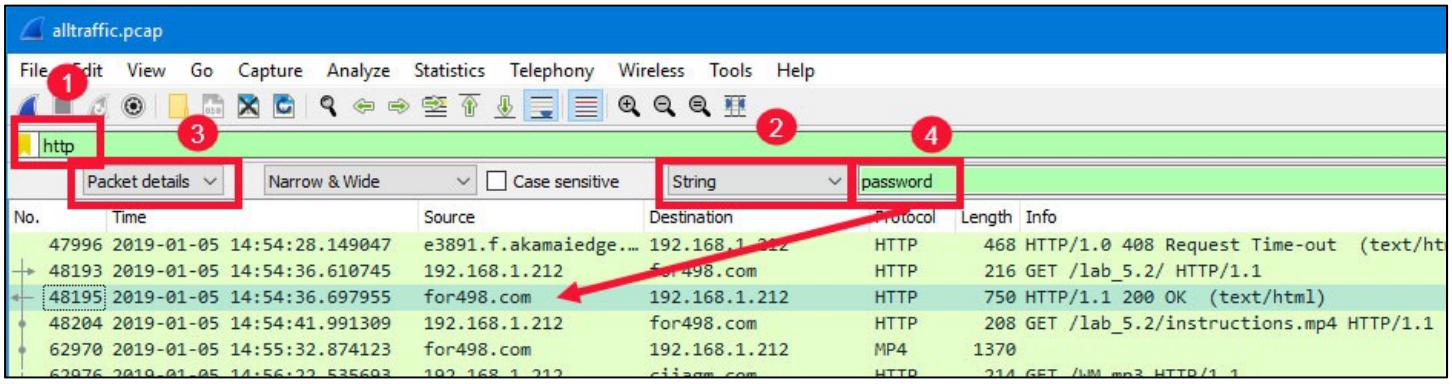
12. Close the **Requests** dialog to get back to the main **Wireshark** interface.

13. You have information that someone accessed a password protected resource and you need to find more information. You suspect that someone accessed a web page that indicates the password, but you do not know the **host** of the web page that was accessed. **Wireshark** allows you to look for strings inside packets, so let's see if we can find the **host**.

    Change the **Display filter** to the following:

```
http
```

14. In **Wireshark**, press **CTRL-F** to bring up the search dialog. Change the rightmost drop down to **String**, then change the leftmost drop down to **Packet details**, in that order. In the textbox, enter **`password`** and press the **Find** button to start the search.



    a. What is the **frame number** where the string was found? (**Hint**: Look at the value in the **No.** column on the far left. The same number is also shown in the details view below the packet list)

    _____

    b. What is the **Source** where the data came from (record both the **hostname** and the **IP address**)? (**Hint**: you can get the **IP address** in the **Internet Protocol Version 4** section of the packet details)

    _____

BONUS: What web server software is running on this host?

_____

15. In the packet details for frame number **48195**, expand the **Line-based text data** section.

    a. What is shown under this section?

    _____

    _____

    b. Are there any file names that may be of interest? What is the file name?

    _____

16. Now that you know the **host** and **IP address**, let's find all the packets that involve that **host**. In the **Display filter** dialog, enter the following filter and press **Enter**.

```
ip.src == 70.32.97.206 and http
```

    a. How many packets were found? Write their frame numbers below.

    _____

17. The first frame, **48195**, is the same as we identified above, but the second frame, **62970** looks interesting too. Select frame **62970** in the packet list.

    a. In the packet list, what is the **Protocol** of this frame?

    _____

18. With frame **62970** selected, look in the packet details. Notice there is a section that matches the protocol used, **MP4**. Notice also that there is a **Hypertext Transfer Protocol** section. Expand the **Hypertext Transfer Protocol** section and answer the following:

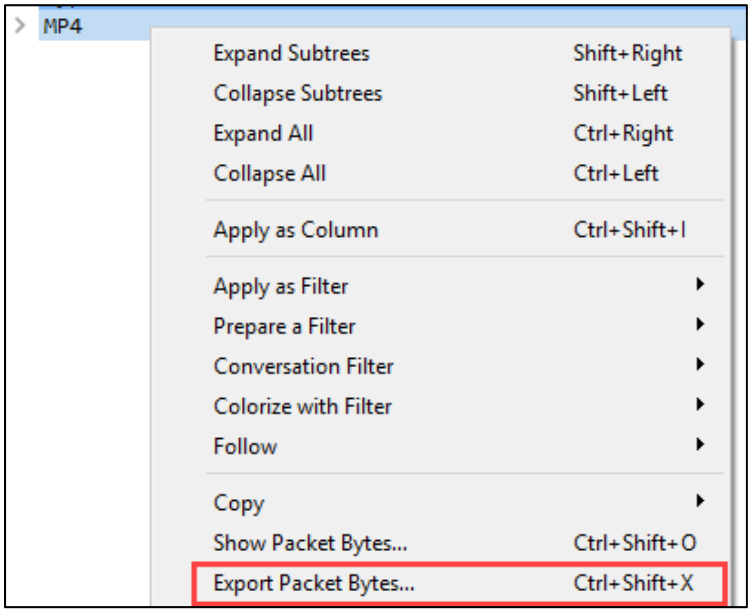
    a. How big is the **MP4** file? (Hint: Look for the **Content-Length** property)

    _____

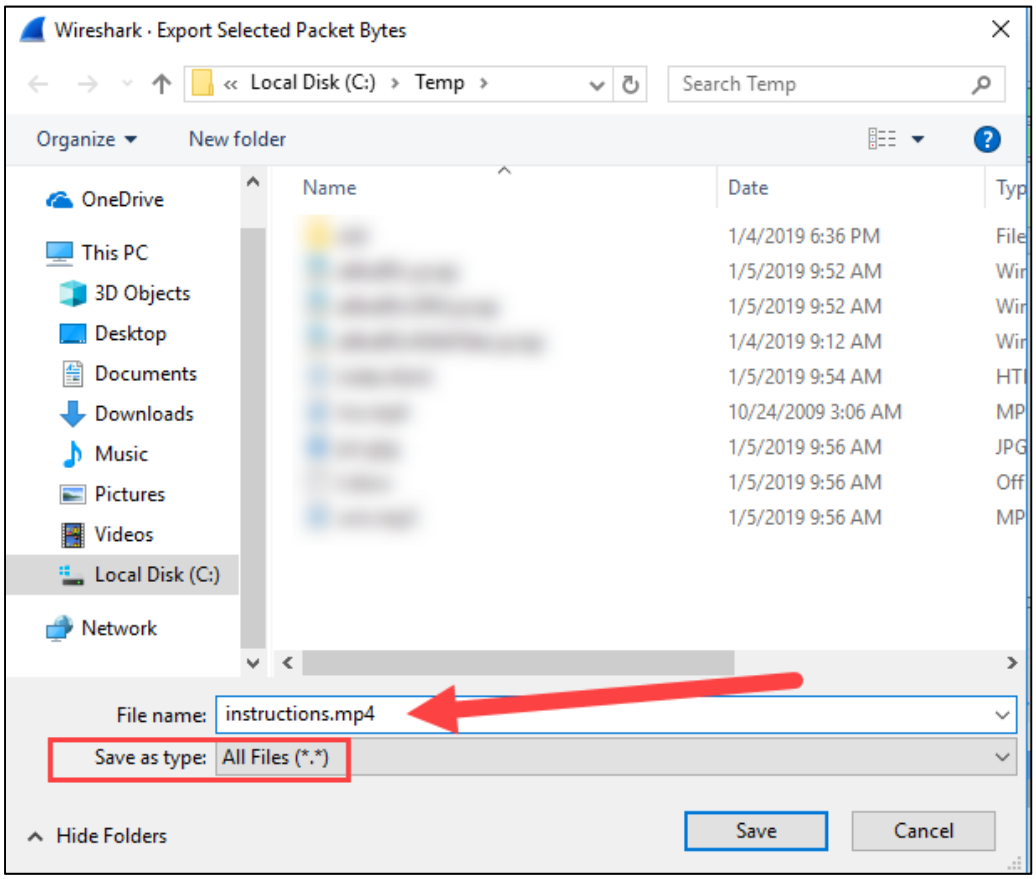
    b. When was the **MP4** last modified? (Hint: Check the **Hypertext Transfer Protocol** section)

    _____

19. **Wireshark** allows for saving out streams of data to a file. To save the **MP4** that was downloaded, right click on the **MP4** section, then choose **Export Packet Bytes…**



This brings up a dialog allowing you to specify the directory and file name to save the data to. Select the **C:\Temp** folder and name the file after the interesting file name you found earlier (**instructions.mp4**). Be sure to change the drop down for **Save as type** to **All Files**.



Click **Save** to write the bytes to the specified file.

20. Use **File Explorer** to navigate to **C:\Temp** and play **instructions.mp4** using your media player of choice. We like **VLC**, but anything should work. After watching the video, answer the following questions.

    a. What just happened?

    _____

    _____

BONUS: Recall from earlier where we initially found out about the **mp4** filename that there was also a reference to a password of some sort. Can you figure out what the plaintext password is?

_____

Next, let's explore the filtered pcap file.

21. Start another copy of **Wireshark**, then load **C:\Network\trafficWithFilter.pcap** via **File → Open.**

22. In the display filter, enter the following filter and press **Enter**.

```
icmp
```

    a. How many packets were found? Why?

    _____

    _____

23. In **DNS** speak, an **A** record is used to point a domain name to an IP address. In many investigations, we want to know not only where data is going to (the domain) but what the IP address is as this provides an investigative lead since whoever controls that IP address may be able to provide additional information about it. In **Wireshark**, **A** records have a type of "1" (one).

    In the **Display filter** dialog, enter the following filter and press **Enter**.

```
dns.qry.type == 1
```

You should see something that looks like this:

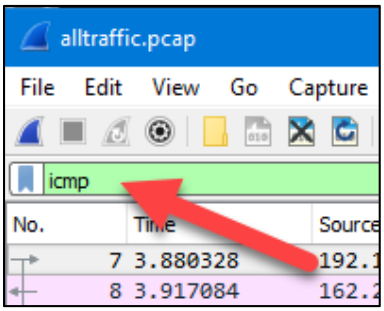
Info
Standard query response 0x8dbf A www.msn.com CNAME www-msn-com.a-0003.a-msedge.net CNAME a-
Standard query response 0x94a4 A sb.scorecardresearch.com A 0.0.0.0
Standard query response 0x0eb3 A c.msn.com A 0.0.0.0
Standard query response 0x8f73 A otf.msn.com A 0.0.0.0
Standard query response 0xaf64 A img.s-msn.com CNAME wildcard.s-msn.com.edgekey.net CNAME e
Standard query response 0x944c A img-s-msn-com.akamaized.net CNAME a1834.dspg2.akamai.net A
Standard query response 0xedd4 A static-spartan-eus-s-msn-com.akamaized.net CNAME a1903.g2.
Standard query response 0x78e0 A images.taboola.com A 0.0.0.0
Standard query response 0x9dd6 A assets.msn.com CNAME assets.msn.com.edgekey.net CNAME e285
Standard query response 0xc134 A sam.msn.com CNAME www.msn.com CNAME www-msn-com.a-0003.a-m
Standard query response 0xf39e A api-s2s.taboola.com A 0.0.0.0
Standard query response 0x010d A login.live.com CNAME login.msa.akadns6.net CNAME vs.login.
Standard query response 0xa515 A tpx.everquote.com A 54.85.17.129 A 54.210.28.164 A 34.206.

This is just scratching the surface of what is possible when it comes to network traffic analysis, but it provides a framework for how to explore packet captures to find relevant data for your investigation as it relates to data being exchanged between computers, including desktops, laptops, mobile, and even IoT devices.

Exercise - Questions with Step-by-Step

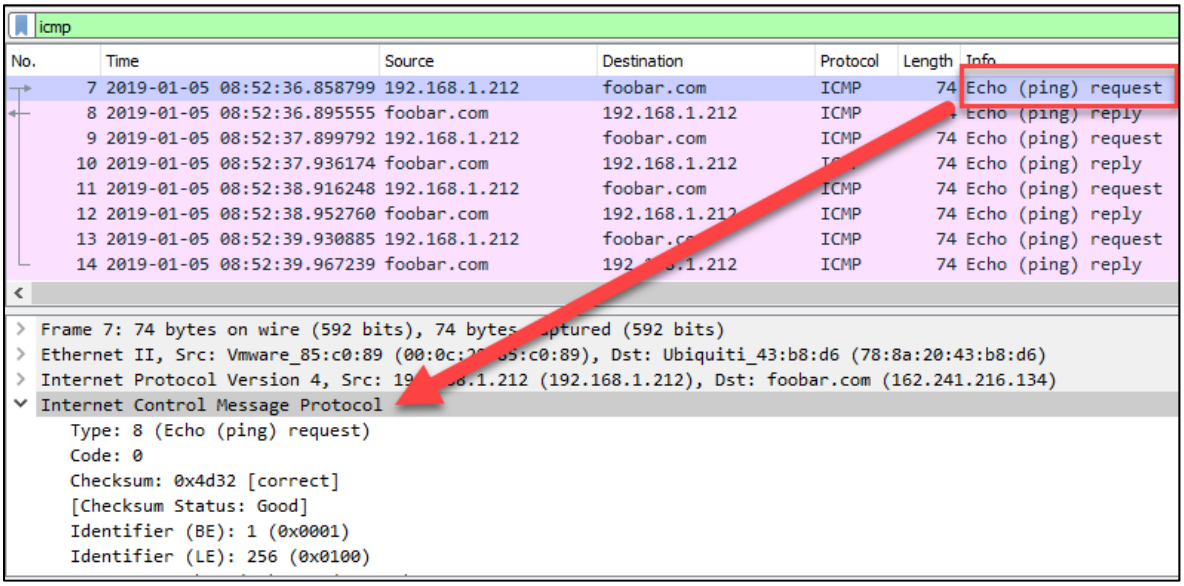
1. Recall in the previous exercise we pinged several domains. The **ping** command uses the **Internet Control Message Protocol (ICMP)** to do its work. We can find these ping requests by using a display filter in **Wireshark**, specifically, the **icmp** expression. In the display filter box, enter the filter below and then press **Enter**.

```
icmp
```



Notice what happened here. All the packets were filtered for only those that match the filter.

2. With the packets now only showing **icmp**, select the first packet in the list and, if necessary, expand the **Internet Control Message Protocol** section.

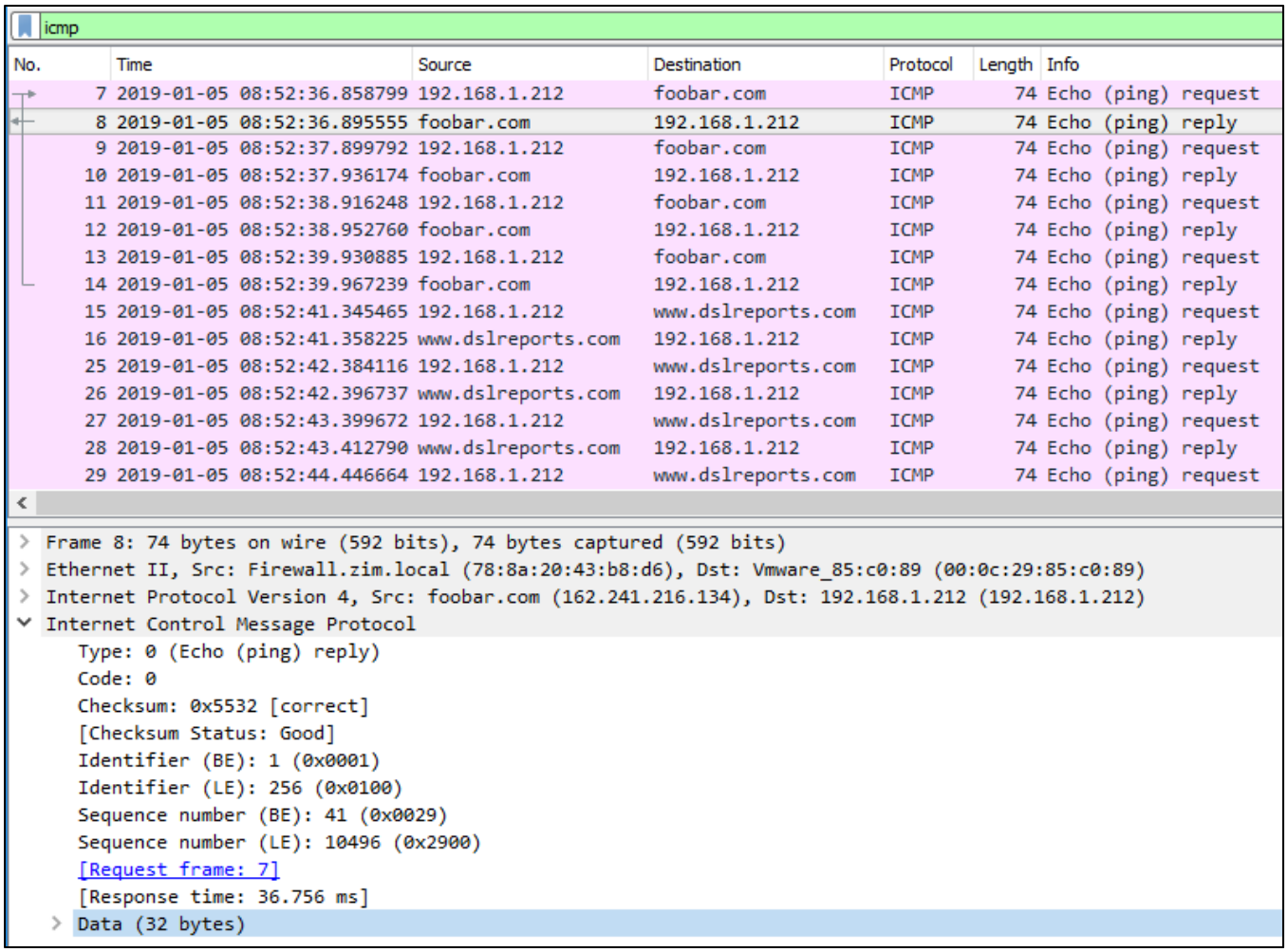


In the packet list, we can see the **Source** of the **ping** (where the request came from), and the **Destination** (the value that was pinged). The **Info** column shows that the first packet is a request, which can also be seen in the details at the bottom.

a. What is the **ICMP** type for a **ping** request?

Type 8. This is shown where the arrow is pointing to in the above image.

3. Next, select the **Echo (ping) reply** (it should be the next packet in the list) for the request we just looked at. When looking for the **reply** (assuming there is one that is), you can find it easily by looking for the packet where the **Source** and **Destination** are reversed, as seen below.



a. What is the **ICMP** type for a **ping** reply?

Type 0 as shown in the above image

b. In frame 8, how long did it take for the host that was pinged to respond? (Hint: Look for the **Response time** property)

36.756 milliseconds (this is right below the Request frame value in the above image)

Also note that, once you are looking at a **reply**, you can see the **Request frame** number of the request. This works the other way too, in that the **request** packet would reference the **reply** frame. This lets you verify you are working on the same "pair" of **request** and **reply**.

4. Replace the display filter with the following and press **Enter**.

```
icmp.type == 0
```

a. What did this filter do?

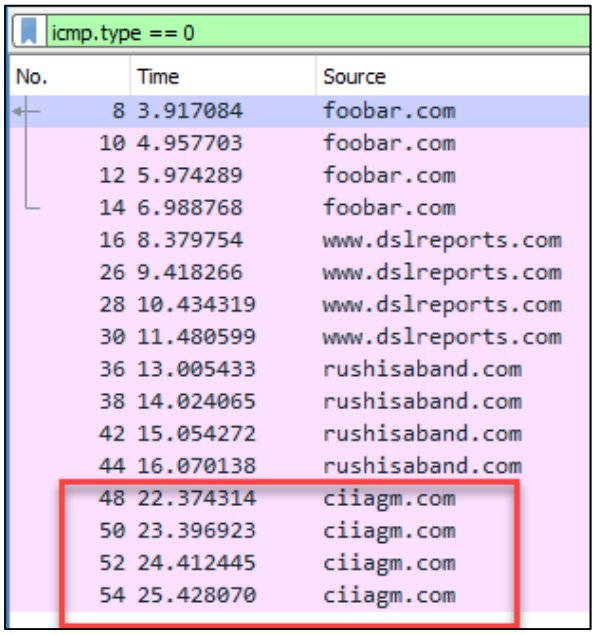
Filters out everything except ping replies.

b. How many packets match the above filter? (Hint: Look in the lower right for the **Displayed** counter)

16

c. Can you find another host that was pinged in addition to the ones specifically mentioned in the **exercise 5.2A**? If so, what is the hostname?

Looking at the list of packets, there is a domain we did not specifically ping; ciiagm.com.



NOTE: We initially pinged the following hosts: **foobar.com**, **dslreports.com**, and **rushisaband.com**

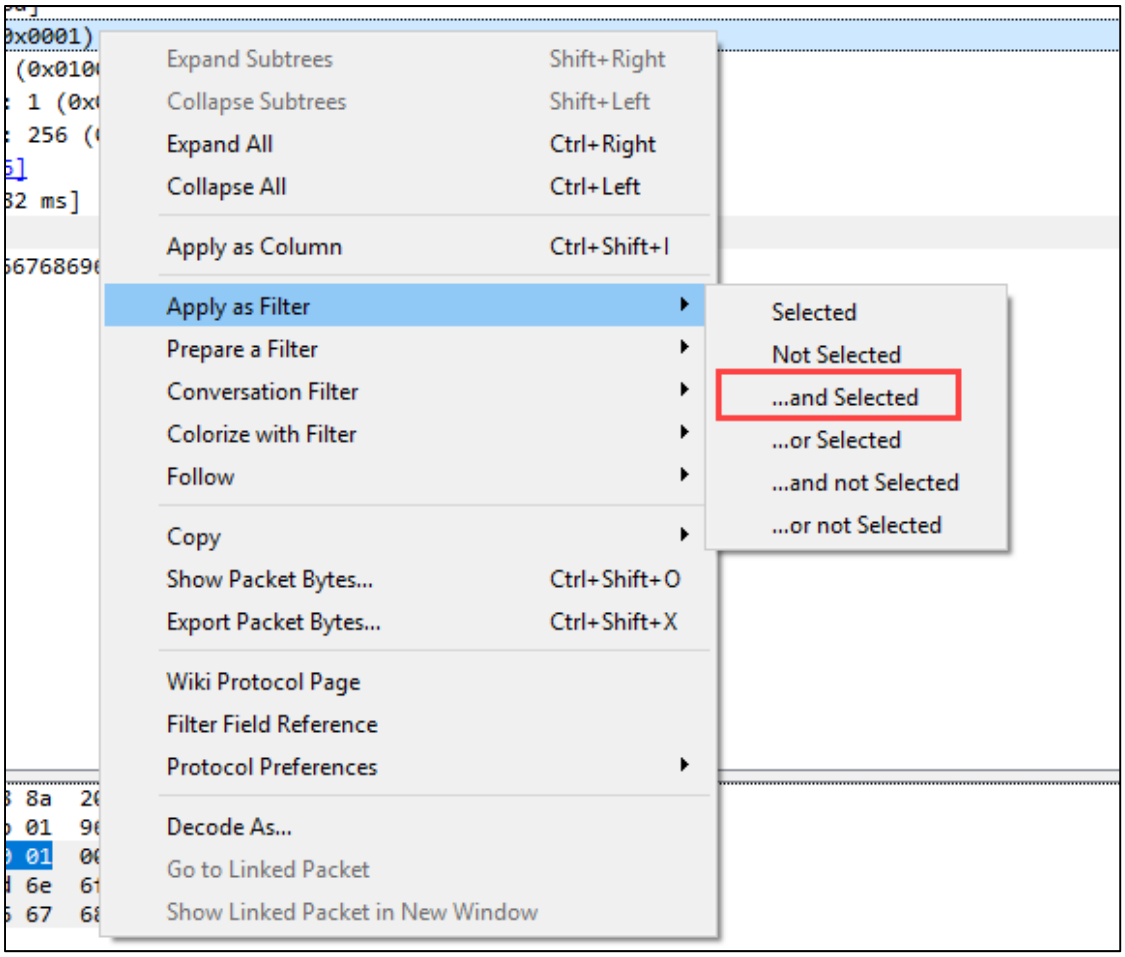
d. If the host responded to the ping, how long did it take?

Approximately 53 milliseconds. This is visible in the details pane in the middle after selecting one of the rows containing ciiagm.com

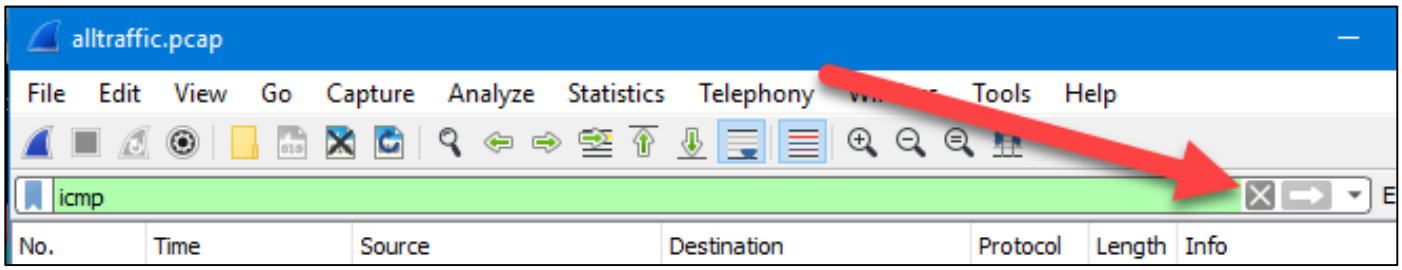
Remember, you can drag and drop a value from the information displayed at the bottom to quickly filter on anything you like.

PROTIP: If you want to *add* additional filter criteria, select the property to add, then right click and choose **Apply as Filter | …and Selected**

Here is an example of this:

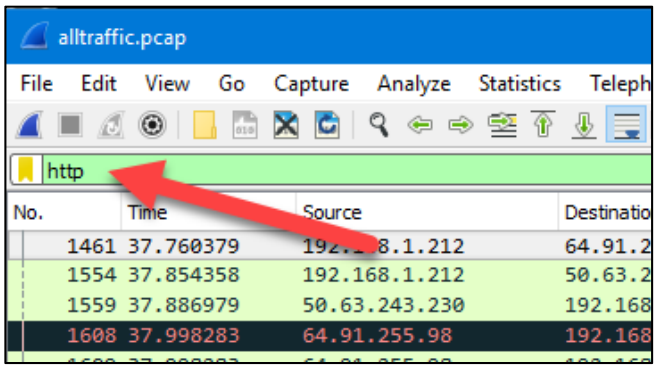


5. Clear any display filters in **Wireshark** by clicking the **X** as shown below.



6. In the display filter dialog, enter the following filter and press **Enter**. Notice what the **Protocol** column now shows.

```
http
```



   a. How many packets were found? (Look in the status bar on the bottom)

      201 packets

7. Spend a few minutes reviewing the kinds of information available in various **HTTP** requests.

8. For most of the **HTTP** packets, the **Info** pane starts with something like "**GET**" or "**POST**". These are **HTTP** "verbs" that describe interaction with a web server. A "**GET**" would be a request to get data from a web site, whereas a "**POST**" would be sending data to a web site.

   To find all the **GET** requests, change the display filter to the following and press **Enter**.

```
http.request.method == "GET"
```

   a. How many **GET** requests were found? (See the **Displayed** counter in lower right again)

      93

9. Change the display filter to find all the "**POST**" methods using the following filter:

```
http.request.method == "POST"
```

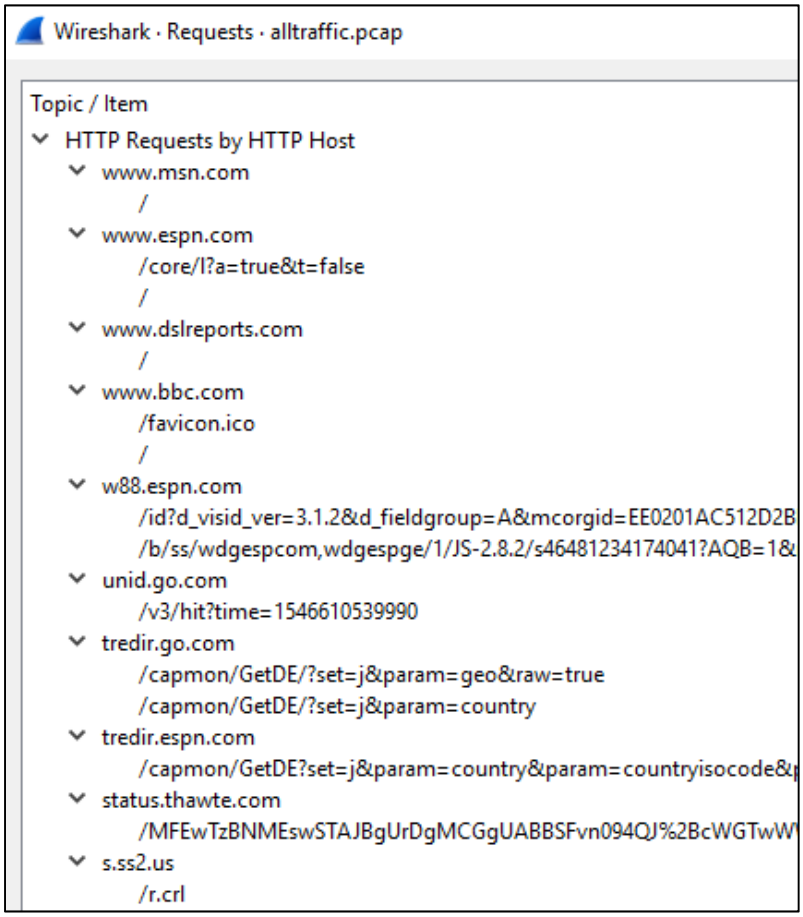
   a. How many **POST** requests were found? (See the **Displayed** value in lower right again)

      2

b. Why would it be useful in an investigation to see where data is being **POST**ed to?

This serves as a good indicator of where data is being sent to, either intentionally, or due to malicious software running on a host.

10. **HTTP** conversations can be very useful in a wide range of investigations. They can also be voluminous, and reviewing potentially thousands of requests can become overwhelming. **Wireshark** can help in this regard by providing a nice summary of all **HTTP** requests. Under the **Statistics** menu, click **HTTP | Requests**.



At the bottom of the report is a Display filter that allows for entering filters just like you did in the main Wireshark dialog. For example, entering **`http.host == "www.dslreports.com"`** limits the results to only requests where the provided host was found.

There is also a handy button for saving out the report to a text file.

11. In the **Requests** dialog, use the **Display filter** below and answer the following question:

```
http.host == www.dslreports.com
```

    a. What is the **URL** related to a comment that was requested from **www.dslreports.com**?

    /comment/1519/95873

    This is shown in the HTTP requests report under the www.dslreports.com group.

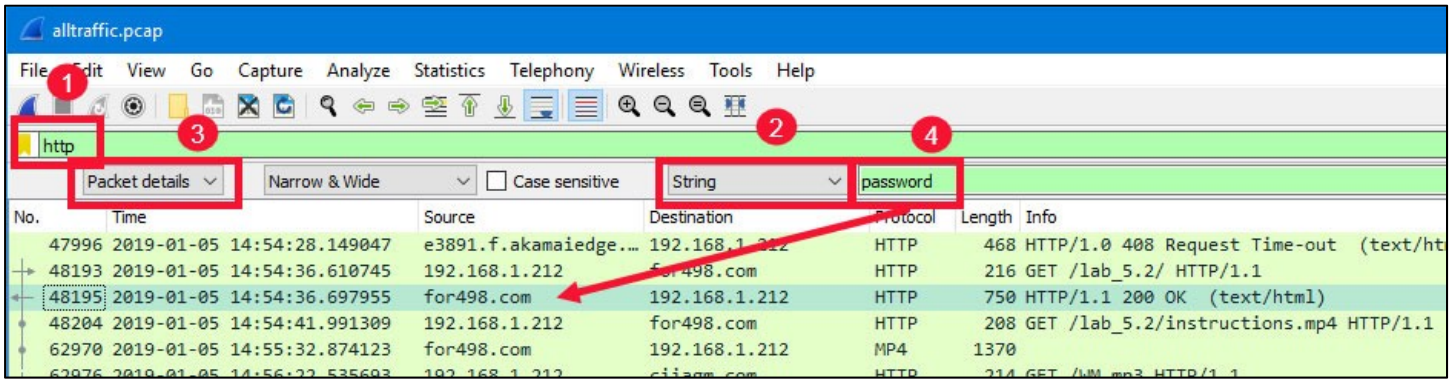
12. Close the **Requests** dialog to get back to the main **Wireshark** interface.

13. You have information that someone accessed a password protected resource and you need to find more information. You suspect that someone accessed a web page that indicates the password, but you do not know the **host** of the web page that was accessed. **Wireshark** allows you to look for strings inside packets, so let's see if we can find the **host**.

    Change the **Display filter** to the following:

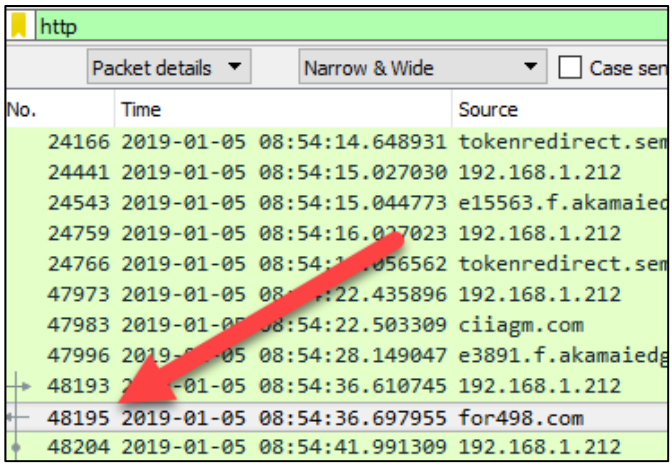
```
http
```

14. In **Wireshark**, press **CTRL-F** to bring up the search dialog. Change the rightmost drop down to **String**, then change the leftmost drop down to **Packet details**, in that order. In the textbox, enter **`password`** and press the **Find** button to start the search.



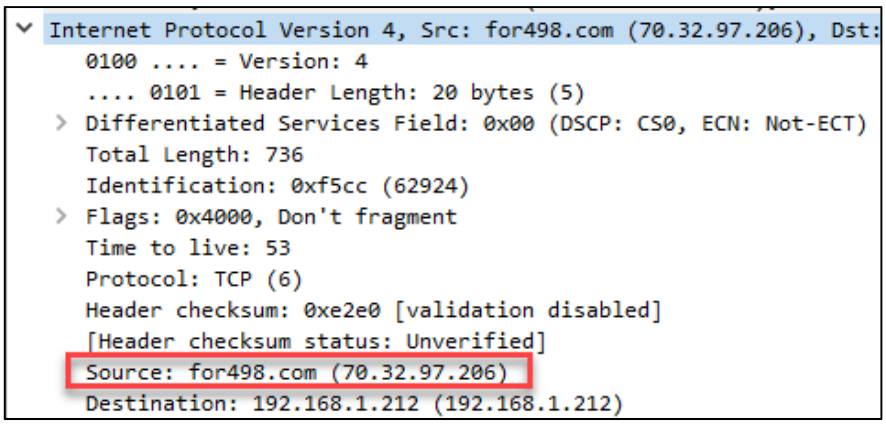
a. What is the **frame number** where the string was found? (**Hint**: Look at the value in the **No.** column on the far left. The same number is also shown in the details view below the packet list)

48195



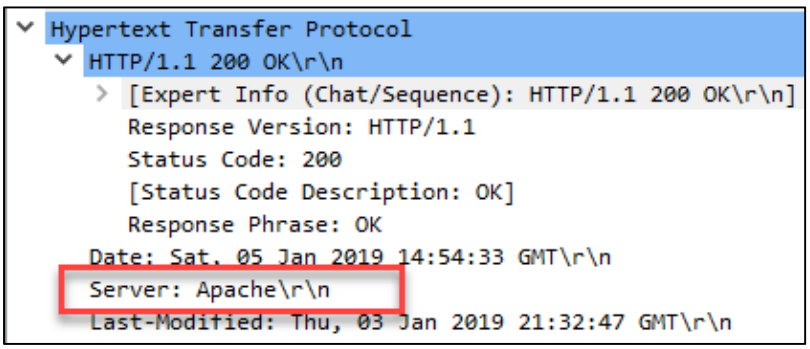
b. What is the **Source** where the data came from (record both the **hostname** and the **IP address**)? (**Hint**: you can get the **IP address** in the **Internet Protocol Version 4** section under packet details).

The hostname is for498.com and the IP address is 70.32.97.206



BONUS: What web server software is running on this host?

Under the Hypertext Transfer Protocol section, the server is shown as Apache.



15. In the packet details for frame number **48195**, expand the **Line-based text data** section.

    a. What is shown under this section?

    The HTML code that was sent to the browser when this request was made

    b. Are there any file names that may be of interest? What is the file name?

    There is a file named *instructions.mp4* mentioned in the HTML code

16. Now that you know the **host** and **IP address**, let's find all the packets that involve that **host**. In the **Display filter** dialog, enter the following filter and press **Enter**.

```
ip.src == 70.32.97.206 and http
```

    a. How many packets were found? Write their frame numbers below

    2 packets, frame numbers are 48195 and 62970

17. The first frame, **48195**, is the same as we identified above, but the second frame, **62970** looks interesting too. Select frame **62970** in the packet list.

    a. In the packet list, what is the **Protocol** of this frame?

    MP4

18. With frame **62970** selected, look in the packet details. Notice there is a section that matches the protocol used, **MP4**. Notice also that there is a **Hypertext Transfer Protocol** section. Expand the **Hypertext Transfer Protocol** section and answer the following:

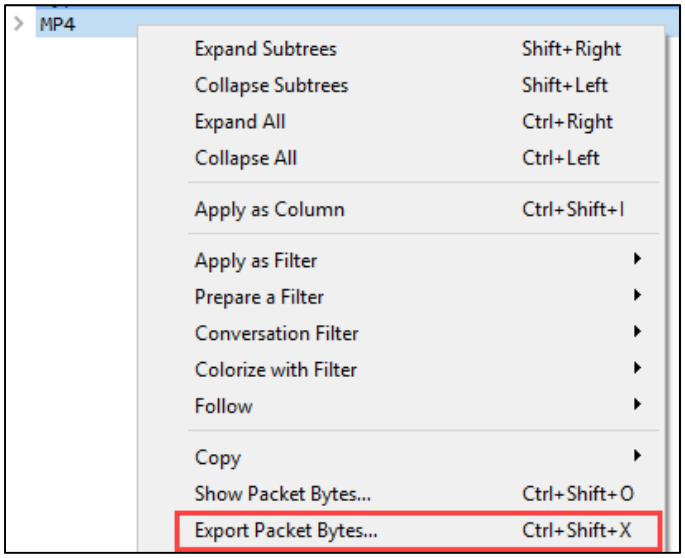
    a. How big is the **MP4** file? (**Hint**: Look for the **Content-Length** property)

    17950263 bytes

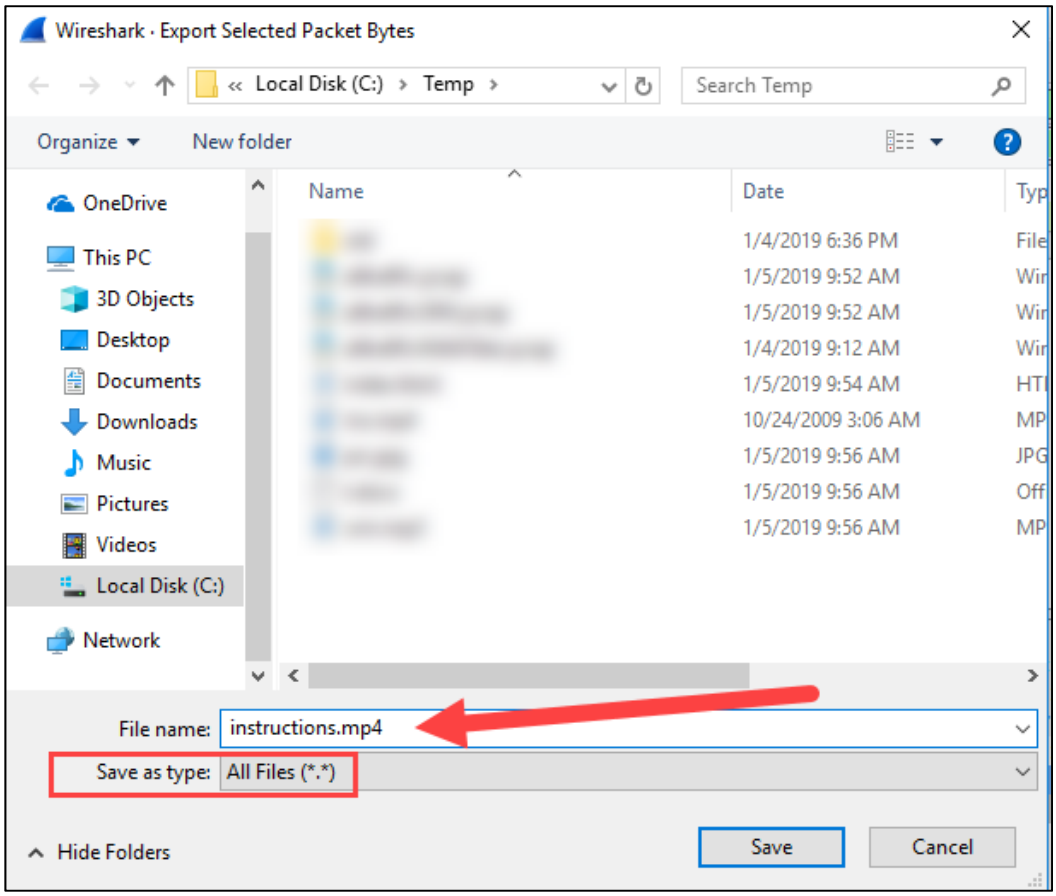
    b. When was the **MP4** last modified? (**Hint**: Check the **Hypertext Transfer Protocol** section)

    Wed, 19 Dec 2018 19:16:27 GMT

19. **Wireshark** allows for saving out streams of data to a file. To save the **MP4** that was downloaded, right click on the **MP4** section, then choose **Export Packet Bytes…**



This brings up a dialog allowing you to specify the directory and file name to save the data to. Select the **C:\Temp** folder and name the file after the interesting file name you found earlier (**instructions.mp4**). Be sure to change the drop down for **Save as type** to **All Files**.



Click **Save** to write the bytes to the specified file.

20. Use **File Explorer** to navigate to **C:\Temp** and play **instructions.mp4** using your media player of choice. We like **VLC**, but anything should work. After watching the video, answer the following questions.

    a. What just happened?

       I was Rickrolled

BONUS: Recall from earlier where we initially found out about the **mp4** filename that there was also a reference to a password of some sort. Can you figure out what the plaintext password is?

There is an indication of the password in the HTML. The value shown is:

110 101 118 101 114 103 111 110 110 97 103 105 118 101 121 111 117 117 112

Treating each number as an ASCII value, we can convert each to its character equivalent, which shows the password is:

nevergonnagiveyouup

You can use a website like https://www.branah.com/ascii-converter to do this conversion online.

Next, let's explore the filtered pcap file.

21. Start another copy of **Wireshark**, then load **C:\Network\trafficWithFilter.pcap** via **File → Open.**

22. In the display filter, enter the following filter and press **Enter**.

```
icmp
```

    a. How many packets were found? Why?

       Out of 184 total packets that were recorded, zero are displayed. This is due to the filter condition we set that only captured traffic on port 53.

23. In **DNS** speak, an **A** record is used to point a domain name to an IP address. In many investigations, we want to know not only where data is going to (the domain) but what the IP address is as this provides an investigative lead since whoever controls that IP address may be able to provide additional information about it. In **Wireshark**, **A** records have a type of "1" (one).

    In the **Display filter** dialog, enter the following filter and press **Enter**.

```
dns.flags.response == 1 and dns.qry.type == 1
```

You should see something that looks like this:

Info
Standard query response 0x8dbf A www.msn.com CNAME www-msn-com.a-0003.a-msedge.net CNAME a-
Standard query response 0x94a4 A sb.scorecardresearch.com A 0.0.0.0
Standard query response 0x0eb3 A c.msn.com A 0.0.0.0
Standard query response 0x8f73 A otf.msn.com A 0.0.0.0
Standard query response 0xaf64 A img.s-msn.com CNAME wildcard.s-msn.com.edgekey.net CNAME e
Standard query response 0x944c A img-s-msn-com.akamaized.net CNAME a1834.dspg2.akamai.net A
Standard query response 0xedd4 A static-spartan-eus-s-msn-com.akamaized.net CNAME a1903.g2.
Standard query response 0x78e0 A images.taboola.com A 0.0.0.0
Standard query response 0x9dd6 A assets.msn.com CNAME assets.msn.com.edgekey.net CNAME e285
Standard query response 0xc134 A sam.msn.com CNAME www.msn.com CNAME www-msn-com.a-0003.a-m
Standard query response 0xf39e A api-s2s.taboola.com A 0.0.0.0
Standard query response 0x010d A login.live.com CNAME login.msa.akadns6.net CNAME vs.login.
Standard query response 0xa515 A tpx.everquote.com A 54.85.17.129 A 54.210.28.164 A 34.206.

This is just scratching the surface of what is possible when it comes to network traffic analysis, but it provides a framework for how to explore packet captures to find relevant data for your investigation as it relates to data being exchanged between computers, including desktops, laptops, mobile, and even IoT devices.

Exercise—Key Takeaways

- Wireshark provides detailed and full insight into network packets and allows for robust filtering to find items of interest.
- Leveraging one or more tools to automate network analysis can significantly speed up getting to answers.

Exercise 5.2C—PCAP Analysis – Network Miner

Background

Network traffic is a rich source of forensic information that allows for discovering things such as computers that are engaged in conversations, files being transferred, host name lookups, and more. It is important for forensic practitioners to know how to interpret network traffic in order to be able to extract this kind of detail.

Objectives

Use NetworkMiner to get an overview of conversations, files transferred, and more.

Exercise Preparation

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

   In the previous lab, we saw that **Wireshark** is a powerful tool with many options to drill into network traffic. In many cases, however, it will not be the first tool you might use in an investigation, due to the amount of work it takes to get to the answers you are looking for.

   If this is the case, why did we start with **Wireshark** vs. some other tool? Remember that it is important to understand where the data lives and how the data is organized vs. only how to get answers by running software. This is a key difference between being able to only use tools vs. understanding the data itself, regardless of the tool that interprets it. By learning how things work and parsing things in a more manual fashion with **Wireshark**, you will have a better understanding of what is going on in more automated tools. You will also be in a much better position to know what to do (and have the skills to be able to do so) should automated tools fail.

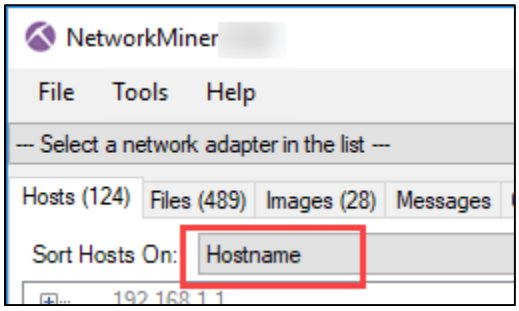
   With this in mind, let's look at how **NetworkMiner** works and see how it can be used to answer important questions in an investigation.

3. Start **NetworkMiner** from the shortcut in the **Network Tools** fence on the **Desktop** and load the **C:\Network\alltraffic.pcap** file from the **File → Open** menu.

4. **NetworkMiner** will then parse the pcap file and start to populate the various sections in the interface with data. While it is running, the interface will appear to be stuck, so be patient, especially for larger pcap files.

Exercise – Questions

1. When processing is complete, click on the **Hosts** tab. Now select **Hostname** from the dropdown menu in the **Sort Hosts On:** field.



2. Expand the entry for IP address **192.168.1.51**.

   a. Who manufactured the **network interface card (NIC)** in this computer?

   _____

3. Expand the entry for IP address **192.168.1.212**, then expand the **Host Details** section.

   a. What is the value for **Web Browser User-Agent 3**?

   _____

   b. List some of the **DNS names** this computer performed a lookup on (3-4 or so is fine):

   _____

   _____

c. What **NetBIOS** name was queried for IP address **192.168.1.212**?

_____

4. Expand the **Outgoing sessions** section for IP address **192.168.1.212**.

   Locate the entry for the host **ciiagm.com** (its IP address starts with 67) and expand the details for this host by double-clicking the host or using the **+** sign to the left.

   a. What operating system is running on **ciiagm.com**?

   _____

   b. How many **TCP** sessions were opened to **ciiagm.com**?

   _____

   c. What is the **IP address** of **ciiagm.com**?

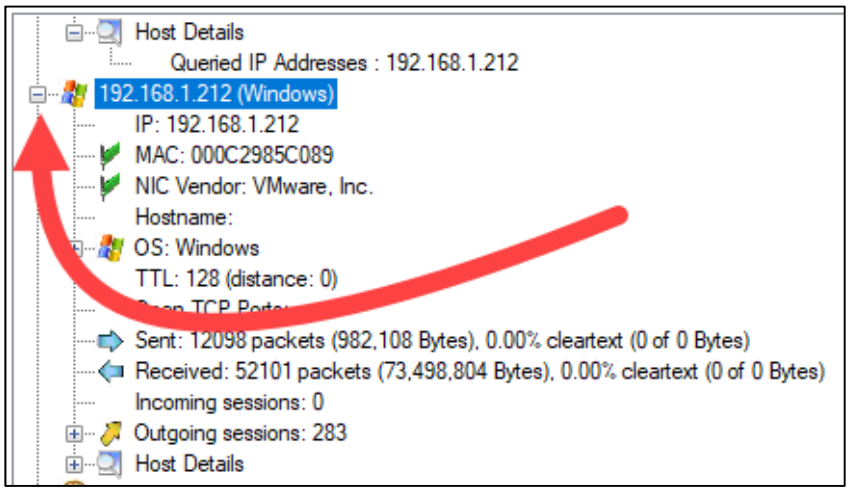
   _____

   d. When did the session with the *largest* amount of data **sent** start and end to **ciiagm.com**?

   _____

   _____

5. Collapse the details for IP address **192.168.1.212** by clicking the **minus** sign to the left of it.



6. Locate the **IP address** for **ciiagm.com** in the list (67.222.18.45) and expand its details.

   a. What web server is running on the host?

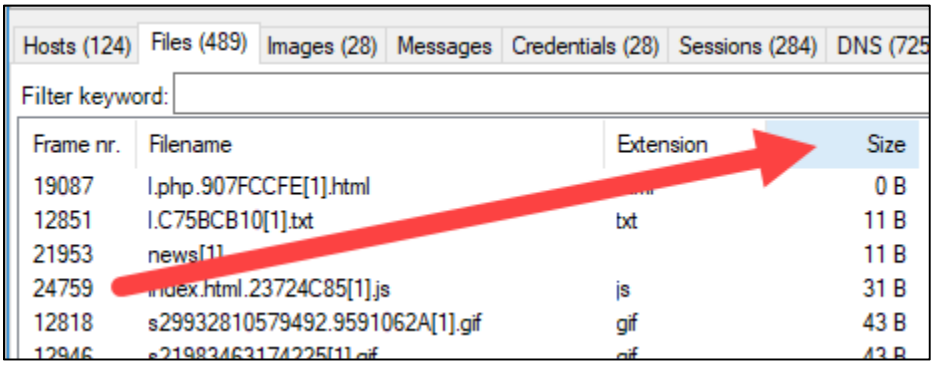
   _____

7. Click on the **Files** tab and take a minute to scroll through the list of transferred files that were found.

   a. How many total files did **NetworkMiner** find?

   _____

8. The data on the **Files** tab can be voluminous, but luckily, every column in the list is sortable. Click on the **Size** column header to sort the data by size, from smallest to largest.



9. Since the largest files are at the bottom, scroll to the bottom.

   a. What is the name of the largest file?

   _____

   b. How big was it?

   _____

10. Using the horizontal scroll bar at the bottom, scroll to the right to see additional details, like the **Details** column, which contains the full URI used to retrieve the file.

    a. What is the **Timestamp** for when **instructions.mp4** was transferred?

    _____

11. Right-click on **instructions.mp4** to display a context menu.

    a. What are the first 5 characters of the **SHA-1** hash of **instructions.mp4**?

    _____

    b. Right-click **instructions.mp4** and choose **Open file**. What just happened?

    _____

12. With the **Files** data still sorted by **Size**, answer the following questions.

    a. What is the name of the second largest file?

    _____

    b. What host was it downloaded from?

    _____

13. Right-click on **WM.mp3** and select **Open file**.

    a. What time does work start?

    _____

    b. Why doesn't the singer have time for living?

    _____

14. Right-click on **WM.mp3** and select **Open folder** from the context menu. This folder contains all the files downloaded from the host.

    a. How many files in total were downloaded?

    _____

    b. What other files besides **WM.mp3** were downloaded?

    _____

    _____

    _____

    In the **File Explorer** instance that starts when you selected **Open folder**, navigate up two directory levels. Notice how there is a directory for each IP address where files were transferred. This can be a useful way to find quick wins.

    NOTE: If the PCAP file contained anything malicious, any security software running on the computer would possibly find it as **NetworkMiner** extracts and saves the malware to disk. Pay attention to your security software for any alerts.

15. Open **image.jpg** by double-clicking on the file in **File Explorer**.

    a. Who is driving the golf cart?

    _____

    b. Do you recognize anyone else from the photo?

    _____

    c. Locate this same photo on the **Images** tab. How many bytes is the file?

    _____

16. Right-click **Gazette.docx** and open the file by clicking **Open with EZViewer**. After reading **Gazette.docx**, you want to know when this file was downloaded. Look for this document in the list of files.

    a. What is the **Timestamp** for when **Gazette.docx** was transferred?

    _____

    BONUS: Who was the last person to save **Gazette.docx**?

    _____

17. Click on the **DNS** tab.

    a. What were the first three unique hosts that were resolved?

    _____

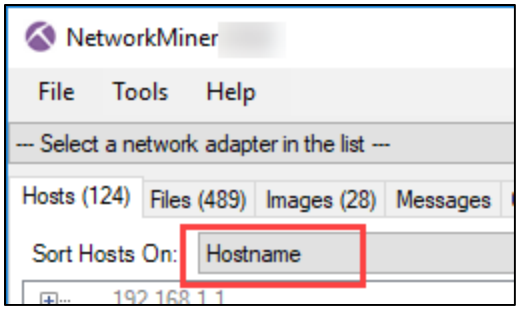
    _____

    _____

18. Spend a few minutes browsing around the other tabs in the **NetworkMiner** interface. The **Credentials** tab can be very useful depending on the traffic in the PCAP.

NetworkMiner does a lot of heavy lifting for us to get answers quickly, including files transferred and DNS lookups. It also extracts out files and organizes them by IP address that they originated from. This is very useful, because often when you find one file of interest, other files from the same location will also be of interest.

1. When processing is complete, click on the **Hosts** tab. Now select **Hostname** from the dropdown menu in the **Sort Hosts On:** field.



2. Expand the entry for IP address **192.168.1.51**.

    a. Who manufactured the network interface card (NIC) in this computer?

    ASUSTek COMPUTER INC.

3. Expand the entry for IP address **192.168.1.212**, then expand the **Host Details** section.

    a. What is the value for **Web Browser User-Agent 3**?

    Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.17134.407

    b. List some of the **DNS names** this computer performed a lookup on (3-4 or so is fine):

    Foobar.com, rushisaband.com, ciiagm.com, www.msn.com, and so on.

    c. What **NetBIOS** name was queried for IP address **192.168.1.212**?

    DESKTOP-CFPG3U4<1C>

4. Expand the **Outgoing sessions** section for IP address **192.168.1.212**.

    Locate the entry for the host **ciiagm.com** (its IP address starts with 67) and expand the details for this host by double-clicking the host or using the **+** sign to the left.

    a. What operating system is running on **ciiagm.com**?

    Linux

    b. How many **TCP** sessions were opened to **ciiagm.com**?

    5 sessions

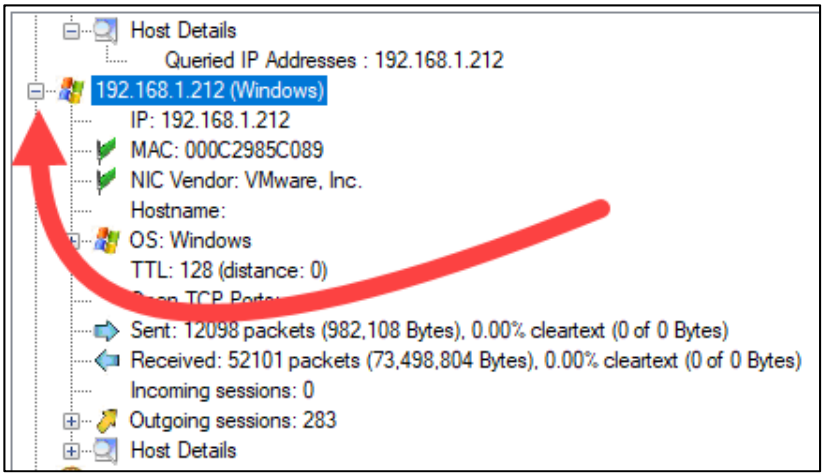
c. What is the **IP address** of **ciiagm.com**?

67.222.18.45

d. When did the session with the *largest* amount of data **sent** start and end to **ciiagm.com**?

The third session sent 928350 bytes. It started at 2019-01-05 14:56:22 and ended at 14:56:31 UTC

5. Collapse the details for IP address **192.168.1.212** by clicking the **minus** sign to the left of it.



6. Locate the **IP address** for **ciiagm.com** in the list (67.222.18.45) and expand its details.

a. What web server is running on the host?

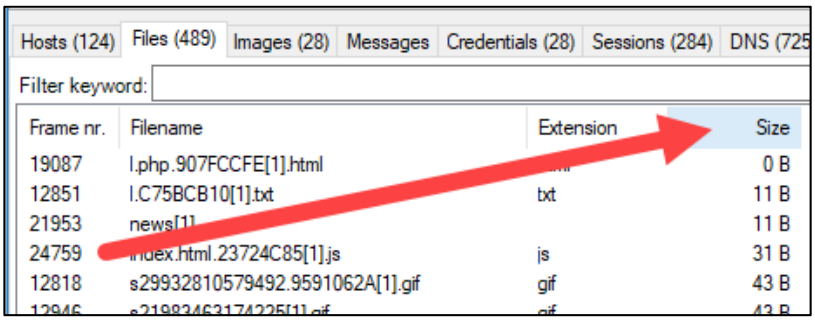
Under the Host Details, it shows Apache is running

7. Click on the **Files** tab and take a minute to scroll through the list of transferred files that were found.

a. How many total files did **NetworkMiner** find?

489 files

8. The data on the **Files** tab can be voluminous, but luckily, every column in the list is sortable. Click on the **Size** column header to sort the data by size, from smallest to largest.



9. Since the largest files are at the bottom, scroll to the bottom.

    a. What is the name of the largest file?

    Instructions.mp4

    b. How big was it?

    17950263 bytes

10. Using the horizontal scroll bar at the bottom, scroll to the right to see additional details, like the **Details** column, which contains the full URI used to retrieve the file.

    a. What is the **Timestamp** for when **instructions.mp4** was transferred?

    2019-01-05 14:54:41 UTC

11. Right-click on **instructions.mp4** to display a context menu.

    a. What are the first 5 characters of the **SHA-1** hash of **instructions.mp4**?

    Using the context menu option to generate the hash, the SHA-1 starts with 2c08a

    b. Right-click **instructions.mp4** and choose **Open file**. What just happened?

    Rickrolled…again.

12. With the **Files** data still sorted by **Size**, answer the following questions.

    a. What is the name of the second largest file?

    WM.mp3.mpeg

    b. What host was it downloaded from?

    ciiagm.com

13. Right-click on **WM.mp3** and select **Open**.

    a. What time does work start?

    Work starts at 9

    b. Why doesn't the singer have time for living?

    He's got no time for livin' because he's workin' all the time

14. Right-click on **WM.mp3** and select **Open folder** from the context menu. This folder contains all the files downloaded from the host.

    a. How many files in total were downloaded?

       4 files

    b. What other files besides **WM.mp3** were downloaded?

       Gazette.docx
       Image.jpg
       Index.html

    In the **File Explorer** instance that starts when you selected **Open folder**, navigate up two directory levels. Notice how there is a directory for each IP address where files were transferred. This can be a useful way to find quick wins.

    NOTE: If the PCAP file contained anything malicious, any security software running on the computer would possibly find it as **NetworkMiner** extracts and saves the malware to disk. Pay attention to your security software for any alerts.

15. Open **image.jpg** by double-clicking on the file in **File Explorer**.

    a. Who is driving the golf cart?

       Geddy Lee

    b. Do you recognize anyone else from the photo?

       None other than course co-author, Kevin Ripa!

    c. Locate this same photo on the **Images** tab. How many bytes is the file?

       254,997 Bytes

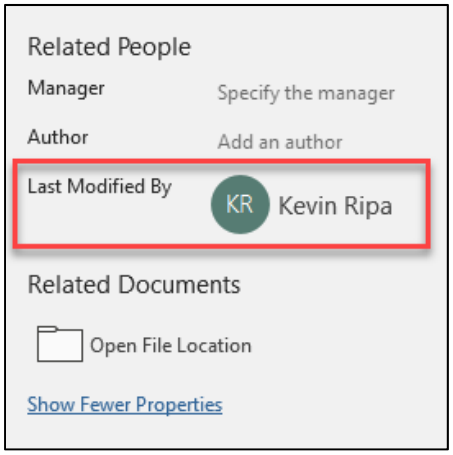
16. Right-click **Gazette.docx** and open the file by clicking **Open with EZViewer**. After reading **Gazette.docx**, you want to know when this file was downloaded. Look for this document in the list of files.

    a. What is the **Timestamp** for when **Gazette.docx** was transferred?

       2019-01-05 14:56:37 UTC

BONUS: Who was the last person to save **Gazette.docx**?

Looking at the metadata for the document in Word, we can see it is Kevin Ripa



ExifTool could also be used to find this. To see this, open a new PowerShell window and type the following command:

```
exiftool.exe -f
"C:\Tools\NetworkMiner\AssembledFiles\67.222.18.45\TCP-
80\Gazette.docx"
```

17. Click on the **DNS** tab.

    a. What were the first three unique hosts that were resolved?

    Foobar.com
    Rushisaband.com
    Ciiagm.com

18. Spend a few minutes browsing around the other tabs in the **NetworkMiner** interface. The **Credentials** tab can be very useful depending on the traffic in the PCAP.

NetworkMiner does a lot of heavy lifting for us to get answers quickly, including files transferred and DNS lookups. It also extracts out files and organizes them by IP address where they originated from. This is very useful, because often when you find one file of interest, other files from the same location will also be of interest.

Exercise—Key Takeaways

- NetworkMiner provides an easy way to inspect the contents of a PCAP file.
- Being able to see and interact with files transferred over the network can provide additional context and leads in an investigation.

This page intentionally left blank.

Optional - Out of Class
Exercise 5.2D—PCAP Analysis – CLI Tools

Background

Network traffic is a rich source of forensic information that allows for the discovering of things such as computers that are engaged in conversations, files being transferred, host name lookups, and more. It is important for forensic practitioners to know how to interpret network traffic in order to be able to extract this kind of detail.

Objectives

- Use tshark to inspect pcap data
- Use passivedns to extract out all DNS related information

Exercise Preparation

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Open a **PowerShell** window and navigate to **C:\Program Files\Wireshark**

Exercise – Questions

NOTE: A great reference for all the possible fields is available at **http://for498.com/ovepr**

Wireshark is an excellent way to explore pcap data and get familiar with how filtering works, but in some cases, you may want to slice and dice packets in a more efficient manner. Enter **tshark**!

The good news is that once you develop some useful display filters using **Wireshark**, you can transfer that directly to **tshark**. Additionally, **tshark** allows us to extract out only certain bits of information from packets. Let's see how we can do this.

1. Go to the **PowerShell** window you previously opened during the preparation of this exercise.

```
cd 'C:\Program Files\Wireshark'
```

2. In **PowerShell**, enter the following command, then press **Enter.**

```
.\tshark.exe -r C:\Network\trafficWithFilter.pcap -Y
"dns.flags.response == 1" -T fields -e frame.time -e dns.qry.name
-e dns.a
```

So, what is going on here?

Value	Meaning
`-Y "dns.flags.response == 1"`	Only include DNS responses (vs requests)
`-T fields`	Output format. **`fields`** allows us to only include certain properties, designated by **`-e`**
`-e frame.time`	The time of the request
`-e dns.qry.name`	The domain that was looked up
`-e dns.a`	The answer(s)

3. Repeat the above command, but redirect the output to a text file.

```
.\tshark.exe -r C:\Network\trafficWithFilter.pcap -Y
"dns.flags.response == 1" -T fields -e frame.time -e dns.qry.name
-e dns.a > C:\Temp\dnsLookups.txt
```

4. Open **C:\Temp\dnsLookups.txt** in a text editor and review the data.

5. In **PowerShell**, enter the following command and press **Enter**.

```
.\tshark.exe -r C:\Network\trafficWithFilter.pcap -Y "http"
```

   a. How many lines of data were displayed? Why?

   _____

   _____

6. Because we limited the packets that were recorded in **trafficWithFilter.pcap** to only traffic related to port 53, we do not see any **HTTP** related traffic as nothing from that display filter was found in the capture.

   Let's see what we can find in our other packet capture, by repeating the command against **alltraffic.pcap.**

   In **PowerShell**, enter the following command and press **Enter**.

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "http"
```

   a. Did **tshark** display **HTTP** related data this time?

      _____

7. Let's get a bit more specific and look for something from a particular **HTTP** server. Run the following command:

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "http.host ==
for498.com"
```

   a. How many packets were found?

      _____

   b. What files were requested from **for498.com**?

      _____

   c. Was this easier or harder to find as compared to using **Wireshark**?

      _____

8. In **PowerShell**, enter the following command and press **Enter**.

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "http.host ==
for498.com" -T fields -e frame.time -e ip.src -e
http.request.full_uri
```

   a. Summarize what this command returned and how it could be useful in an investigation.

      _____

      _____

      _____

9. In **PowerShell**, enter the following command and press **Enter**.

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "arp"
```

   a. What did this command find and how it could be useful in an investigation?

   _____

   _____

   _____

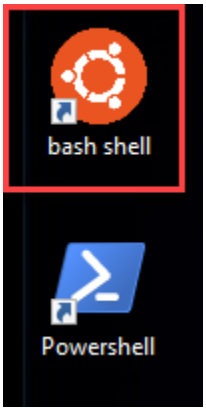
   For this filter, notice how we did not specify **`-T`** fields (and the **`-e`** parameters that follow). Without those options, **tshark** can still display information about the packets in a meaningful way, but if you really need to dissect those packets, leverage **`-T`** and **`-e`**!

OPTIONAL, OUT OF CLASS HOMEWORK: Repeat some of the filters we used in **Wireshark** but use **tshark** with and without **`-T`** and **`-e`** options.

NOTE: The next tool we will look at runs in Linux, so we will be starting a **bash** shell using the Windows Subsystem for Linux.

We saw how useful DNS history can be for several investigative purposes, but opening **Wireshark**, loading a pcap, opening a report and saving it takes time. Let's look at a command line tool that specializes in finding and extracting out DNS related information from pcap files. This tool is called **passivedns**.

10. Open a **bash** shell using the **Desktop** shortcut.



11. Run the following command to process **alltraffic.pcap**, generate report files, and display a summary of what was done.

    NOTE: switch related to **dnslog.txt** is a lower case '**ell**'

```
passivedns -r /mnt/c/Network/alltraffic.pcap -l
/mnt/c/Temp/dnslog.txt -L /mnt/c/Temp/nxdomain.txt
```

    Also note that the path is case-sensitive, so you may need to adjust '**Temp'** to '**temp**' depending on what exists on your computer or when saving the output to a different file. To make this easier, we recommend using tab completion to verify/build paths vs. typing the entire command.

    a. How many **DNS packets** were received via **IPv4/UDP**?

    _____

    b. How many **DNS records** were allocated?

    _____

12. To make sure the **DNS** entries are in order, sort the file with the following command:

```
cat /mnt/c/Temp/dnslog.txt | sort > /mnt/c/Temp/dnslogSorted.txt
```

13. Open **C:\temp\dnslogSorted.txt** in a text editor of your choice. In your **bash** shell, type the following command:

    date -u -d @<*timestamp*>

    where **<*timestamp*>** is the first field in **dnslog.txt**. For example, if the left-most value for a record is "**1216691862.993958**", the command would be:

```
date -u -d @1216691862.993958
```

    which results in a date/time of **Tue Jul 22 01:57:42 UTC 2008**

    a. Looking at **dnslogSorted.txt**, when was the first DNS query made?

    _____

b. Looking at **dnslogSorted.txt**, when was the last DNS query made?

_____

14. Search **dnslogSorted.txt** for **rushisaband.com**.

a. How many times was this host found in the log file?

_____

b. When was it first resolved?

_____

c. When was it last resolved?

_____

d. There are two entries for **rushisaband.com**. What is different (besides the timestamp) for each entry? Recall that the columns are:

Timestamp||ClientIP||ServerIP||Class||Query||Type||Answer||TTL||Count

_____

15. Search **dnslogSorted.txt** for **ciiagm.com**.

a. How many times was this host found in the log file?

_____

b. When was it first resolved?

_____

16. How many unique Client IP addresses are there in **dnslogSorted.txt**?

    a. List the unique Client IPs found:

    _____

    Hint: While **dnslog.txt** is somewhat manageable to figure this out by hand, using a command like the following can make this very easy to accomplish:

```
cut -d '|' -f 3 /mnt/c/Temp/dnslogSorted.txt | sort | uniq
```

    `-d '|'` defines the delimiter to use (cut only accepts single character separators).
    `-f 3` determines which column, based on the delimiter, to extract. In our case, the **Client IP**

    We then pipe the output of **cut** to the **sort** command which alphabetizes the **Client IP**s found. Finally, the **uniq** command gets rid of duplicates.

Exercise – Questions with Step-by-Step

NOTE: A great reference for all the possible fields is available at **http://for498.com/ovepr**

Wireshark is an excellent way to explore pcap data and get familiar with how filtering works, but in some cases, you may want to slice and dice packets in a more efficient manner. Enter **tshark**!

The good news is that once you develop some useful display filters using **Wireshark**, you can transfer that directly to **tshark**. Additionally, **tshark** allows us to extract out only certain bits of information from packets. Let's see how we can do this.

1. Go to the **PowerShell** window you previously opened during the preparation of this exercise.

```
cd C:\Program Files\Wireshark
```

2. In **PowerShell**, enter the following command, then press **Enter.**

```
.\tshark.exe -r C:\Network\trafficWithFilter.pcap -Y
"dns.flags.response == 1" -T fields -e frame.time -e dns.qry.name
-e dns.a
```

So, what is going on here?

Value	Meaning
`-Y "dns.flags.response == 1"`	Only include DNS responses (vs requests)
`-T fields`	Output format. **`fields`** allows us to only include certain properties, designated by **`-e`**
`-e frame.time`	The time of the request
`-e dns.qry.name`	The domain that was looked up
`-e dns.a`	The answer(s)

3. Repeat the above command, but redirect the output to a text file.

```
.\tshark.exe -r C:\Network\trafficWithFilter.pcap -Y
"dns.flags.response == 1" -T fields -e frame.time -e dns.qry.name
-e dns.a > C:\Temp\dnsLookups.txt
```

4. Open **C:\Temp\dnsLookups.txt** in a text editor and review the data.

5. In **PowerShell**, enter the following command, then press **Enter.**

```
.\tshark.exe -r C:\Network\trafficWithFilter.pcap -Y "http"
```

   a. How many lines of data were displayed? Why?

      Nothing was returned because of the filter we used to collect the traffic. Since we only captured traffic related to port 53, no http traffic was recorded.

6. Because we limited the packets that were recorded in **trafficWithFilter.pcap** to only traffic related to **port 53**, we do not see any **HTTP** related traffic as nothing from that display filter was found in the capture.

   Let's see what we can find in our other packet capture, by repeating the command against **alltraffic.pcap.**

   In **PowerShell**, enter the following command, then press **Enter.**

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "http"
```

   a. Did **tshark** display **HTTP** related data this time?

      Yes, and quite a bit of it. Depending on your case, you can redirect the output to a file or use a more restrictive filter (which we will do below) to reduce the amount of data returned.

7. Let's get a bit more specific and look for something from a particular **HTTP** server. Run the following command:

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "http.host ==
for498.com"
```

   a. How many packets were found?

      2 packets

   b. What files were requested from **for498.com**?

      The root folder was requested (index.htm or index.html) along with a file named *instructions.mp4*.

c. Was this easier or harder to find as compared to using **Wireshark**?

Once you know what filter you want to use, using the command line becomes slightly easier to find certain data, but if you want to explore the details of things (the contents of packets, HTTP payloads, etc.) then it is often easier to use Wireshark. Also keep in mind that the size of pcap files becomes an important factor when choosing your analysis tool. Command line tools such as tshark are typically much more efficient and effective for analyzing very large pcap files.

8. In **PowerShell**, enter the following command and press **Enter**.

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "http.host ==
for498.com" -T fields -e frame.time -e ip.src -e
http.request.full_uri
```

a. Summarize what this command returned and how it could be useful in an investigation.

This command returns the time of the request, the IP address that made the request, and the URL that was accessed.

This can be very useful because it serves as a summary of the information that can then be used to pivot into other logs or information sources, such as the hosts that made the request, or firewall logs showing which other hosts in an environment requested files from the same remote server.

BONUS: Try the "tshark -G fields" command to see a list of all available fields!

9. In **PowerShell**, enter the following command and press **Enter**.

```
.\tshark.exe -r C:\Network\alltraffic.pcap -Y "arp"
```

a. What did this command find and how it could be useful in an investigation?

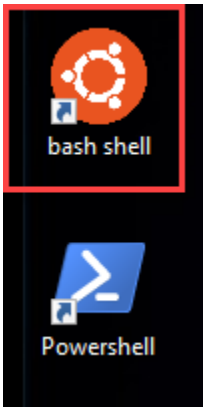
This command found all the ARP requests in the packet capture, along with the vendor of the network interface. The ARP data includes both the MAC address and the IP address of the computer. The IP address can be used to locate a host, while the vendor information can provide a clue as to the type of device to look for. In many cases, it will be difficult to account for all hosts in an environment, but if you notice requests being made where a vendor is VMWare or Synology, then it becomes easier to know you have to account for virtualization and a Network Attached Storage (NAS) server.

For this filter, notice how we did not specify **`-T`** fields (and the **`-e`** parameters that follow). Without those options **tshark** can still display information about the packets in a meaningful way, but if you really need to dissect those packets, leverage **`-T`** and **`-e`**!

NOTE: The next tool we will look at runs in Linux, so we will be starting a **bash** shell using the Windows Subsystem for Linux.

We saw how useful DNS history can be for several investigative purposes, but opening Wireshark, loading a pcap, opening a report and saving it takes time. Let's look at a command line tool that specializes in finding and extracting out DNS related information from pcap files, passivedns.

10. Open a **bash** shell using the **Desktop** shortcut.



11. Run the following command to process **alltraffic.pcap**, generate report files, and display a summary of what was done.

    NOTE: switch related to **dnslog.txt** is a lower case '**ell**'

```
passivedns -r /mnt/c/Network/alltraffic.pcap -l
/mnt/c/Temp/dnslog.txt -L /mnt/c/Temp/nxdomain.txt
```

    Also note that the path is case-sensitive, so you may need to adjust '**Temp'** to '**temp**' depending on what exists on your computer or when saving the output to a different file. To make this easier, we recommend using tab completion to verify/build paths vs. typing the entire command.

    a. How many **DNS packets** were received via **IPv4/UDP**?

       392

    b. How many **DNS records** were allocated?

       287

12. To make sure the **DNS** entries are in order, sort the file with the following command:

```
cat /mnt/c/Temp/dnslog.txt | sort > /mnt/c/Temp/dnslogSorted.txt
```

13. Open **C:\temp\dnslogSorted.txt** in a text editor of your choice. In your **bash** shell, using the following command:

    `date -u -d @<timestamp>`

    where **<*timestamp*>** is the first field in **dnslogSorted.txt**. For example, if the left-most value for a record is "**1216691862.993958**", the command would be:

```
date -u -d @1216691862.993958
```

    which results in a date/time of **Tue Jul 22 01:57:42 UTC 2008**

a. Looking at **dnslogSorted.txt**, when was the first DNS query made?

    2019-01-05 14:52:36 UTC

b. Looking at **dnslogSorted.txt**, when was the last DNS query made?

    2019-01-05 14:54:36 UTC

14. Search **dnslogSorted.txt** for **rushisaband.com**.

a. How many times was this host found in the log file?

    Two times

    Using this command displays the two records, as does a manual search in a text editor.

```
grep "rushisaband.com" /mnt/c/Temp/dnslogSorted.txt
```

b. When was it first resolved?

    2019-01-05 14:52:45 UTC

c. When was it last resolved?

    2019-01-05 14:52:45 UTC

    Note that although the timestamp for b and c are the same, the sub-second precision is different for both queries:

    1546699965.938023
    1546699965.957136

d. There are two entries for **rushisaband.com**. What is different (besides the timestamp) for each entry? Recall that the columns are:

Timestamp||ClientIP||ServerIP||Class||Query||Type||Answer||TTL||Count

Two different DNS servers did a lookup, 192.168.1.51 and 192.168.1.50

15. Search **dnslogSorted.txt** for **ciiagm.com**.

a. How many times was this host found in the log file?

Just once

b. When was it first resolved?

2019-01-05 14:52:55 UTC

16. How many unique **Client IP** addresses are there in **dnslogSorted.txt**?

a. List the unique **Client IP**s found:

1 unique IP: 192.168.1.212

Hint: While **dnslog.txt** is somewhat manageable to figure this out by hand, using a command like the following can make this very easy to accomplish:

```
cut -d '|' -f 3 /mnt/c/Temp/dnslogSorted.txt | sort | uniq -c
```

`-d '|'` defines the delimiter to use (**cut** only accepts single character separators).
`-f 3` determines which column, based on the delimiter, to extract. In our case, the **Client IP.**

We then pipe the output of **cut** to the **sort** command which alphabetizes the **Client IP**s found. Finally, the **uniq** command gets rid of duplicates.

Exercise—Key Takeaways

- tshark provides all the power of Wireshark, but in a command line, and therefore, scriptable package.
- passivedns makes it easy to quickly find all DNS information from a PCAP file.

This page intentionally left blank.

Exercise 6.1—Data and Stream Carving

Background

File carving is considered one of the core digital forensics skills. Being able to perform a *targeted* file carve against free/unallocated space in a logical file system image can be more difficult than it sounds. In fact, carving is only the first step in the process and is followed by analyzing the recovered files. This process can be quite cumbersome without additional context. Even with "successful" file carving, you may end up with many legitimate files recovered (1,000s in fact), but depending on the settings you used and what you carved for, you may end up with just as many (or more) false positives. As a result, many forensicators tend to limit file carving for key file types that they believe are most beneficial to their case.

The goal of the first part of the exercise is to expose you to file carving using PhotoRec. After we carve for files, we will analyze some of the data using a metadata extraction utility called ExifTool as well as other dedicated parsers for .lnk files (LECmd) and prefetch files (PECmd) to extract details contained in these file types.

Once we have carved data from a forensic image, we will use AXIOM to perform stream carving. Because stream carving essentially looks "inside" other files for chunks of relevant data, it can be performed on active files already in an image, or against files that have been recovered using file carving. We will use AXIOM in conjunction with files we recover by using PhotoRec to see how it can complement traditional file carving.

Objectives

- Use PhotoRec to carve files from a Windows forensic image
- Sort the recovered files into directories by extension
- Explore how ExifTool is used to extract metadata from files
- Use ExifTool to parse recovered lnk files
- Use PECmd to parse recovered prefetch files
- Use LECmd to parse recovered lnk files
- Use AXIOM to stream carve data

Exercise Preparation

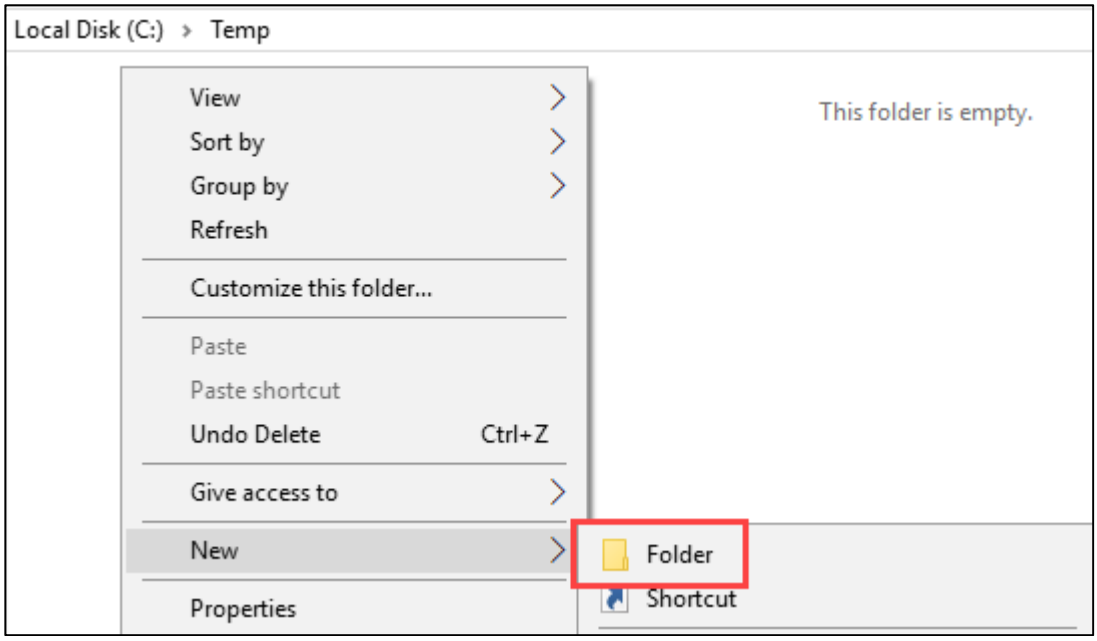
1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Verify **WindowsImage.E01** is mounted (refer to the **Mounting Evidence** exercise for mounting instructions).

NOTE: Be sure you mount **WindowsImage.E01** BEFORE you start **PhotoRec** or you will not see the mounted image as an available selection in **PhotoRec**'s media list!

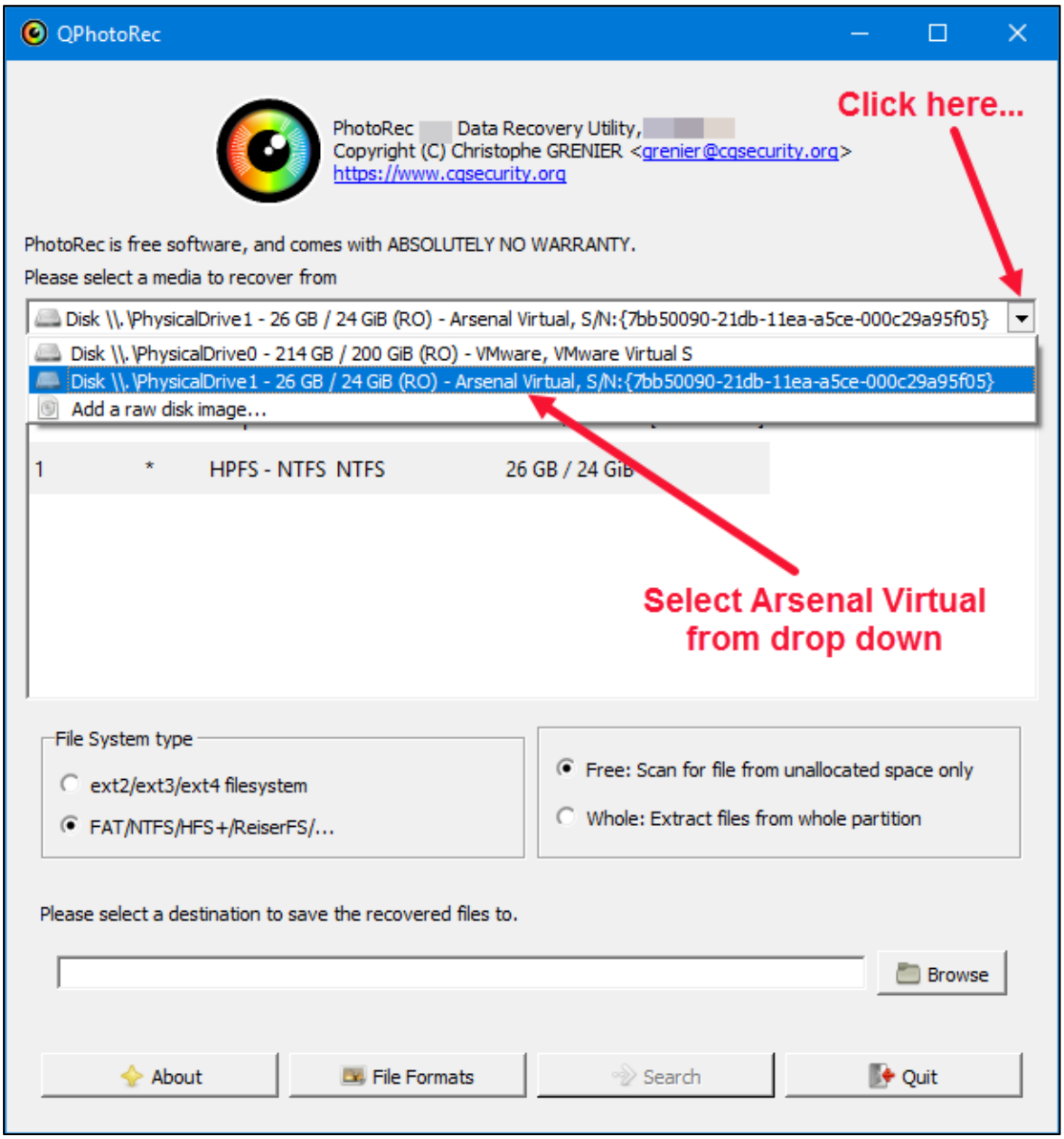
4. Open **File Explorer** and navigate to **C:\Temp**. Right click on an empty part of the folder, then select **New → Folder**. Name the folder **`Recovered`** as this is where all our recovered files will end up.



5. Using **File Explorer**, navigate to **C:\Tools**.

6. Start **PhotoRec GUI** via the shortcut inside the **Utilities** fence on your **Desktop**, or by double clicking on **qphotorec_win.exe** in **C:\Tools\PhotoRec**

7. The main interface will load.

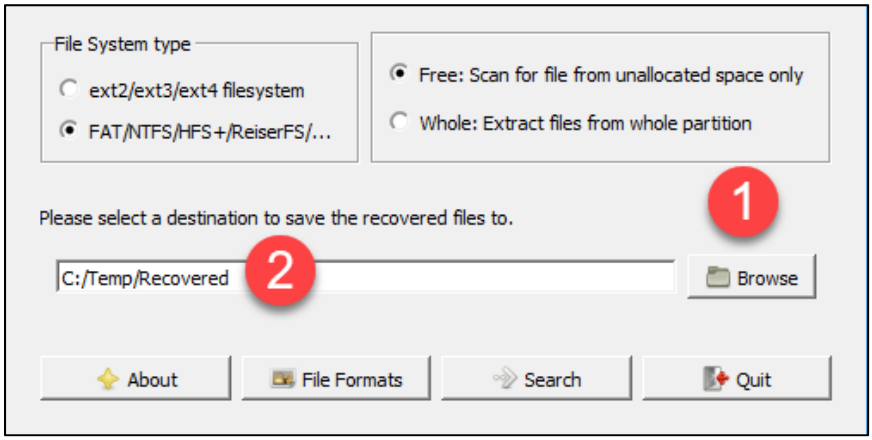


8. Select the drive that was assigned when mounting the **WindowsImage.E01** file from the drop-down. If you have more than two disks, look for the one that ends with "**Arsenal Virtual**".

   NOTE: If you already had **PhotoRec** running when you mounted the E01 image, you will NOT see the disk in the drop down. In this case, exit **PhotoRec** and restart it to refresh available media.

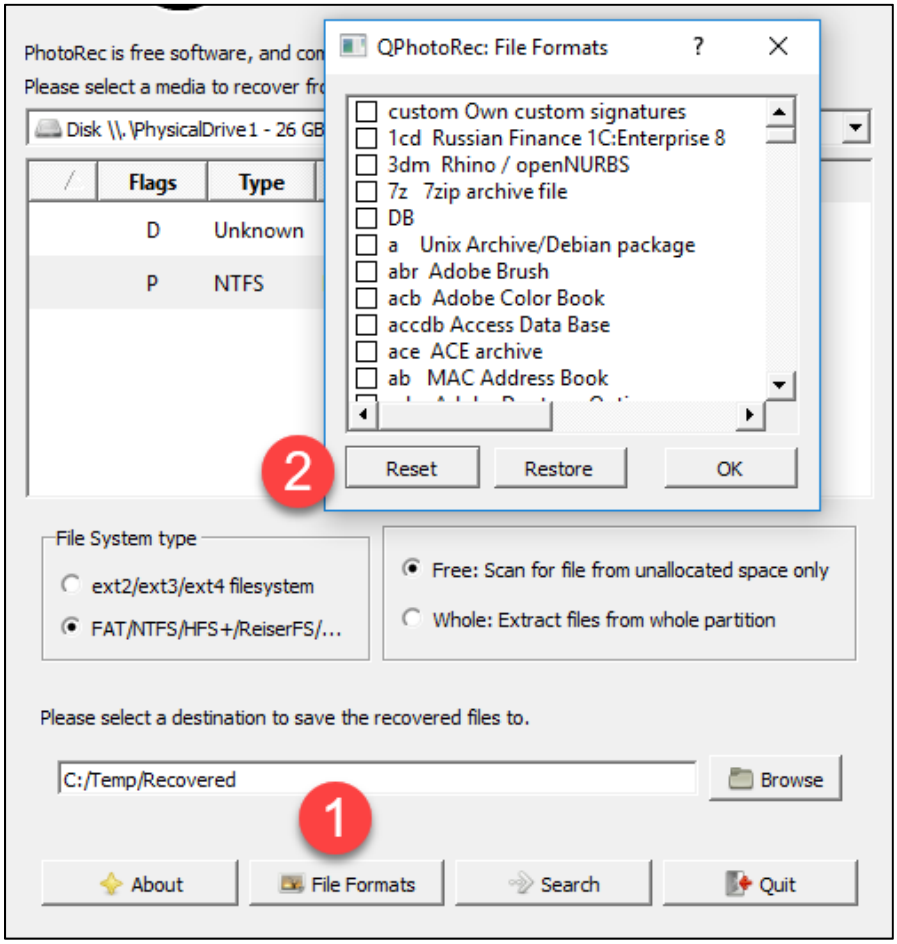
9. Verify that the **File System type** option has the NTFS radio button selected (the bottom one).

10. Click the **Browse** button, navigate to **C:\Temp\Recovered**, then click the **Select Folder** button in the folder selection dialog.



This folder is where directories containing recovered files will end up.

11. Verify the option for **Free: Scan for file from unallocated space only** is selected. While **PhotoRec** can look for files everywhere, this is not what we want, as we only to want to find deleted files.
12. The last thing to configure is the types of files we want **PhotoRec** to look for. Click the **File Formats** button, then click **Reset** to uncheck everything.



13. Recall that you rarely, if ever, want to carve for everything. We are only interested in the following file types:

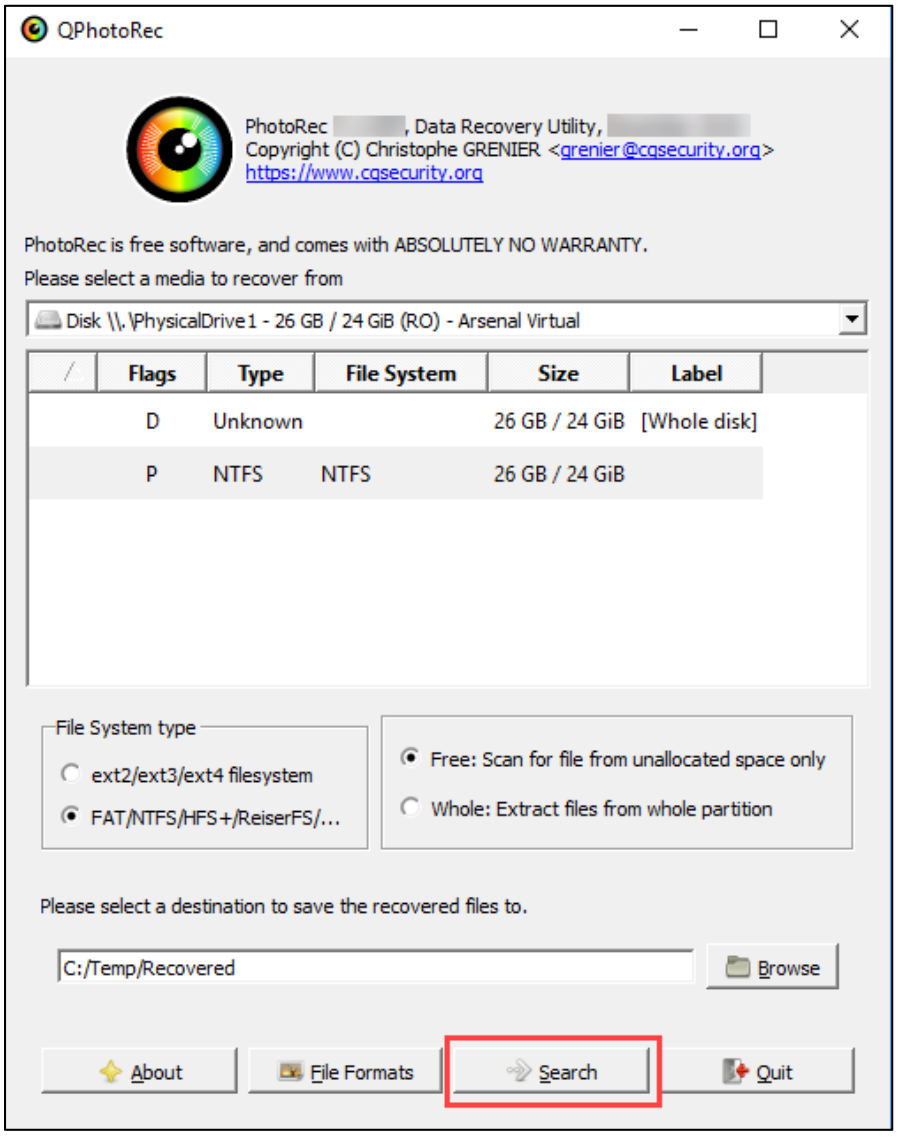
Extension	File type description
zip	Archive files
lnk	Windows shortcut files
pf	Windows Prefetch files

Find each of these in the list and check the box next to each entry.

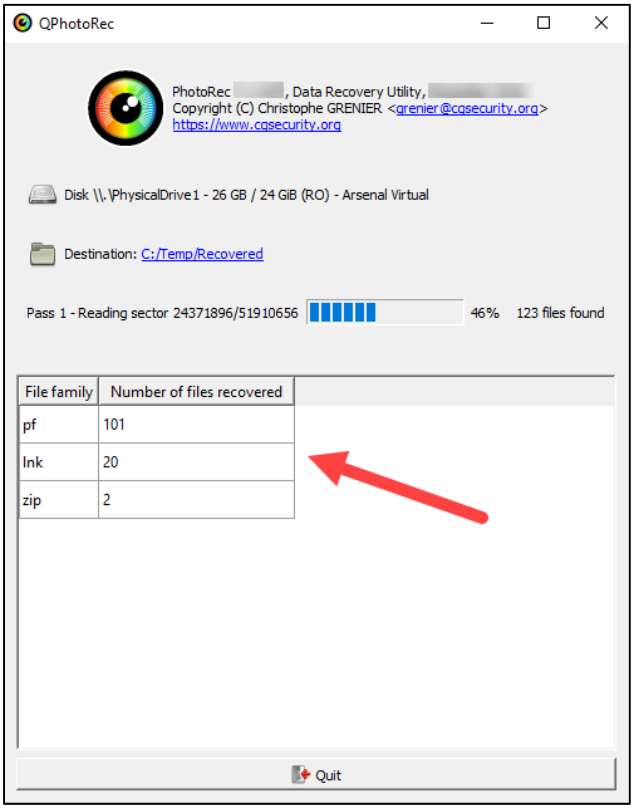
Note: You can type the extension to jump to it in the list. For example, quickly typing **`pf`** will take you right to the correct option for **Prefetch**.

14. With all three file types checked, click the **OK** button.

15. With everything configured, click the **Search** button.



16. **PhotoRec** will now begin looking for the selected file types and provide feedback as it works.



Exercise—Questions

1. When **PhotoRec** is finished processing the drive, answer the following questions:

   a. How many **pf** files were recovered?

   _____

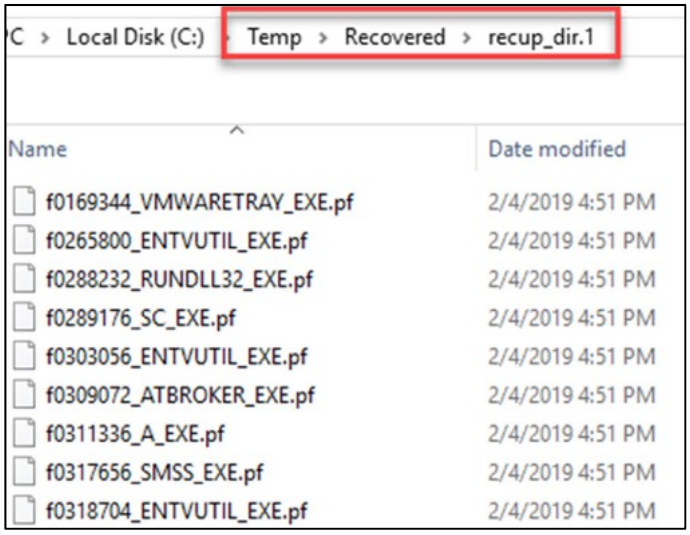
   b. How many **lnk** files were recovered?

   _____

   c. How many **zip** files were recovered?

   _____

2. Using **File Explorer**, navigate to **C:\Temp\Recovered**. Notice there is a new directory named **recup_dir.1**. Recall from earlier that **PhotoRec** will create directories like this for every 500 files it recovers. Going into this directory, we see the files recovered.



   a. How many **total** files did **PhotoRec** recover (i.e. how many files exist in this directory)?

      _____

   b. How many files have a .**docx** extension?

      _____

   c. How many files have an .**xlsx** extension?

      _____

3. With recovery complete, let's make finding these kinds of answers a bit easier by sorting the recovered files into their own directories!

4. Open a **PowerShell** window using the shortcut on the **Desktop**. You should be at the **C:\Tools>** prompt.

5. We will use the **Copy-PhotoRecFilesbyExtension.ps1 PowerShell** script to do the heavy lifting for us. This script takes two parameters:

Parameter	Meaning
RootPhotoRec	Where to find the files PhotoRec created
RootDestFolder	Where we want the sorted results to go

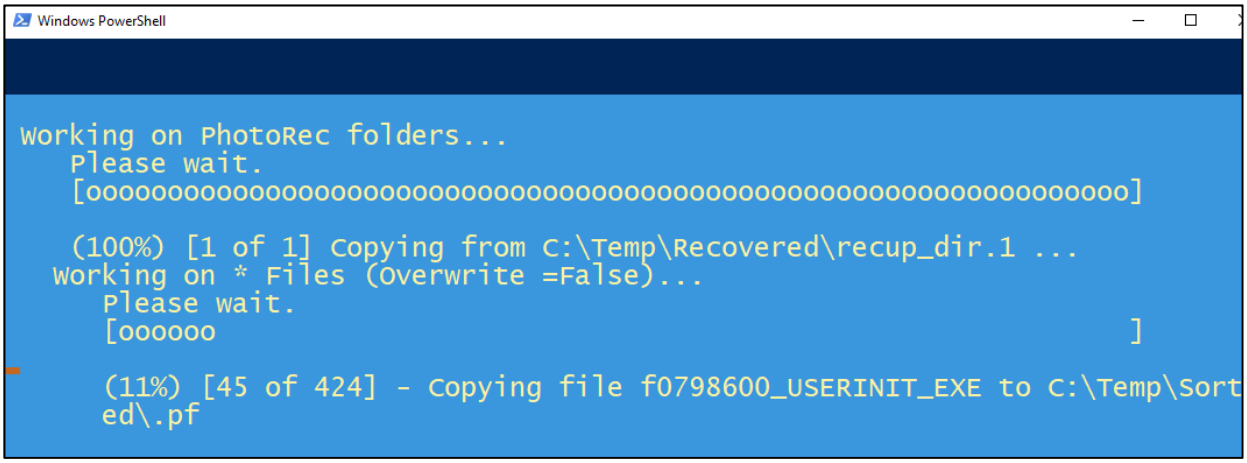
6. Execute the script by entering the following command (don't forget **<TAB>** autocomplete).

   NOTE: The command should all be on one line, with a space between **`Recovered\`** and **`-RootDestFolder`**.

```
.\Copy-PhotoRecFilesbyExtension.ps1 -RootPhotoRec C:\Temp\Recovered\
  -RootDestFolder C:\Temp\Sorted
```

   NOTE: The script will automatically create the directory specified via **`-RootDestFolder`** if it does not exist.

7. The script will then begin its work. Depending on how much data it must go through, it may take a while.



8. When the script is finished, a summary is shown that lists the unique extensions that were found.

   a. List the extensions found by **Copy-PhotoRecFilesbyExtension.ps1**

_____

_____

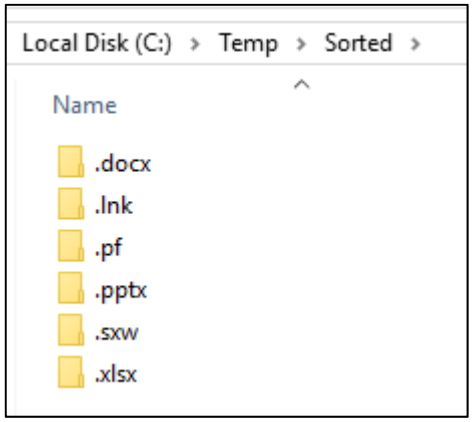
_____

_____

_____

_____

9. Using **File Explorer**, navigate to **C:\Temp\Sorted** and review the directories found.



   a. Do the directories found match the list of extensions as displayed by the **PowerShell** script from the previous step?

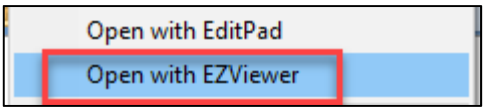
      _____

10. Take a moment to explore some of the directories for each extension to get a feel for how many of each type of file was recovered.

With our recovered files sorted by extension, you can see how much easier it is to now focus on specific file types while ignoring others. For example, if you are not interested in .pptx files at all, you can get rid of all these files by simply deleting the directory containing those files, and so on. We can leverage different tools against them to extract metadata and other forensic artifacts.

11. Open File Explorer and navigate to **C:\Temp\Sorted\.docx**

12. Open **f43892944.docx** by right-clicking on the file and choosing **Open with EZViewer** from the context menu. Read through its contents.



   a. What kind of information is present in the document?

_____

_____

_____

13. Use **exiftool** to examine the **.docx** file's metadata, by running the following command in a **PowerShell** window:

```
.\exiftool.exe C:\Temp\Sorted\.docx\f43892944.docx
```

    a. Who is the **Creator** of the document?

    _____

    b. When was the document created?

    _____

    c. According to the file's metadata, was the document ever altered after it was initially saved? Be sure to justify your answer.

    _____

    _____

14. Use **File Explorer** and navigate to **C:\Temp\Sorted\.xlsx**

15. Using **Timeline Explorer** (or **EZViewe**r via the context menu), open each **Excel** file and note the name of the file, along with the gist of what is stored in each file.

    a. File 1 name:

    _____

    b. File 1 content description:

    _____

    c. File 2 name:

    _____

    d. File 2 content description:

    _____

16. Use **File Explorer** and navigate to **C:\Temp\Sorted\.pf**

17. Locate the **Prefetch** file for **Excel** (It starts with **f439**).

    a. What is the filename of the prefetch file related to **Excel**?

    _____

    b. Where did **PhotoRec** get the name of the executable that it used when saving the files (i.e. f45404072_**SVCHOST_EXE**.pf)?

    _____

18. Open a **PowerShell** window using the shortcut on the **Desktop** and navigate to **C:\Temp\Sorted\.pf**.

    Use **`PECmd.exe`** to parse the **`f43979984_EXCEL_EXE.pf`** file by running the following command:

```
.\PECmd.exe -f C:\Temp\Sorted\.pf\f43979984_EXCEL_EXE.pf
```

19. Answer the following questions based on the output from the above command.

    a. How many times was Excel.exe **executed** (run count)?

    _____

    b. When was the last **time** Excel.exe was **last run**?

    _____

20. Review the **Files referenced** section for the following questions.

    a. Look for any references to files ending with .**XLSX**. List the file names below (just the *name* of the file itself is fine. You do not need to worry about the full path).

    _____

    _____

    _____

    _____

    _____

b. Based on the file names contained inside the prefetch file, what is a good candidate for the original filename for one of the Excel spreadsheets we recovered and reviewed earlier? (hint: compare the data in the **Excel** file with the file names referenced in the prefetch file)

_____

BONUS: Using the same technique, can you find any other file references that might lead you to programs of interest? List the name of the program and what user profile it is associated with (there are at least two).

_____

_____

21. Finally, let's look at the **lnk** files that were recovered. Open a **PowerShell** window using the shortcut on the **Desktop**. Use **LECmd.exe** to parse all .**lnk** files by running the following command:

```
.\LECmd.exe -d C:\Temp\Sorted\.lnk\ --csv C:\Temp\ -q
```

This command processes every **lnk** file found in the directory specified and writes out the results to a CSV file in the **C:\Temp** directory. The **-q** switch prevents the contents of each **lnk** file from being displayed to the screen, which speeds up processing.

a. How many total .**lnk** files were found for processing?

_____

b. How many .**lnk** files were processed successfully?

_____

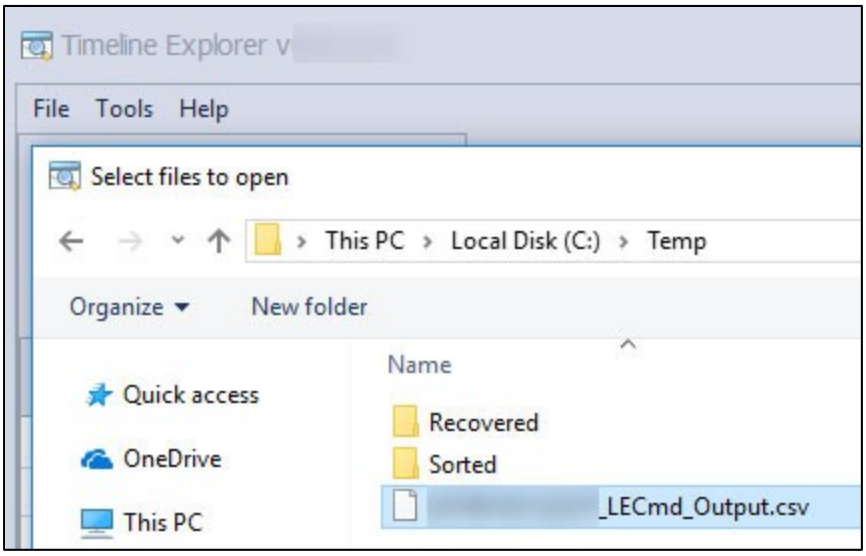
Notice that not even half of the carved lnk files were parsed successfully. Recall from our earlier discussion that file carving often produces a lot of false positives. The **lnk** files that failed to parse are examples of this. For one reason or another, the .**lnk** files that gave errors were corrupted, or otherwise incomplete.

In some cases, this is a function of the data just not being there. In others, it is a function of the tool doing the carving. This is another good time to remind you to test your tools (and use different tools!) to see how they perform in different circumstances. You may just find that one tool carves better than another for certain kinds of things.

As an example, **X-Ways Forensics** was used on **WindowsImage.E01** file to carve for .**lnk** files (in free space only) and it was able to recover 240 files. When **LECmd** was used on these 240 files, **LECmd** successfully parsed 223 out of the 240 files.

This test is a good example of not only how a different tool can potentially find *more* data for you, but also *better* data.

22. Start **Timeline Explorer** from the shortcut in the **Utilities** fence on the **Desktop**. Open the CSV file created by **LECmd** in the previous step by clicking **File → Open** and opening the CSV file located in **C:\Temp**.



23. Once the file is loaded, clear any active filters, then scroll over to the **Local Path** column and answer the following questions:

    a. Do you see any indication that any .**xlsx** files were opened according to the recovered .**lnk** files?

    _____

    b. Do you see any executable names that tie back to what we discovered when looking at the prefetch output for Excel? Which one? (hint: is there an executable name here that was also in the prefetch file list?)

    _____

    c. List any other file names of interest you may want to investigate (non-executables) based on what is in the CSV:

    _____

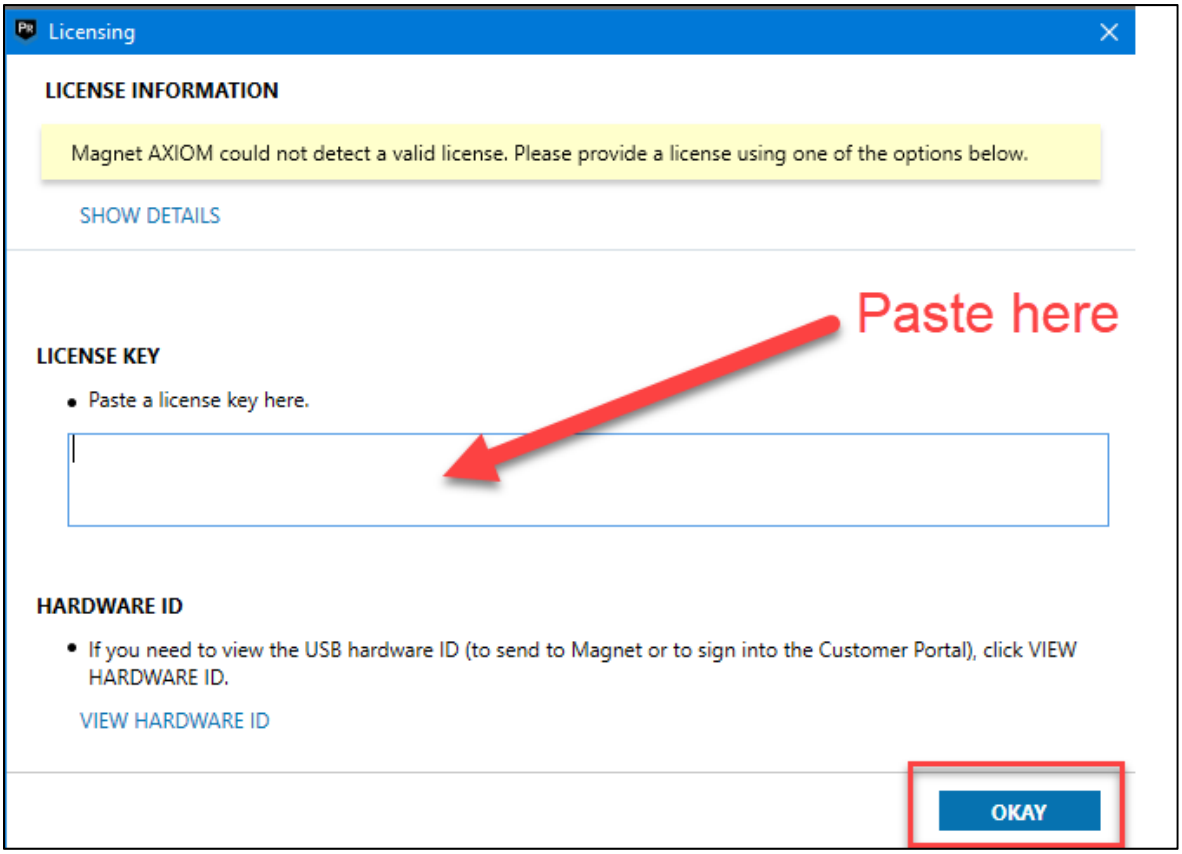
    _____

While **AXIOM** can carve for data from various sources, we are going to continue to look at the data we recovered via **PhotoRec** inside **AXIOM**, to see what else we can find out from this data.

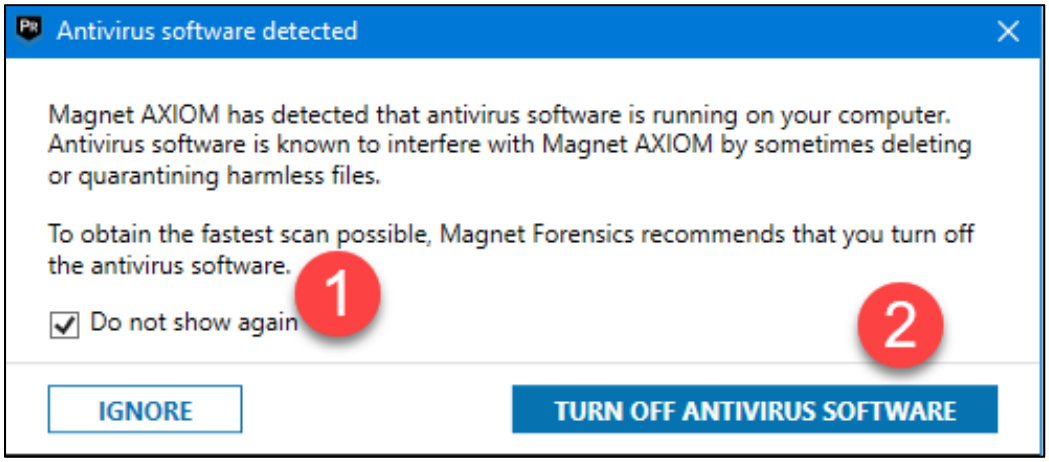
NOTE: If you are prompted to update **AXIOM**, do **NOT** update as it is a large download.

24. Start **AXIOM Process** via the shortcut in the **Forensic Suites** fence on the **Desktop**. This will let us create a new case and begin processing our data. We will use a separate program, **AXIOM Examine**, to review the data once it is processed.

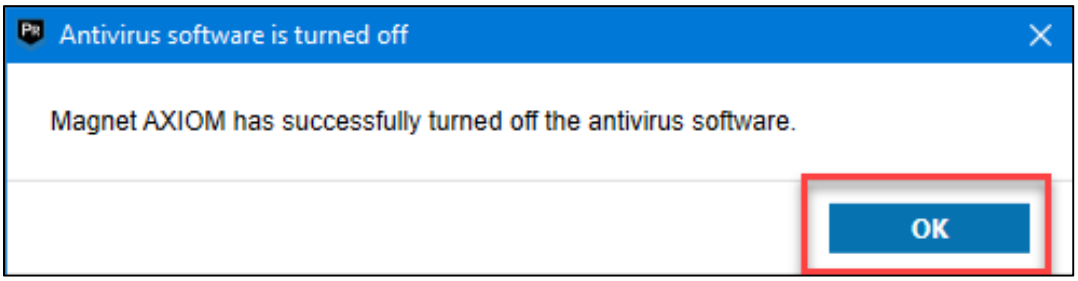
25. If **AXIOM** prompts for licensing information, paste the key provided by the instructor or your OnDemand SME into the box shown below, then click **OKAY**.



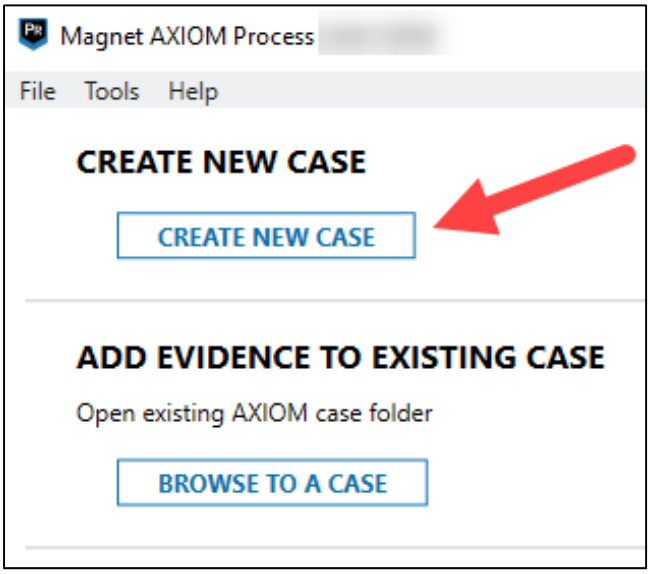
26. If prompted to disable antivirus software, check the **Do not show again** box and click the **TURN OFF ANTIVIRUS SOFTWARE** button.



27. Click **OK** in the confirmation dialog.

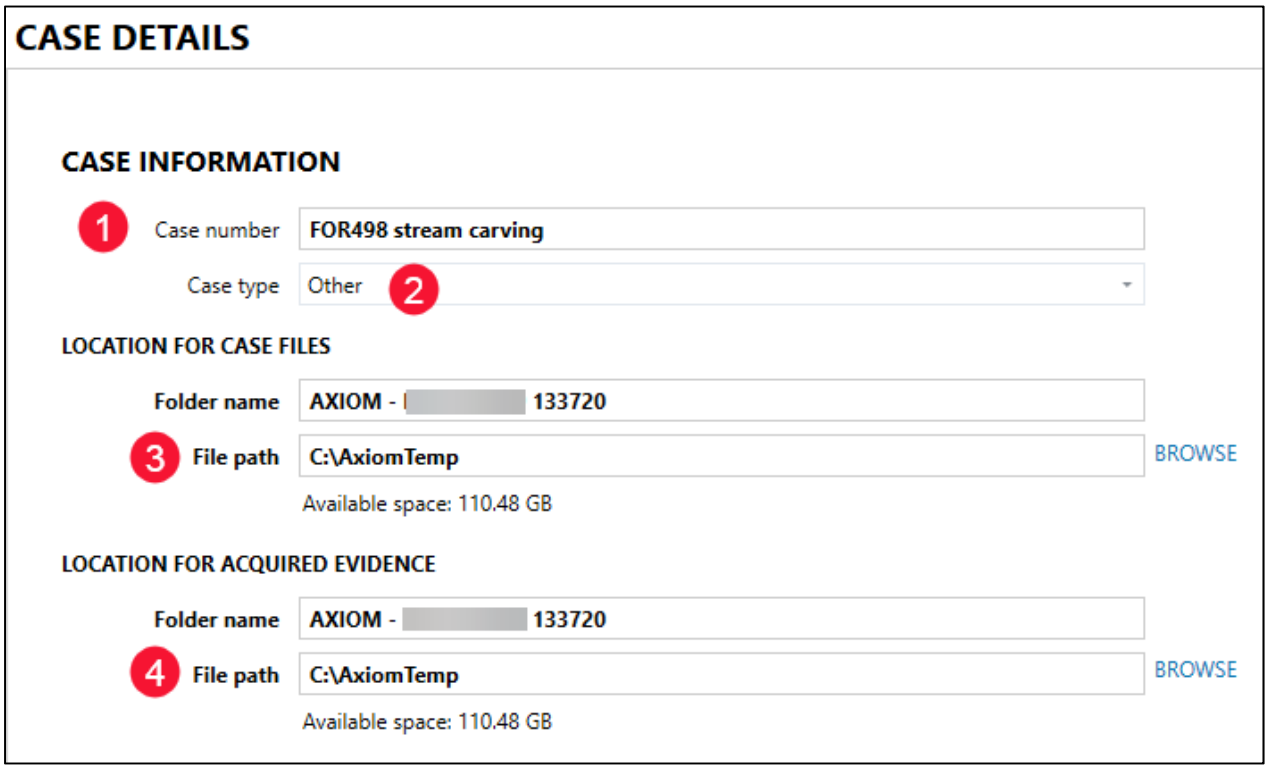


28. Once the interface is loaded, click the **CREATE NEW CASE** button



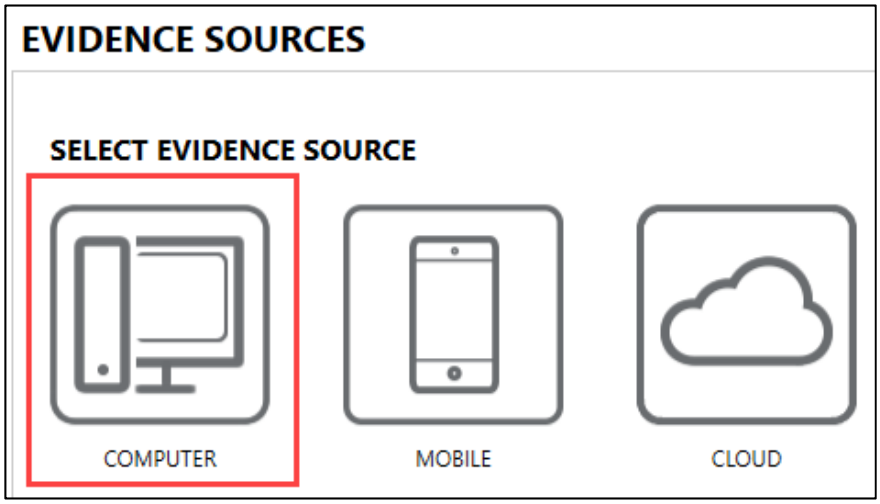
29. In the **CASE DETAILS** dialog, enter the following:

    Case number: **`FOR498 stream carving`**
    Case type: **Other**
    Case files file path: **`C:\AxiomTemp`**
    Location for acquired evidence file path: **`C:\AxiomTemp`**

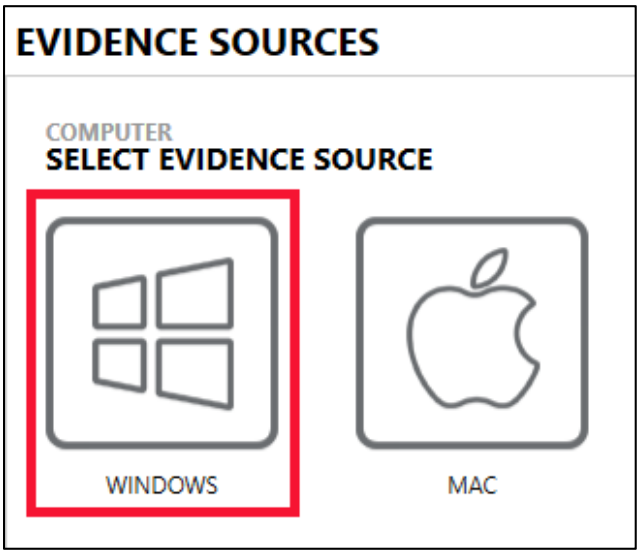


30. Click **GO TO EVIDENCE SOURCES** in the lower right.

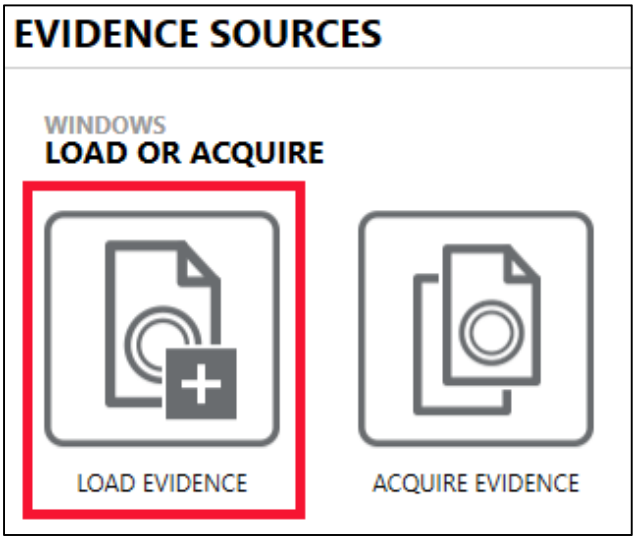
31. In the **EVIDENCE SOURCES** dialog, click **COMPUTER**…



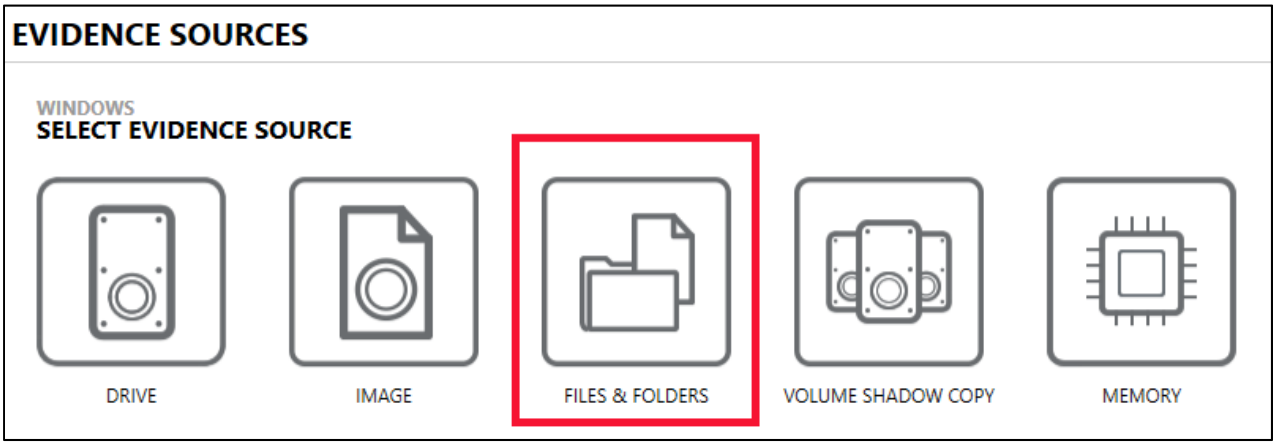
32. …then click on **WINDOWS**, and **NEXT**.



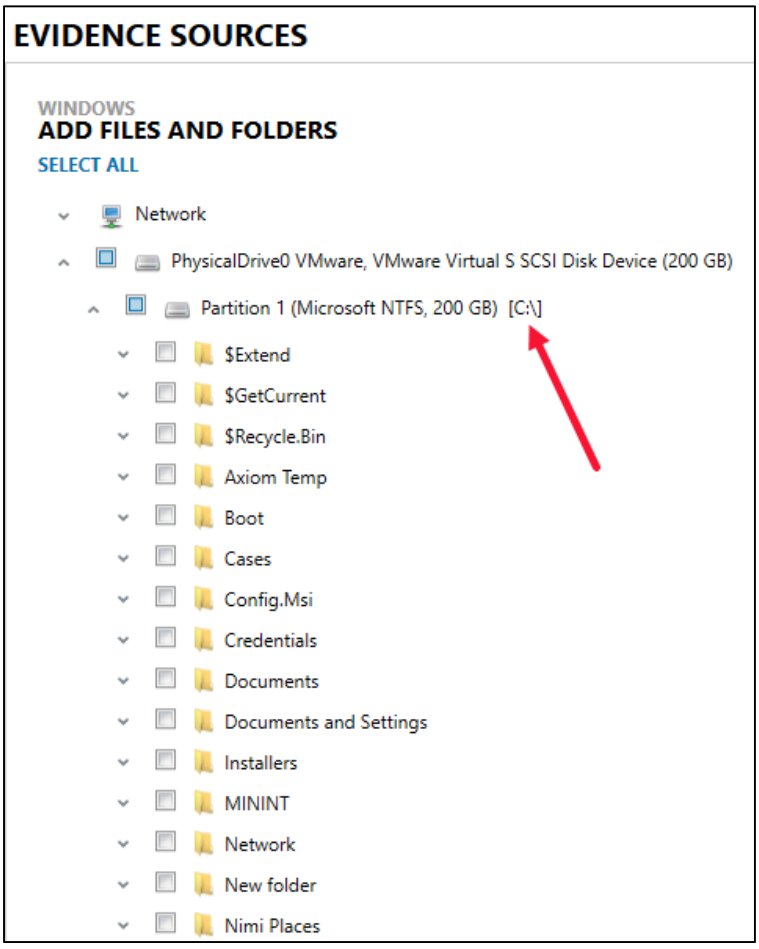
33. In the **LOAD OR ACQUIRE** dialog, click **LOAD EVIDENCE**.

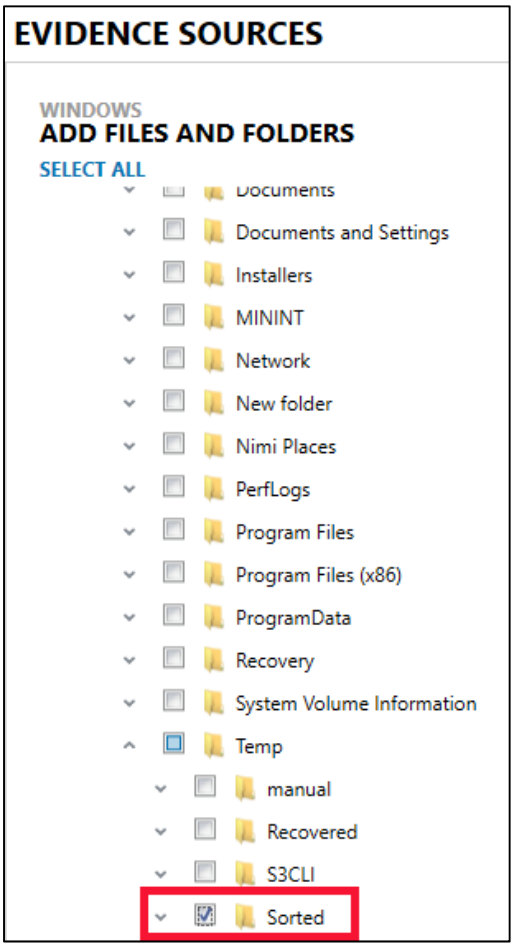


34. In the **SELECT EVIDENCE SOURCE** dialog, click **FILES & FOLDERS**.



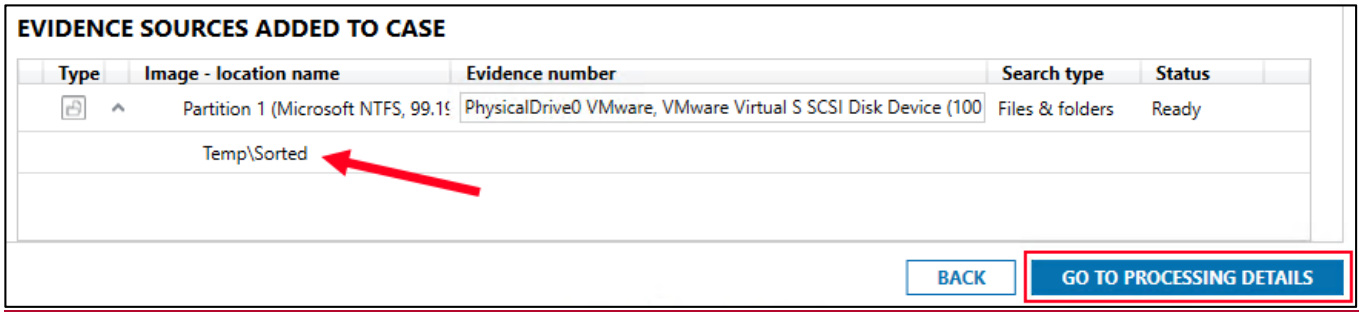
35. Expand the physical drives, looking for **C:\.** Once this is found, expand this folder and scroll down, looking for the **Temp** folder. Once found, expand **Temp** and you should see the **Sorted** directory. Check the box to the left of the **Sorted** directory.





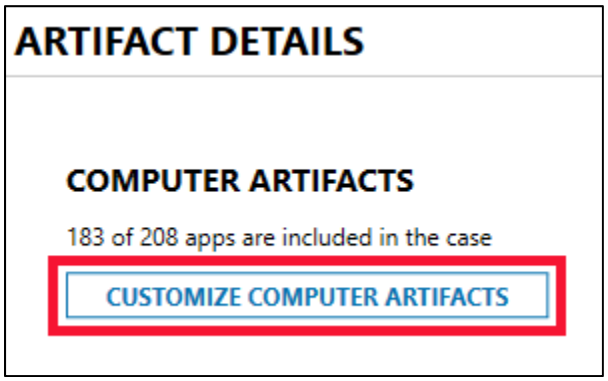
36. Click **NEXT.**

37. Review the evidence sources to confirm the **Temp** folder is listed, then click **GO TO PROCESSING DETAILS** in the lower right.

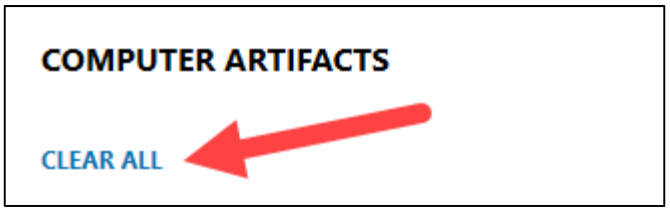


38. **PROCESSING DETAILS** allows you add things like keywords, hash values, etc. but we will skip this functionality. Click **GO TO ARTIFACT DETAILS** in the lower right.

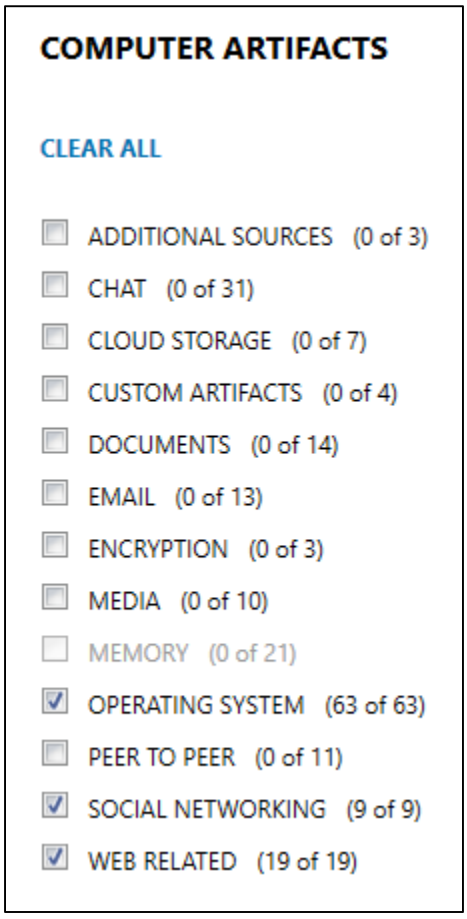
39. Click the **CUSTOMIZE COMPUTER ARTIFACTS** button.



40. Click **CLEAR ALL** at the top of the list.

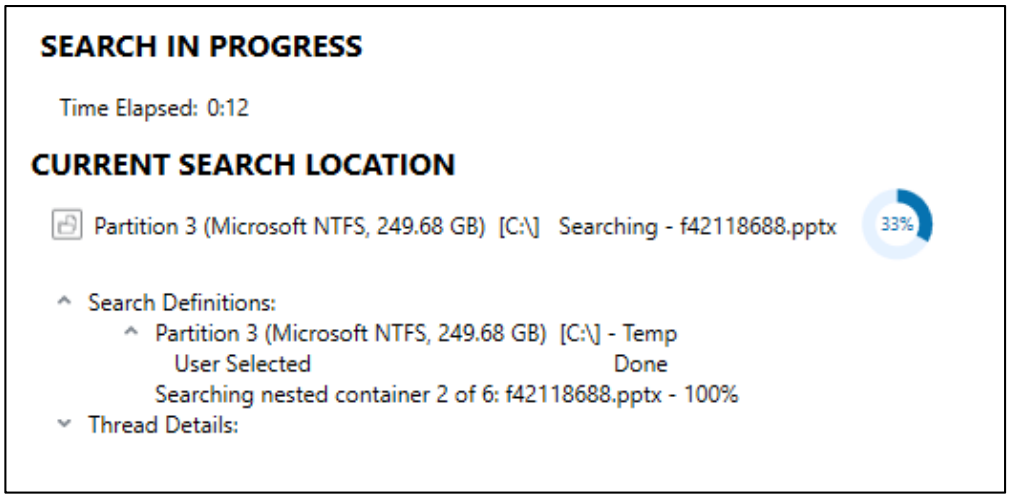


41. Check **OPERATING SYSTEM**, **SOCIAL NETWORKING**, and **WEB RELATED**.



42. Click **GO TO ANALYZE EVIDENCE** in the lower right.

43. Click **ANALYZE EVIDENCE** in the lower right to begin analyzing the data. Once processing has started, progress is displayed.

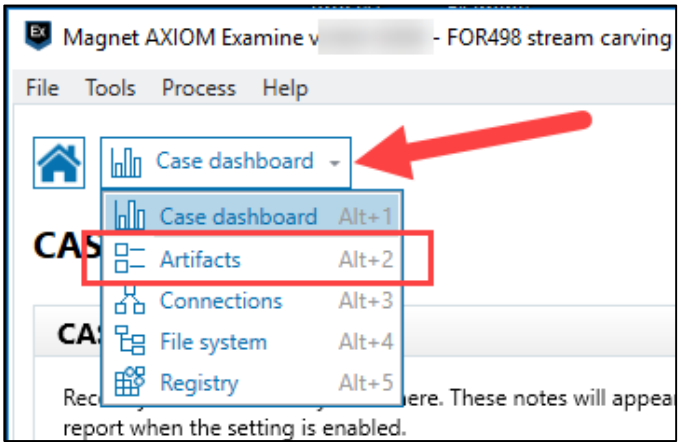


Once processing has started, **AXIOM Examine** will be automatically started with case data loaded.

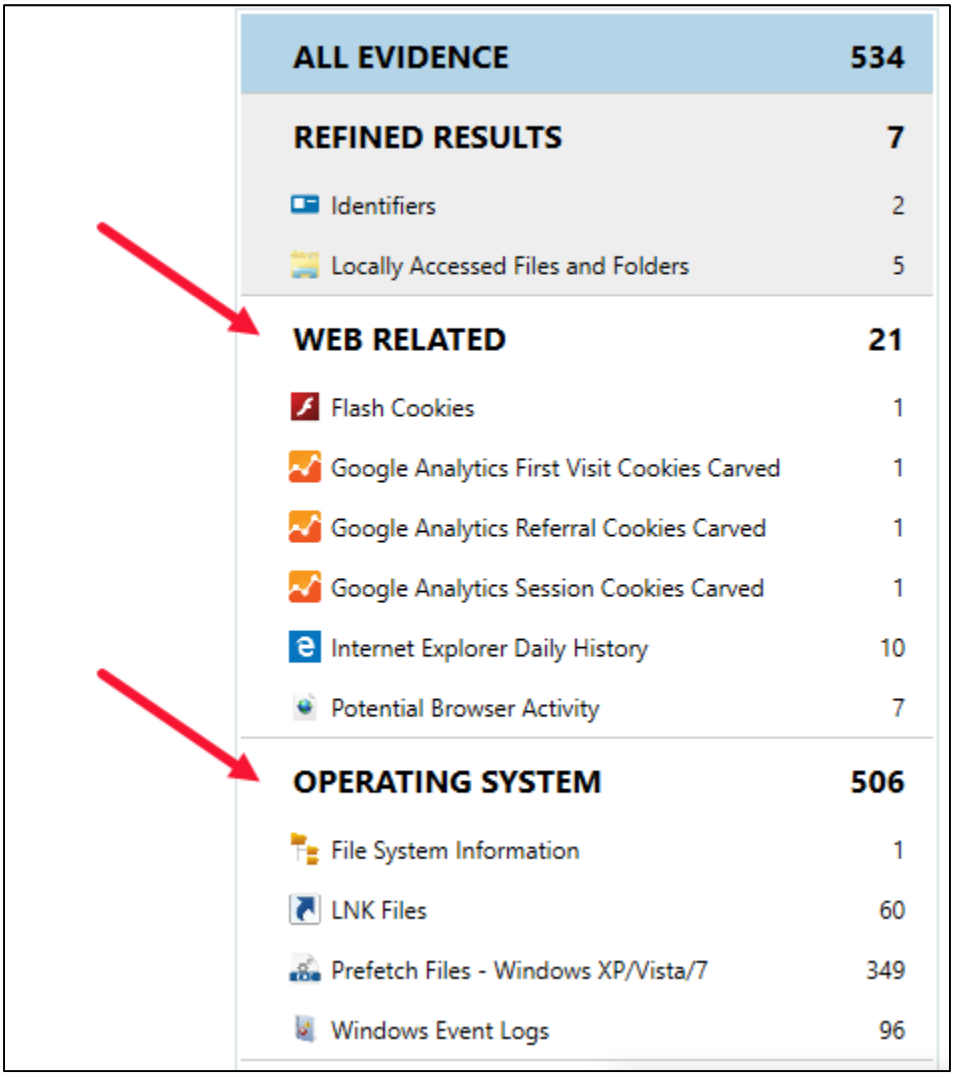
44. Switch to the **AXIOM Examine** window. If a **Quick tips** window pops up, close it with the **X** in the upper right corner of the **Quick tips** dialog.

45. The results will be displayed as they are available in the **Examine** interface. When processing is finished (it shouldn't take too long), click the **OKAY** prompt in the lower left (if necessary) to reload the case, so all data is available.

46. The main **Examine** interface is displayed. In the upper left, click the **Case dashboard** drop down and select **Artifacts** from the list (or use the hotkey **ALT+2**)

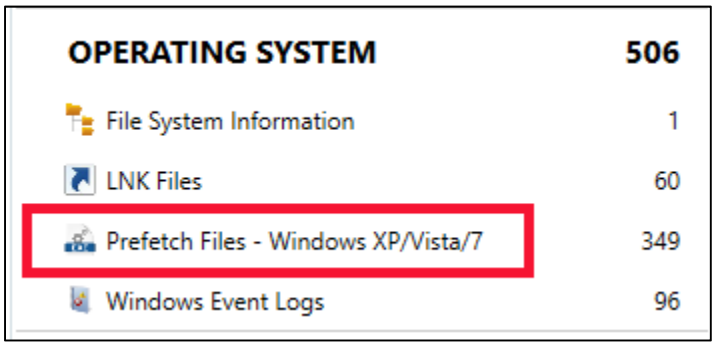


47. Under the **ALL EVIDENCE** section on the left, expand the **WEB RELATED** and **OPERATING SYSTEM** sections by clicking the down arrow to the left of the category name.



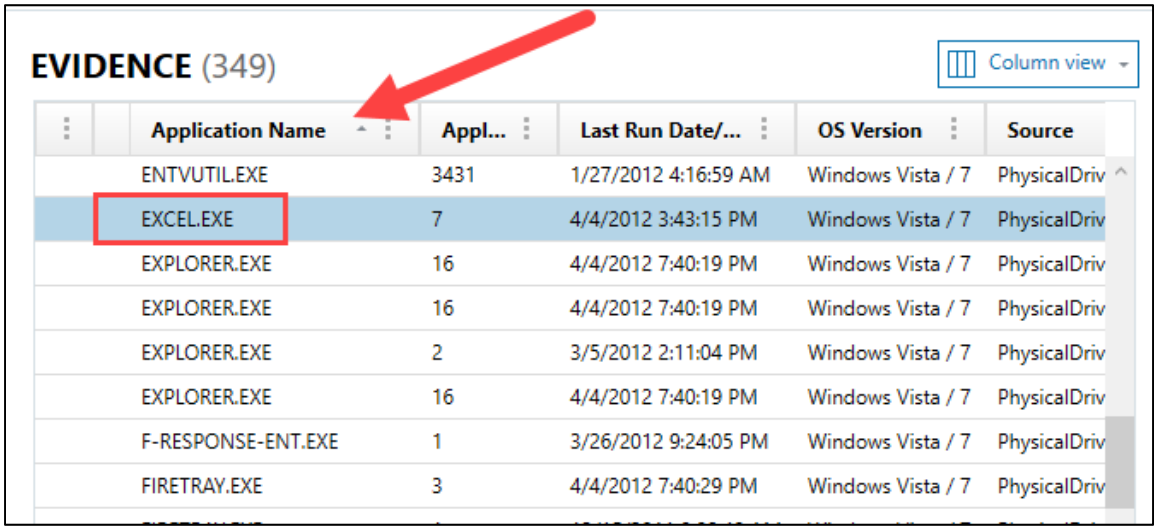
This view of the data provides a summary of the kinds of data found, as well as the number of items found for each item.

48. Click on **Prefetch Files** in the **OPERATING SYSTEM** group.

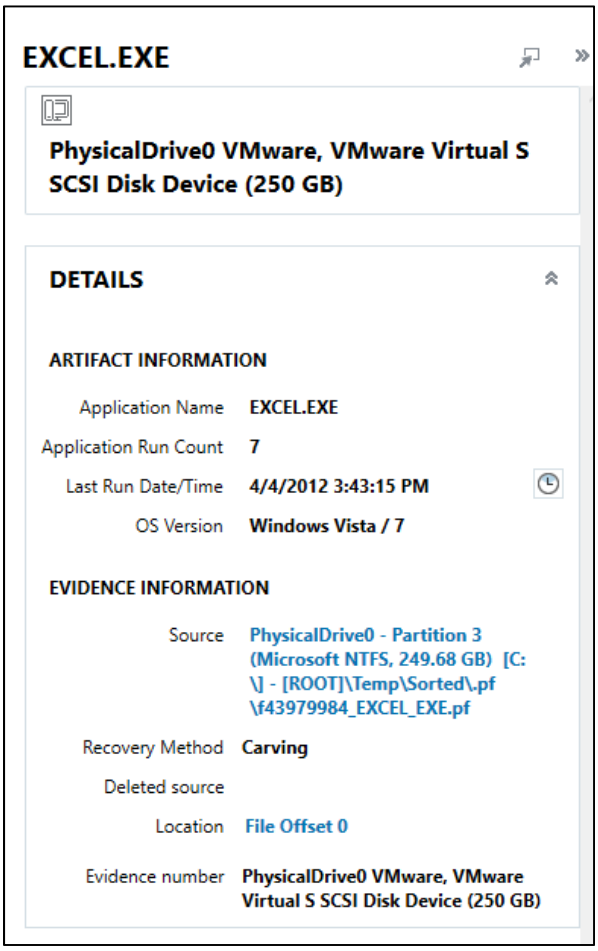


This loads the details in the grid to the left of the summary.

49. Click the **Application Name** column header to sort by that column, then scroll until you find the **EXCEL.EXE** entry.



50. Clicking on **EXCEL.EXE** loads the details for this entry to the right of the grid.



a. How many times was **EXCEL.EXE** run?

_____

b. When was the last time **EXCEL.EXE** was run?

_____

c. Do these values match what you found when using **PECmd** to look at this prefetch file?

_____

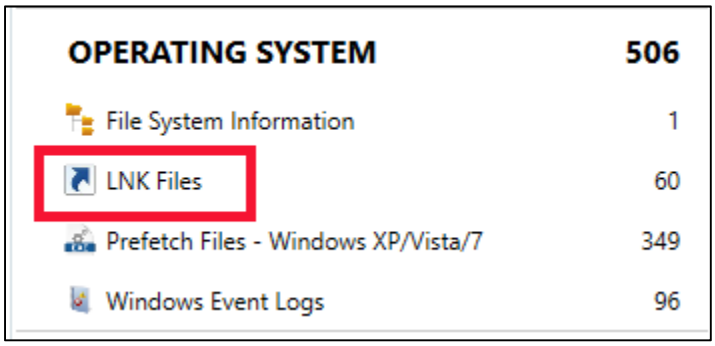
d. What is missing from this view of the prefetch file that **PECmd** showed?

_____

_____

_____

51. Under the **OPERATING SYSTEM** artifact list, click **LNK Files**.



a. How many **lnk** files did **AXIOM** parse?

_____

52. Click the **Linked Path** column header to sort by that column. Review the entries under **Linked Path**. You may need to scroll the list so the entries with data are shown.

a. Does **AXIOM** list any additional files (non-executable) that **LECmd** did not list? If so, what file names?

_____

Recall from earlier that **LECmd** was only able to parse 17 **lnk** files, but here we see 60 were parsed. If you look through the list of **lnk** files, you will see that a lot of files do not have anything listed for **Linked Path** (and many other columns as well).

This is an example of stream carving, in that **AXIOM** parsed out what it could from the available **lnk** files, whereas **LECmd**, when it encountered bad data, stopped processing the **lnk** file.

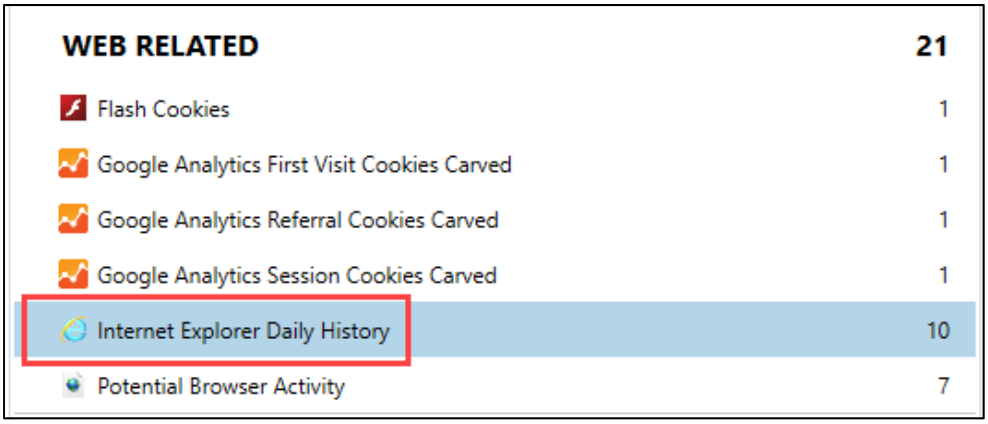
53. Spend a minute or two looking at the **lnk** files where **Linked Path** is blank.

    a. Do any of these contain useful information in the other columns? If so, what?

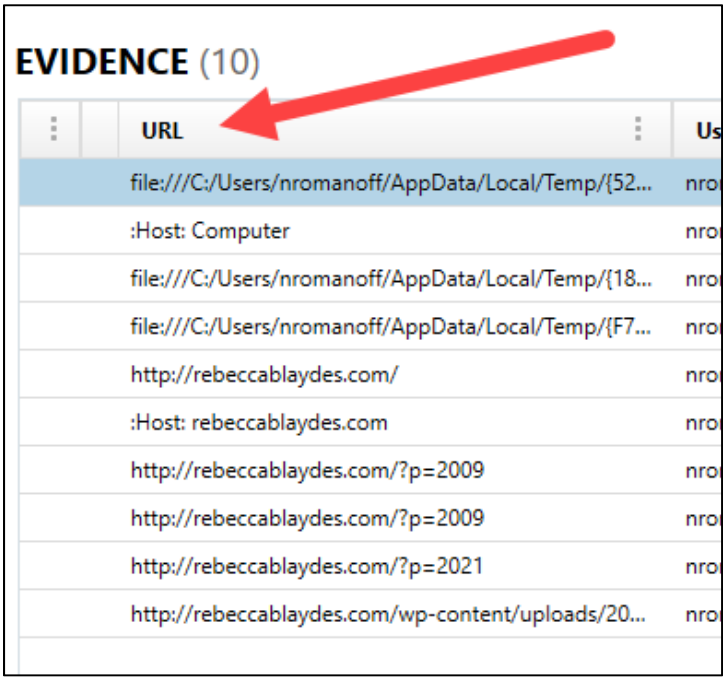
    _____

    This is illustrative of the balancing act that tools take between parsing and displaying what it can with damaged **lnk** files (**AXIOM**) even though the most relevant data may not be found, vs. erroring out when incomplete or unexpected data is encountered (**LECmd**).

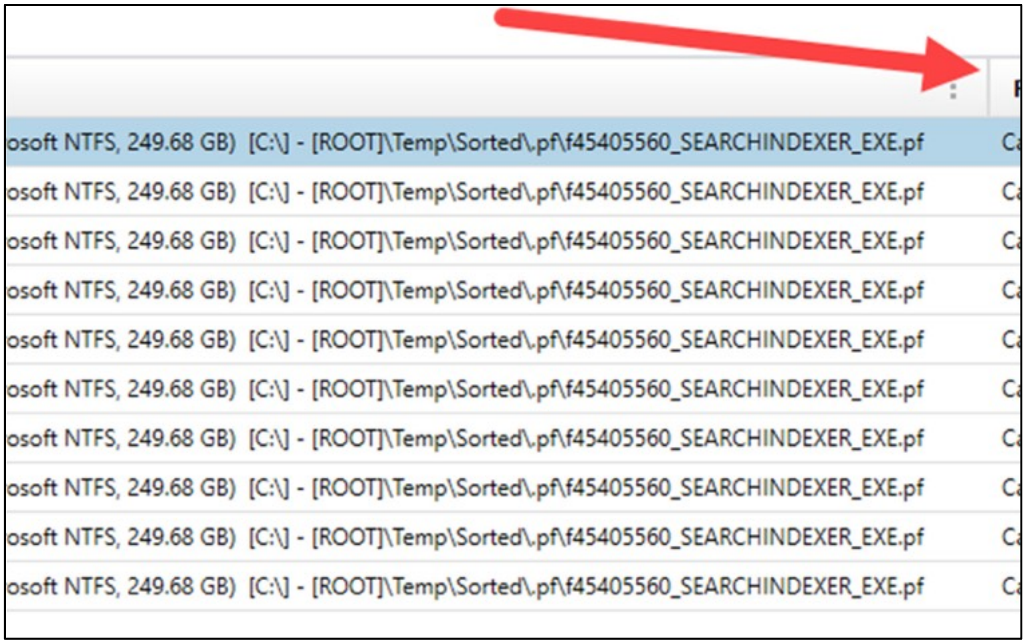
54. Under the **WEB RELATED** artifact list, click **Internet Explorer Daily History**



55. Take a moment to review the URLs under the **URL** column. Recall that we did not file carve for any browser history related artifacts in **PhotoRec**, yet **AXIOM** has found some!



56. Locate the **Source** column and expand it so the entire source is visible. To do this, hover over the column separator, then left click and drag to increase the column's width. You are in the right spot when the cursor changes to a horizontal, double-sided arrow.



a. What is the full path and file name of the file where the URLs were all found?

____

57. Open a **PowerShell** window using the shortcut on the **Desktop**.

58. Use **PECmd** to parse **C:\Temp\Sorted\.pf\f45405560_SEARCHINDEXER_EXE.pf** with the following command:

```
.\PECmd.exe -f C:\Temp\Sorted\.pf\f45405560_SEARCHINDEXER_EXE.pf
```

a. Did the command complete successfully?

____

b. Was **PECmd** able to extract any data from the prefetch file, such as run count and last run timestamp?

____

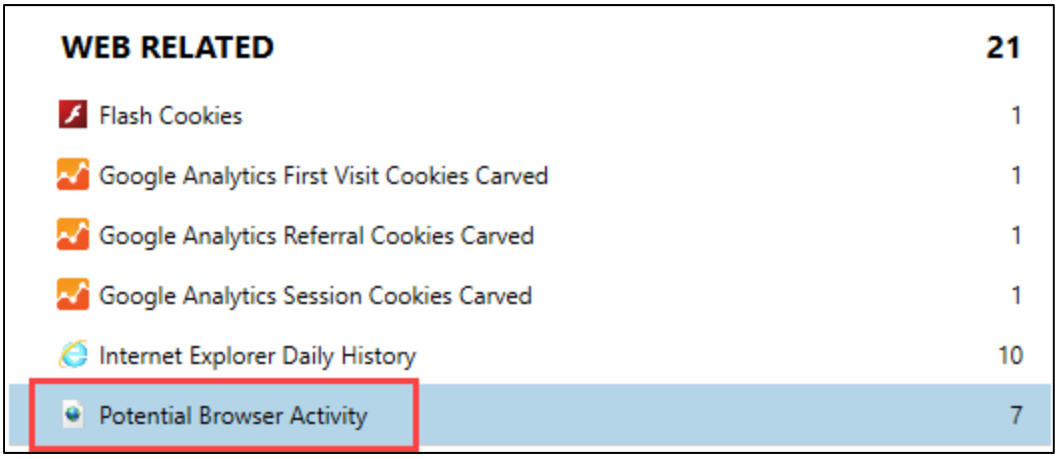
You would not normally expect to find URLs in a prefetch file, but **AXIOM** found some. How did it do this? Recall that **AXIOM** is looking for "data within data". If you were to open the .**pf** file where the URLs were found in a hex editor, you could visually see the URLs, like this:

00009216	55 52 4C 20 02 00 00 00 70 5F E8 1A 9F FB CC 01	URL p_è ŸûÌ
00009232	5D 0F BF 03 C9 FB CC 01 81 40 6D 95 00 00 00 00	] ¿ ÉûÌ @m•
00009248	00 00 00 00 00 00 00 00 00 00 00 00 80 51 01 00	€Q
00009264	60 00 00 00 68 00 00 00 FE 00 10 10 00 00 00 00	` h þ
00009280	04 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00	
00009296	66 40 6D 95 01 00 00 00 00 00 00 00 00 00 00 00	f@m•
00009312	00 00 00 00 EF BE AD DE 3A 32 30 31 32 30 33 30	ï¾-Þ:2012030
00009328	36 32 30 31 32 30 33 30 37 3A 20 6E 72 6F 6D 61	620120307: nroma
00009344	6E 6F 66 66 40 68 74 74 70 3A 2F 2F 72 65 62 65	noff@http://rebe
00009360	63 63 61 62 6C 61 79 64 65 73 2E 63 6F 6D 2F 00	ccablaydes.com/
00009376	EF BE AD DE EF BE AD DE EF BE AD DE EF BE AD DE	ï¾-Þï¾-Þï¾-Þï¾-Þ
00009392	EF BE AD DE EF BE AD DE EF BE AD DE EF BE AD DE	ï¾-Þï¾-Þï¾-Þï¾-Þ
00009408	EF BE AD DE EF BE AD DE EF BE AD DE EF BE AD DE	ï¾-Þï¾-Þï¾-Þï¾-Þ
00009424	EF BE AD DE EF BE AD DE EF BE AD DE EF BE AD DE	ï¾-Þï¾-Þï¾-Þï¾-Þ
00009440	EF BE AD DE EF BE AD DE EF BE AD DE EF BE AD DE	ï¾-Þï¾-Þï¾-Þï¾-Þ
00009456	EF BE AD DE EF BE AD DE EF BE AD DE EF BE AD DE	ï¾-Þï¾-Þï¾-Þï¾-Þ

But how did that data even get in this prefetch file in the first place? Well, as we said before, carving is a best effort endeavor. When **PhotoRec** was looking for prefetch files, it found a valid header, then did its best to recover the prefetch file contents. In this case, however, we have data from a non-prefetch file as well. Because **AXIOM** is looking for data within data, it found and displayed the URL.

This is another great example of stream carving (i.e. looking for data within data) vs. file carving. In many cases, stream carving can recover useful information inside files that may not parse with a native parser (like **PECmd** in this case).

59. Under the **WEB RELATED** artifact list, click **Potential Browser Activity**



60. Locate the **Source** column and expand it so the entire source is visible. To do this, hover over the column separator, then left click and drag to increase the column's width.

    a. How many unique files were these URLs found in (hint: sorting on the **Source** column will make this easier)

    _____

61. In a **PowerShell** window, confirm you are in **C:\Tools**. Use **PECmd** to parse **C:\Temp\Sorted\.pf\f43410352_WMIPRVSE_EXE.pf** with the following command:

```
.\PECmd.exe -f C:\Temp\Sorted\.pf\f43410352_WMIPRVSE_EXE.pf
```

    a. Did the command complete successfully?

    _____

    b. Was **PECmd** able to extract any data from the prefetch file?

    _____

    c. How did parsing this file compare with the data exacted from **f45405560_SEARCHINDEXER_EXE.pf**?

    _____

    _____

    _____

OPTIONAL HOMEWORK: Run **AXIOM** again, but against the entire **WindowsImage.E01**. It is recommended you do this outside of class and let it run overnight as it can take a while to run. Compare and contrast the data we looked at above with all the data that **AXIOM** finds from the entire image.

Exercise Questions—Step-by-Step

1. When **PhotoRec** is finished processing the drive, answer the following questions:

   a. How many **pf** files were recovered?

   349

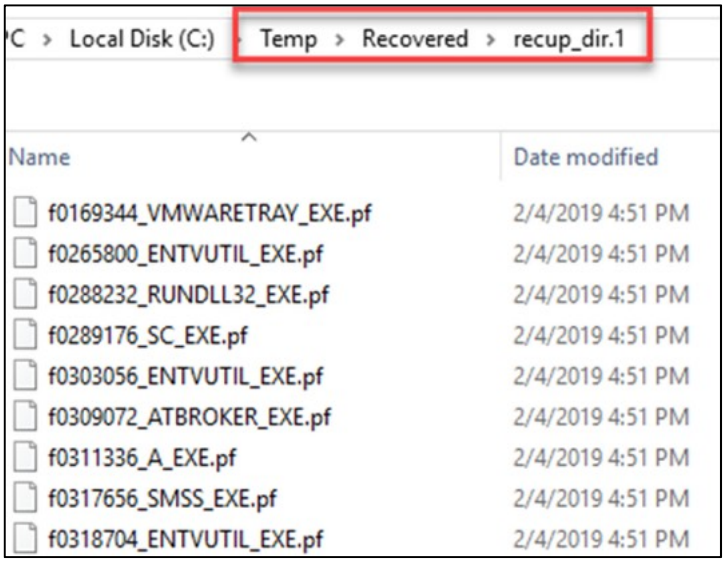
   b. How many **lnk** files were recovered?

   68

   c. How many **zip** files were recovered?

   7

2. Using **File Explorer**, navigate to **C:\Temp\Recovered**. Notice there is a new directory named **recup_dir.1**. Recall from earlier that **PhotoRec** will create directories like this for every 500 files it recovers. Going into this directory, we see the files recovered.



   a. How many **total** files did **PhotoRec** recover (i.e. how many files exist in this directory)?

   424

   b. How many files have a .**docx** extension?

   One

   c. How many files have an .**xlsx** extension?

   Two

3. With recovery complete, let's make finding these kinds of answers a bit easier by sorting the recovered files into their own directories!

4. With our files recovered, let's sort all the recovered files into directories named after the extension of each recovered file.

5. Open a **PowerShell** window using the shortcut on the **Desktop**.

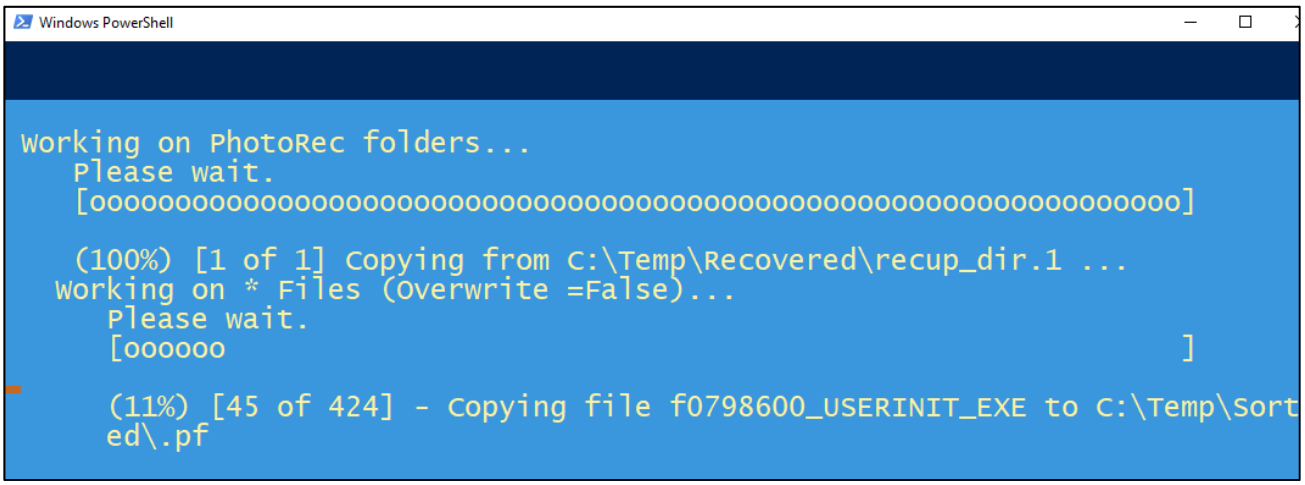
6. Execute the script by entering the following command (don't forget **<TAB>** autocomplete).

   NOTE: The command should all be on one line, with a space between **`Recovered\`** and **`-RootDestFolder`**.

```
.\Copy-PhotoRecFilesbyExtension.ps1 -RootPhotoRec C:\Temp\Recovered\
  -RootDestFolder C:\Temp\Sorted
```

   NOTE: The script will automatically create the directory specified via **`-RootDestFolder`** if it does not exist.

7. The script will then begin its work. Depending on how much data it must go through, it may take a while.

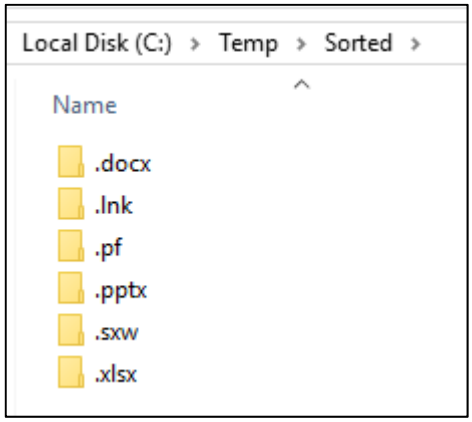


8. When the script is finished, a summary is shown that lists the unique extensions that were found.

   a. List the extensions found by **Copy-PhotoRecFilesbyExtension.ps1**

      .pf
      .lnk
      .xlsx
      .pptx
      .docx
      .sxw

9. Using **File Explorer**, navigate to **C:\Temp\Sorted** and review the directories found.



   a. Do the directories found match the list of extensions as displayed by the **PowerShell** script from the previous step?

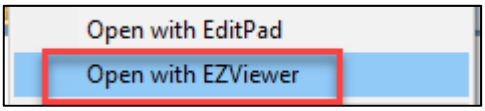
      Yes

10. Take a moment to explore some of the directories for each extension to get a feel for how many of each type of file was recovered.

With our recovered files sorted by extension, you can see how much easier it is to now focus on specific file types while ignoring others. For example, if you are not interested in .pptx files at all, you can get rid of all these files by simply deleting the directory containing those files, and so on. We can leverage different tools against them to extract metadata and other forensic artifacts.

11. Open **File Explorer** and navigate to **C:\Temp\Sorted\.docx**

12. Open **f43892944.docx** by right-clicking on the file and choosing **Open with EZViewer** from the context menu. Read through its contents.



   a. What kind of information is present in the document?

      Information about Iron Man's armor

13. Use **exiftool** to examine the **.docx** file's metadata, by running the following command in a **PowerShell** window:

```
.\exiftool.exe C:\Temp\Sorted\.docx\f43892944.docx
```

    a. Who is the **Creator** of the document?

       Natasha Romanoff

    b. When was the document created?

       2011:09:17 16:43:00Z

    c. According to the file's metadata, was the document ever altered after it was initially saved? Be sure to justify your answer.

       No, it was not. The Modify Date is 2011:09:17 16:43:00Z, which is the same as the Create Date

14. Use **File Explorer** and navigate to **C:\Temp\Sorted\.xlsx**

15. Using **Timeline Explorer** (or **EZViewer** via the context menu), open each **Excel** file and note the name of the file, along with the gist of what is stored in each file.

    a. File 1 name:

       f5537552.xlsx

    b. File 1 content description:

       Credit card numbers

    c. File 2 name:

       f20491760.xlsx

    d. File 2 content description:

       A list of names, addresses, email addresses, and so on.

16. Use **File Explorer** and navigate to **C:\Temp\Sorted\.pf**

17. Locate the **Prefetch** file for **Excel** (it starts with **f439**).

    a. What is the filename of the prefetch file related to **Excel**?

    f43979984_EXCEL_EXE.pf

    b. Where did **PhotoRec** get the name of the executable that it used when saving the files (i.e. f45404072_**SVCHOST_EXE**.pf)?

    The name of the executable is embedded in the .pf file. Once PhotoRec recovered the file, it used this data from inside the file to give the recovered file a more meaningful name.

18. Open a **PowerShell** window using the shortcut on the **Desktop** and navigate to **C:\Temp\Sorted\.pf**.

    Use **`PECmd.exe`** to parse the **`f43979984_EXCEL_EXE.pf`** file by running the following command:

```
.\PECmd.exe -f C:\Temp\Sorted\.pf\f43979984_EXCEL_EXE.pf
```

19. Answer the following questions based on the output from the above command:

    a. How many times was **Excel.exe executed** (run count)?

    7

    b. When was the last **time Excel.exe** was **last run**?

    2012-04-04 15:43:15

20. Review the **Files referenced** section for the following questions.

    a. Look for any references to files ending with .**XLSX**. List the file names below (just the *name* of the file itself is fine. You do not need to worry about the full path)

    - **62E94EAC.XLSX**
    - **UNDERCOVER-AGENTS-LIST-FOR-UNITED-STATES.XLSX**
    - **~UNDERCOVER-AGENTS-LIST-FOR-UNITED-STATES.XLSX**
    - **CC-BACKSTOPPEDACCOUNTS.XLSX**
    - **~$CC-BACKSTOPPED-ACCOUNTS.XLSX**
    - **~$CREDIT-CARD-NUMBERS-FOR-RESEARCH.XLSX**

Protip: You can leverage **PowerShell** to make this a bit easier:

```
.\PECmd.exe -f C:\Temp\Sorted\.pf\f43979984_EXCEL_EXE.pf |
Select-String -Pattern "xlsx"
```

b. Based on the file names contained inside the prefetch file, what is a good candidate for the original filename for one of the **Excel** spreadsheets we recovered and reviewed earlier? (hint: compare the data in the **Excel** file with the file names referenced in the prefetch file)

UNDERCOVER-AGENTS-LIST-FOR-UNITED-STATES.XLSX

BONUS: Using the same technique, can you find any other file references that might lead you to programs of interest? List the name of the program and what user profile is it associated with (there are at least two)?

There is a reference to *NROMANOFF\APPDATA\ROAMING\SKYPE\ROMANOFF.NATASHA* that may yield interesting results.

Another possible interesting file is *TWEETDECK-APP.LOG* which is also under the nromanoff profile

Both of these can be found under the Files referenced section of the PECmd output.

21. Finally, let's look at the **lnk** files that were recovered. Open a **PowerShell** window using the shortcut on the **Desktop**. Use **LECmd.exe** to parse all .**lnk** files by running the following command:

```
.\LECmd.exe -d C:\Temp\Sorted\.lnk\ --csv C:\Temp\ -q
```

This command processes every **lnk** file found in the directory specified and writes out the results to a CSV file in the **C:\Temp** directory. The **-q** switch prevents the contents of each **lnk** file from being displayed to the screen, which speeds up processing.

a. How many total .**lnk** files were found for processing?

68

b. How many .**lnk** files were processed successfully?

17

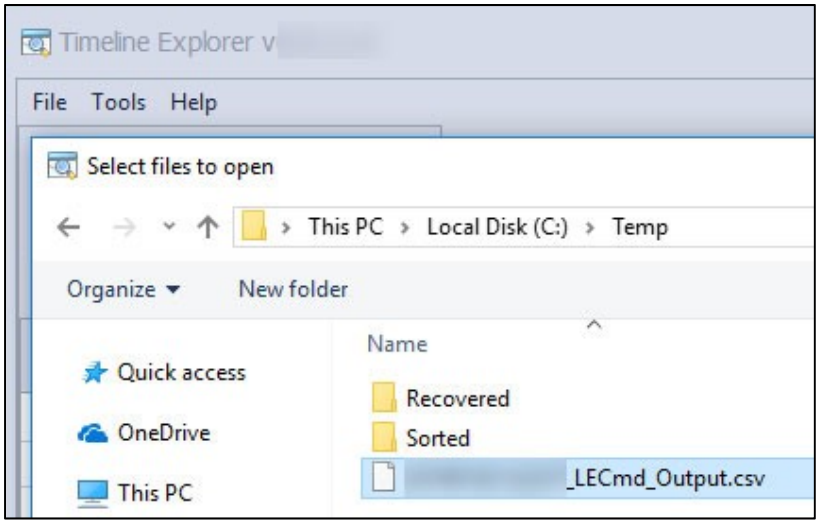
Notice that not even half of the carved lnk files were parsed successfully. Recall from our earlier discussion that file carving often produces a lot of false positives. The **lnk** files that failed to parse are examples of this. For one reason or another, the .**lnk** files that gave errors were corrupted, or otherwise incomplete.

In some cases, this is a function of the data just not being there. In others, it is a function of the tool doing the carving. This is another good time to remind you to test your tools (and use different tools!) to see how they perform in different circumstances. You may just find that one tool carves better than another for certain kinds of things.

As an example, **X-Ways Forensics** was used on **WindowsImage.E01** file to carve for .**lnk** files (in free space only) and it was able to recover 240 files. When **LECmd** was used on these 240 files, **LECmd** successfully parsed 223 out of the 240 files.

This test is a good example of not only how a different tool can potentially find *more* data for you, but also *better* data.

22. Start **Timeline Explorer** from the shortcut in the **Utilities** fence on the **Desktop**. Open the CSV file created by **LECmd** in the previous step by clicking **File → Open** and opening the CSV file located in **C:\Temp**.



23. Once the file is loaded, clear any active filters, then scroll over to the **Local Path** column and answer the following questions:

    a. Do you see any indication that any .**xlsx** files were opened according to the recovered .**lnk** files?

       No

    b. Do you see any executable names that tie back to what we discovered when looking at the prefetch output for Excel? Which one? (hint: is there an executable name here that was also in the prefetch file list?)

       There is a reference to TweetDeck.exe in one of the lnk files.

    c. List any other file names of interest you may want to investigate (non-executables) based on what is in the CSV:

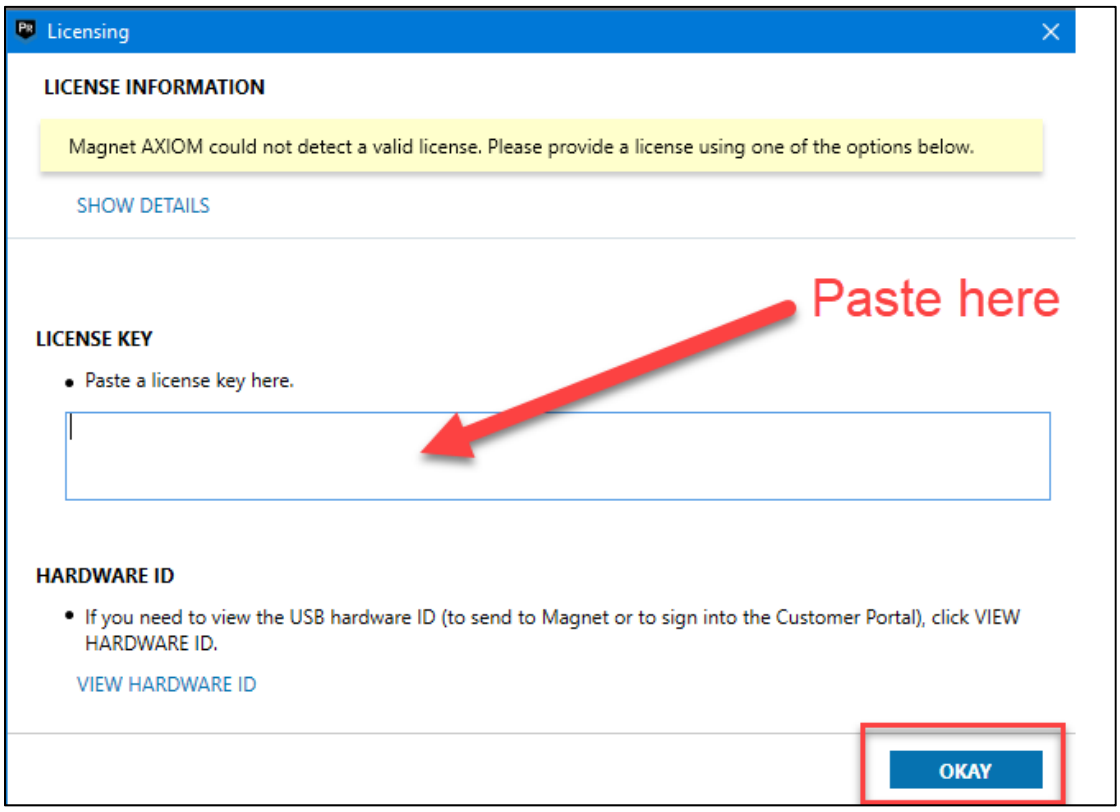
       C:\Users\nromanoff\Documents\Armor Files\0071428674_Section13.pdf
       C:\Users\nromanoff\Documents\Ninja Files\PPT\StickNinja.ppt

While **AXIOM** can carve for data from various sources, we are going to continue to look at the data we recovered via **PhotoRec** inside **AXIOM**, to see what else we can find out from this data.

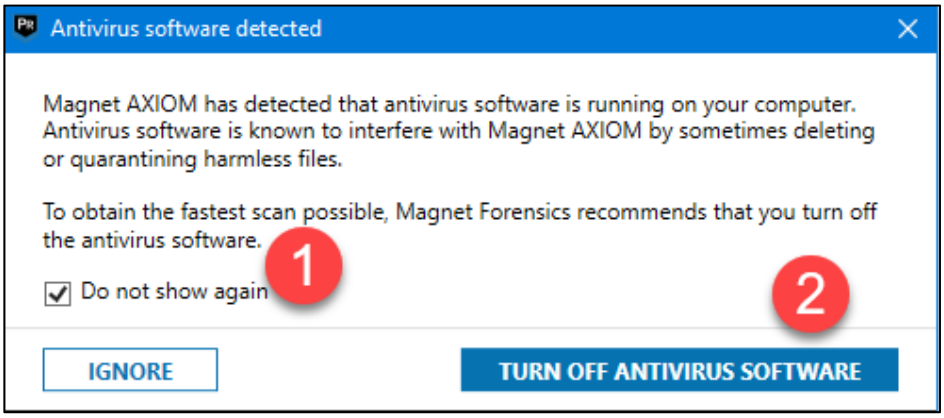
NOTE: If you are prompted to update **AXIOM**, do **NOT** update as it is a large download.

24. Start **AXIOM Process** via the shortcut in the **Forensic Suites** fence on the **Desktop**. This will let us create a new case and begin processing our data. We will use a separate program, **AXIOM Examine**, to review the data once it is processed.

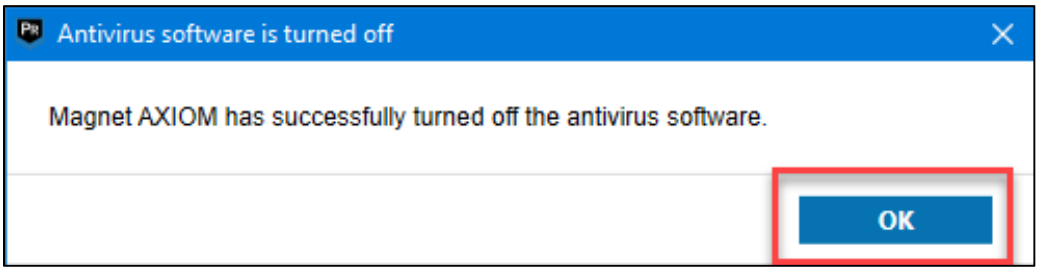
25. If **AXIOM** prompts for licensing information, paste the key provided by the instructor or your OnDemand SME into the box shown below, then click **OKAY**.



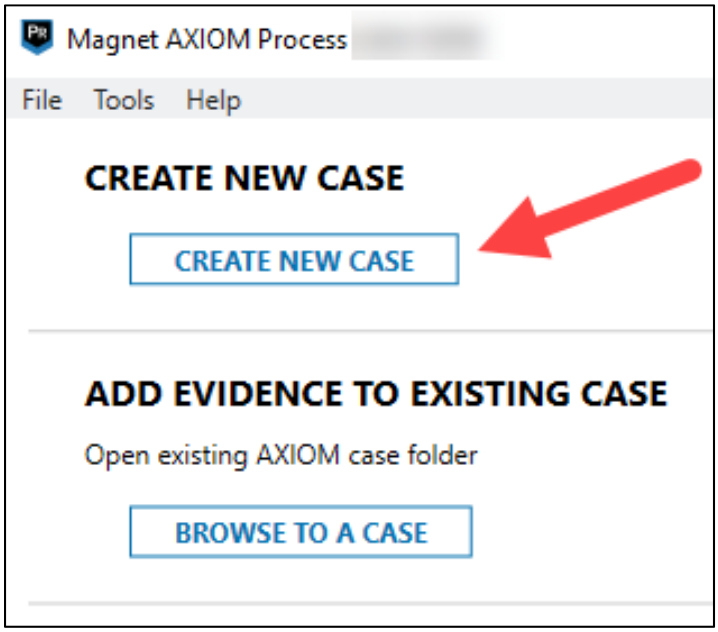
26. When prompted to disable antivirus software, check the **Do not show again** box and click the **TURN OFF ANTIVIRUS SOFTWARE** button.



27. Click **OK** in the confirmation dialog.

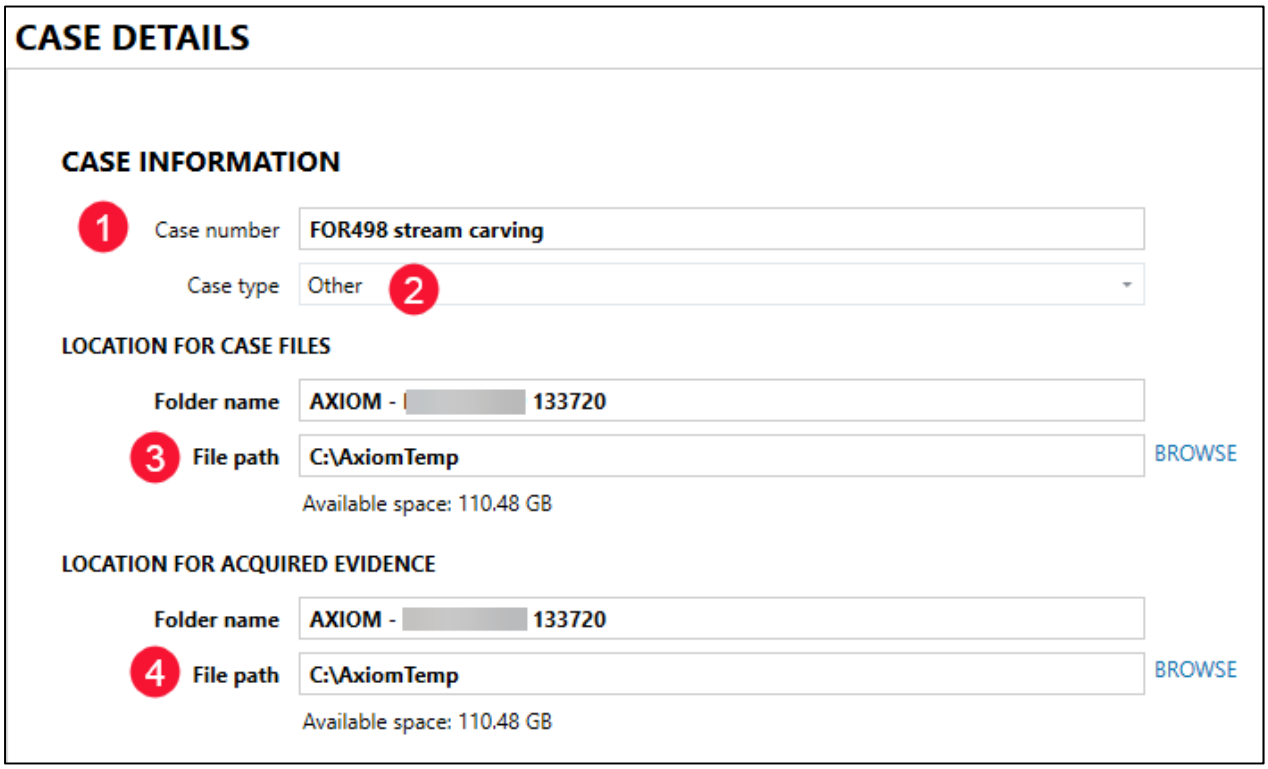


28. Once the interface is loaded, click the **CREATE NEW CASE** button



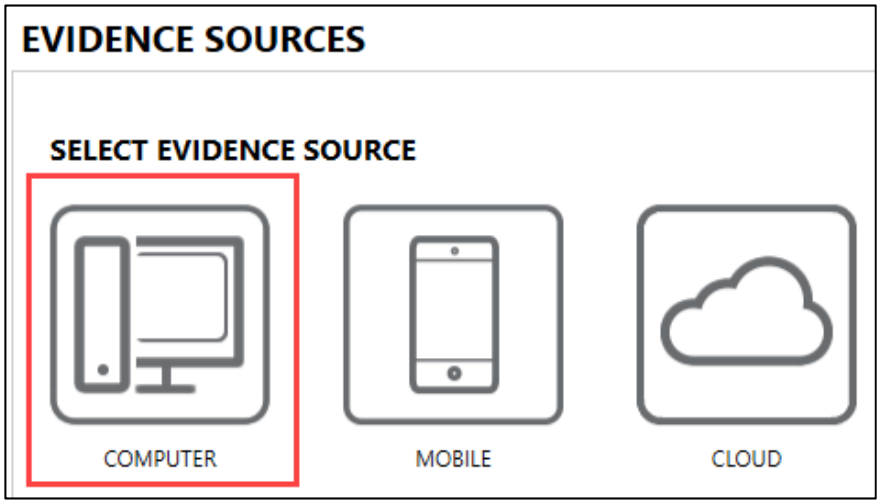
29. In the **CASE DETAILS** dialog, enter the following:

    Case number: **`FOR498 stream carving`**
    Case type: **Other**
    Case files file path: **`C:\AxiomTemp`**
    Location for acquired evidence file path: **`C:\AxiomTemp`**

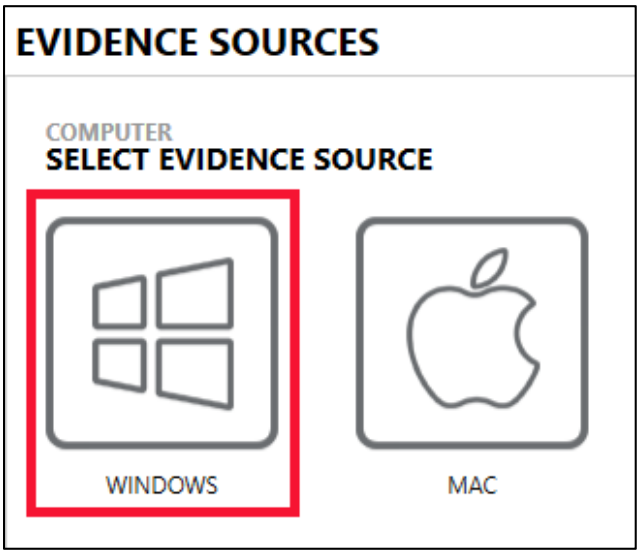


30. Click **GO TO EVIDENCE SOURCES** in the lower right.

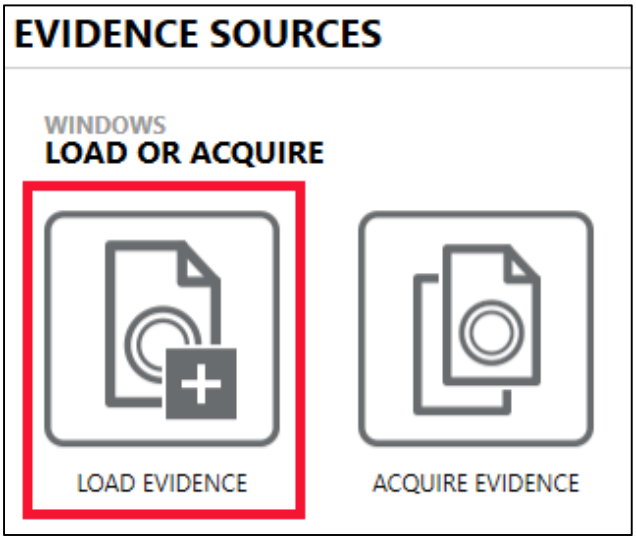
31. In the **EVIDENCE SOURCES** dialog, click **COMPUTER**…



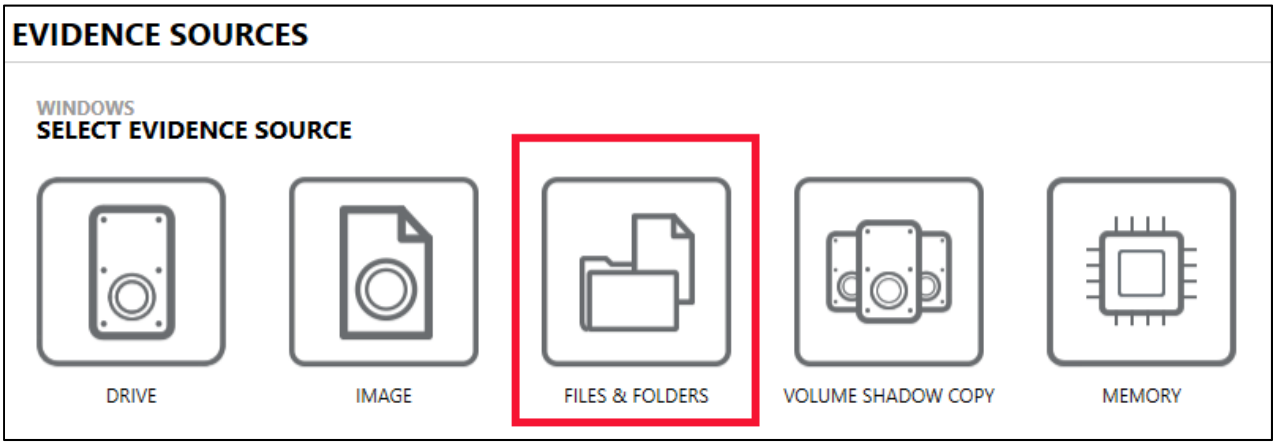
32. …then click on **WINDOWS**, and **NEXT**.



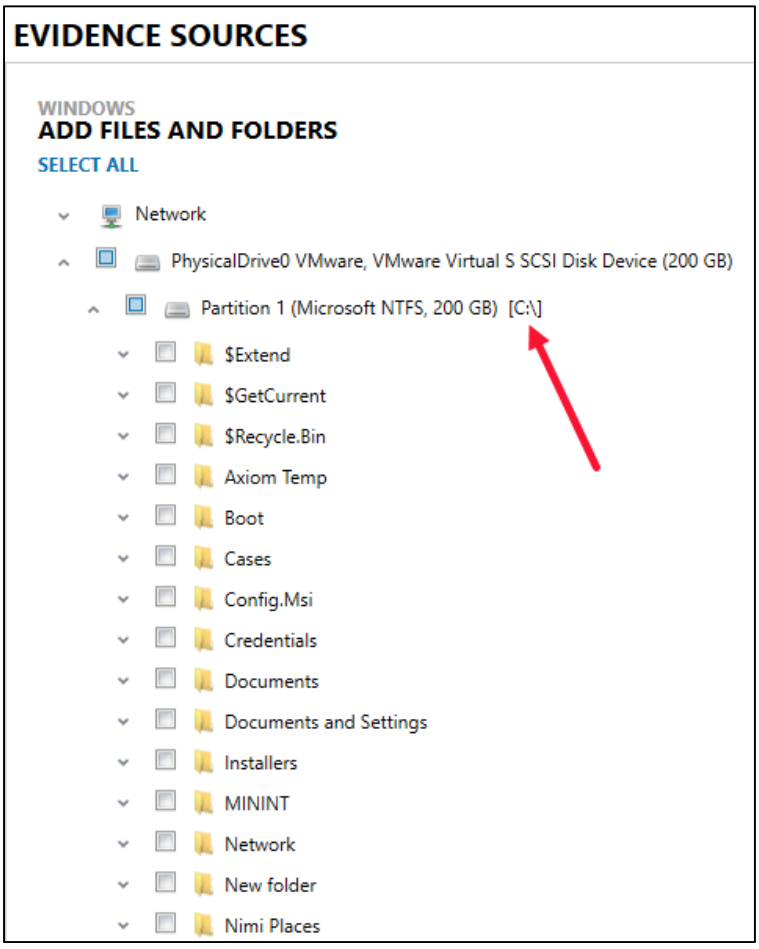
33. In the **LOAD OR ACQUIRE** dialog, click **LOAD EVIDENCE**.

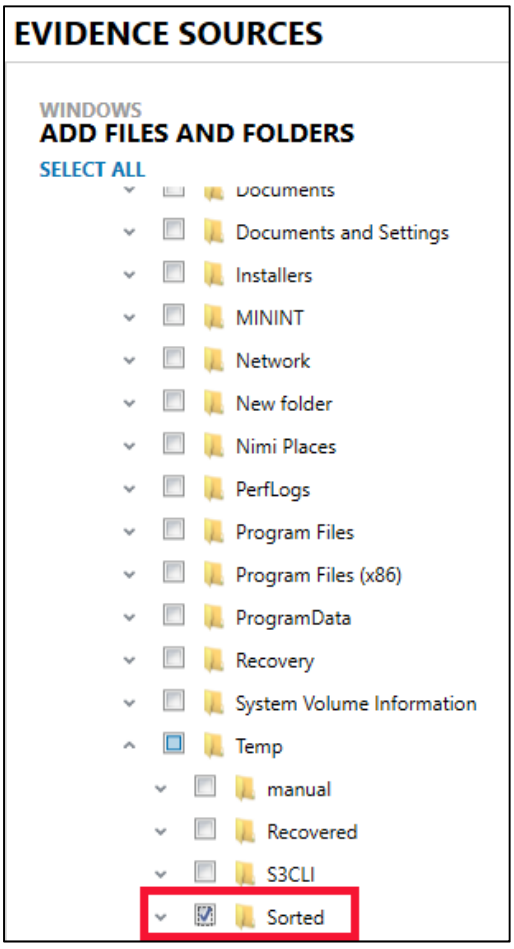


34. In the **SELECT EVIDENCE SOURCE** dialog, click **FILES & FOLDERS**.



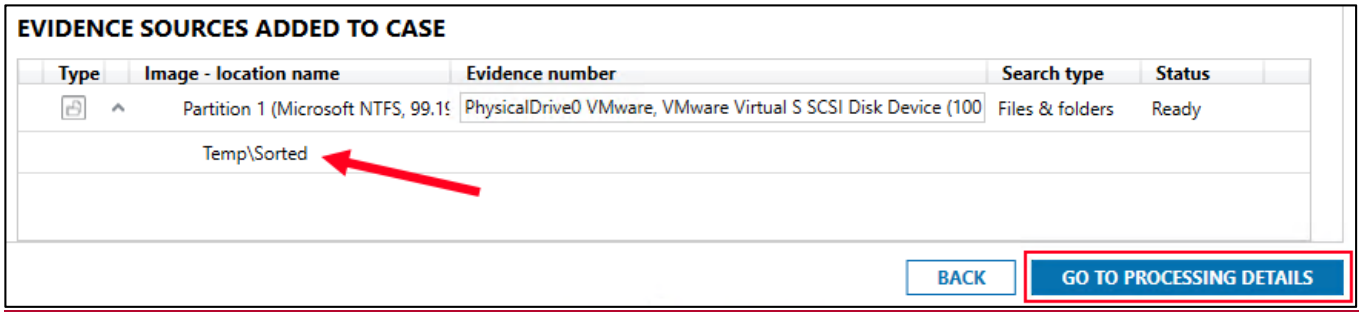
35. Expand the physical drives, looking for **C:\.** Once this is found, expand this folder and scroll down, looking for the **Temp** folder. Once found, expand **Temp** and you should see the **Sorted** directory. Check the box to the left of the **Sorted** directory.





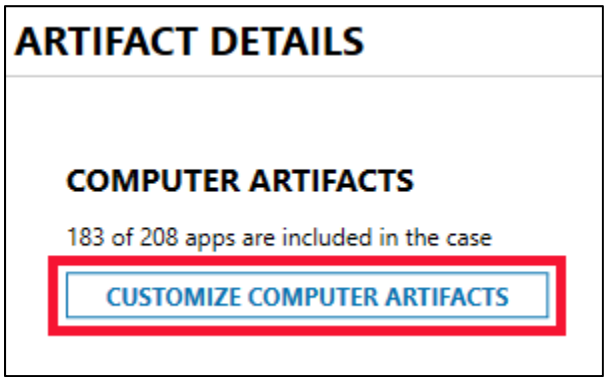
36. Click **NEXT.**

37. Review the evidence sources to confirm the **Temp** folder is listed, then click **GO TO PROCESSING DETAILS** in the lower right.

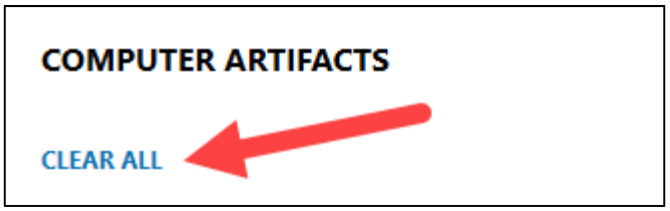


38. **PROCESSING DETAILS** allows you add things like keywords, hash values, etc. but we will skip this functionality. Click **GO TO ARTIFACT DETAILS** in the lower right.

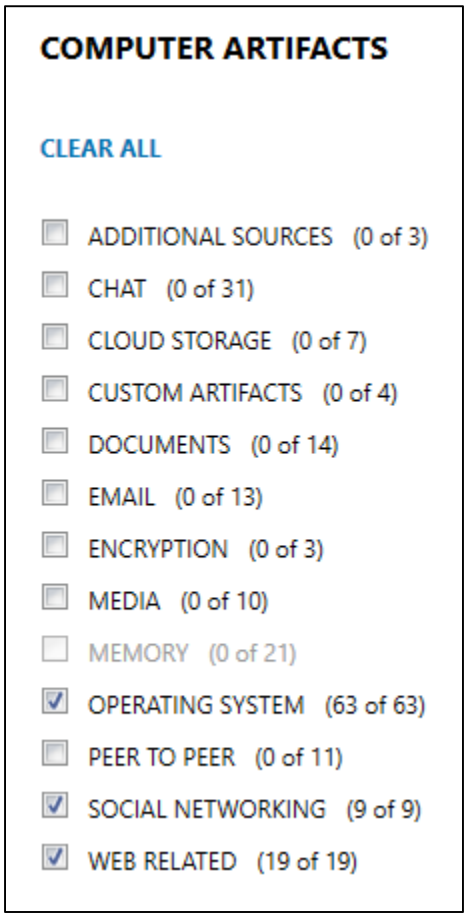
39. Click the **CUSTOMIZE COMPUTER ARTIFACTS** button.



40. Click **CLEAR ALL** at the top of the list.

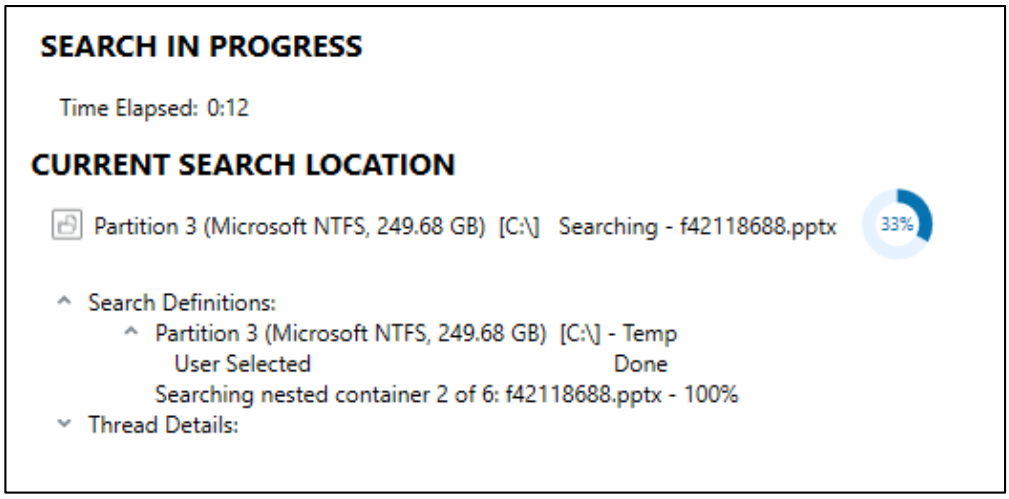


41. Check **OPERATING SYSTEM**, **SOCIAL NETWORKING**, and **WEB RELATED**.



42. Click **GO TO ANALYZE EVIDENCE** in the lower right.

43. Click **ANALYZE EVIDENCE** in the lower right to begin analyzing the data. Once processing has started, progress is displayed.

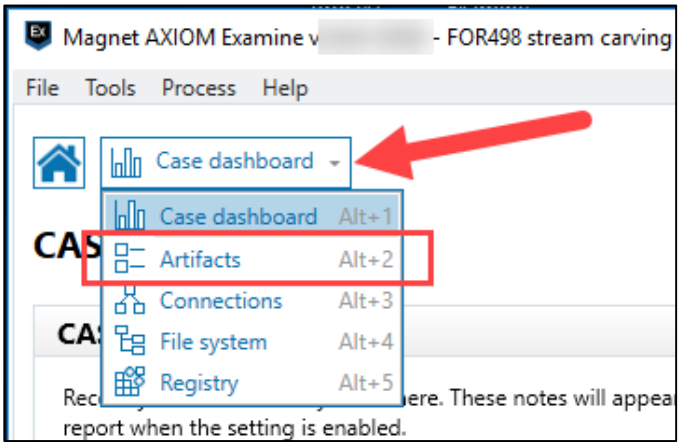


Once processing has started, **AXIOM Examine** will be automatically started with case data loaded.

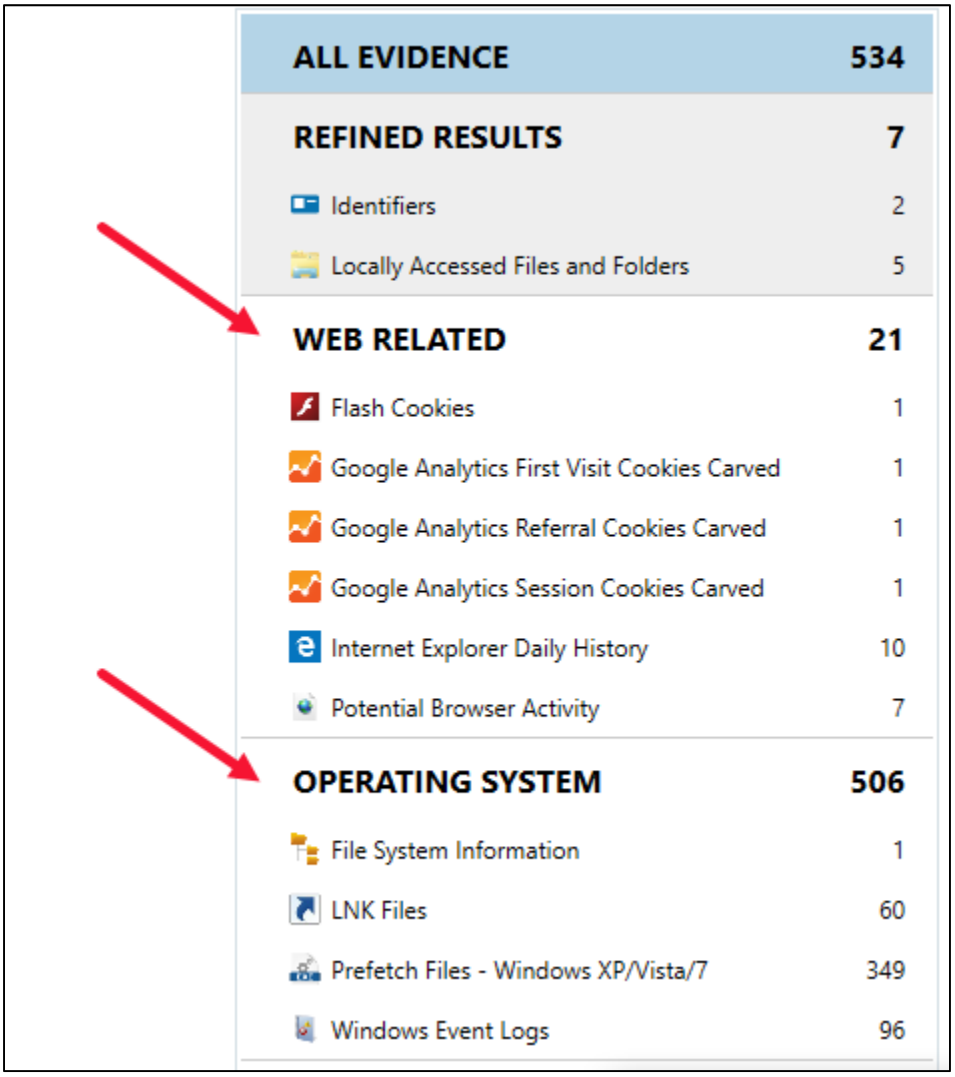
44. Switch to the **AXIOM Examine** window. If a **Quick tips** window pops up, close it with the **X** in the upper right corner of the **Quick tips** dialog.

45. The results will be displayed as they are available in the **Examine** interface. When processing is finished (it shouldn't take too long), click the **OKAY** prompt in the lower left (if necessary) to reload the case, so all data is available.

46. The main **Examine** interface is displayed. In the upper left, click the **Case dashboard** drop down and select **Artifacts** from the list (or use the hotkey **ALT+2**)

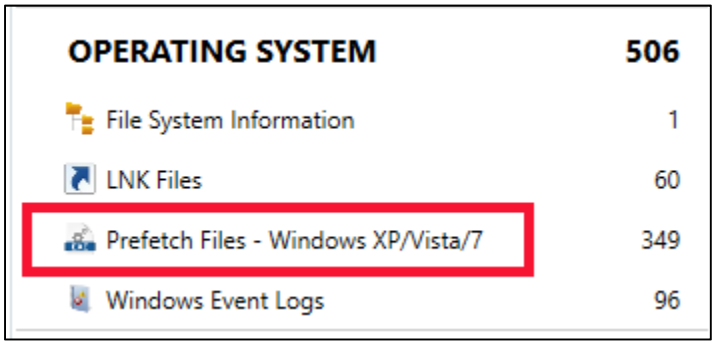


47. Under the **ALL EVIDENCE** section on the left, expand the **WEB RELATED** and **OPERATING SYSTEM** sections by clicking the down arrow to the left of the category name.



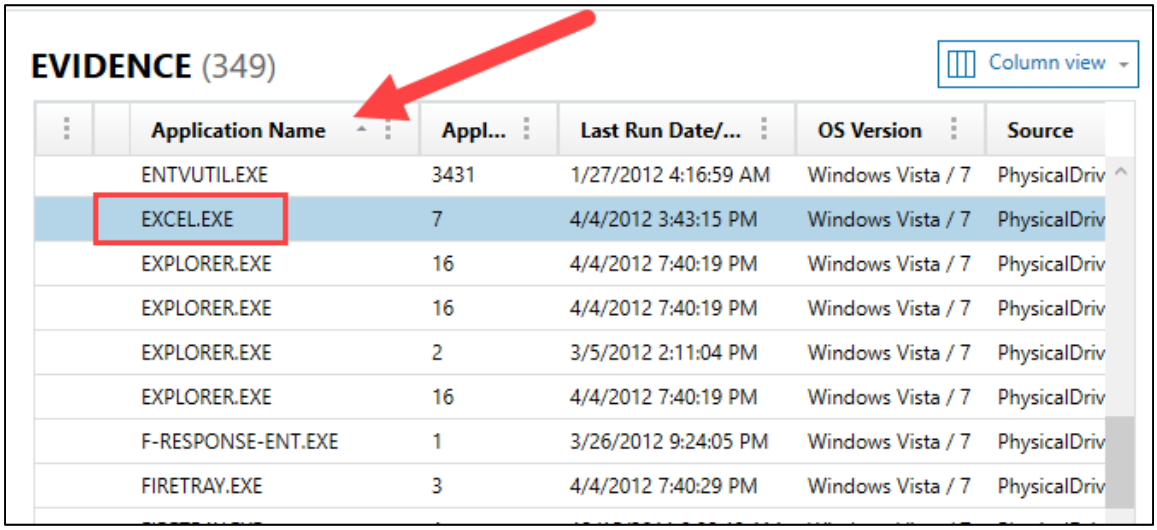
This view of the data provides a summary of the kinds of data found, as well as the number of items found for each item.

48. Click on **Prefetch Files** in the **OPERATING SYSTEM** group.

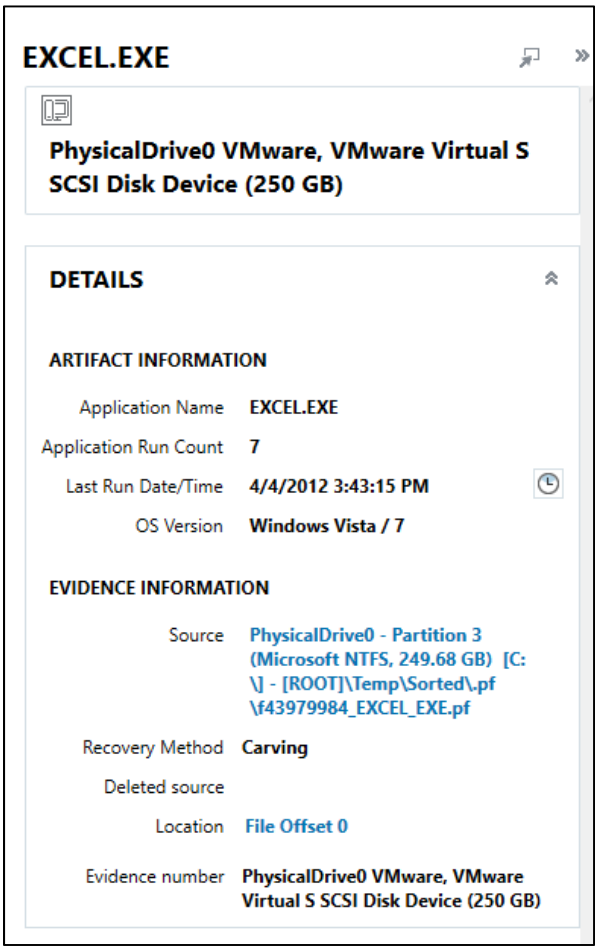


This loads the details in the grid to the left of the summary.

49. Click the **Application Name** column header to sort by that column, then scroll until you find the **EXCEL.EXE** entry.



50. Clicking on **EXCEL.EXE** loads the details for this entry to the right of the grid.



a. How many times was **EXCEL.EXE** run?

7

b. When was the last time **EXCEL.EXE** was run?

2012-04-04 3:43:15 PM

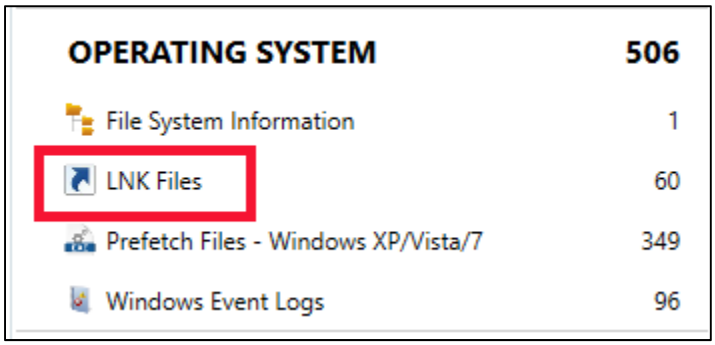
c. Do these values match what you found when using **PECmd** to look at this prefetch file?

Yes, PECmd showed the time in 24 hour time, but the time is the same.

d. What is missing from this view of the prefetch file that **PECmd** showed?

All of the directories and files that were referenced.

51. Under the **OPERATING SYSTEM** artifact list, click **LNK Files**.



a. How many **lnk** files did **AXIOM** parse?

60

52. Click the **Linked Path** column header to sort by that column. Review the entries under **Linked Path**. You may need to scroll the list so the entries with data are shown.

a. Does **AXIOM** list any additional files (non-executable) that **LECmd** did not list? If so, what file names?

No

Recall from earlier that **LECmd** was only able to parse 17 **lnk** files, but here we see 60 were parsed. If you look through the list of **lnk** files, you will see that a lot of files do not have anything listed for **Linked Path** (and many other columns as well).

This is an example of stream carving, in that **AXIOM** parsed out what it could from the available **lnk** files, whereas **LECmd**, when it encountered bad data, stopped processing the **lnk** file.

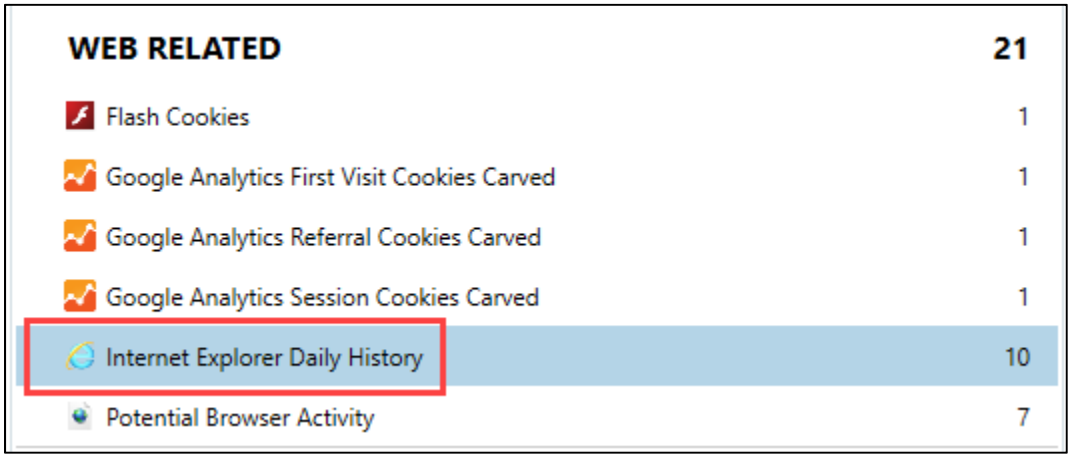
53. Spend a minute or two looking at the **lnk** files where **Linked Path** is blank.

    a. Do any of these contain useful information in the other columns? If so, what?

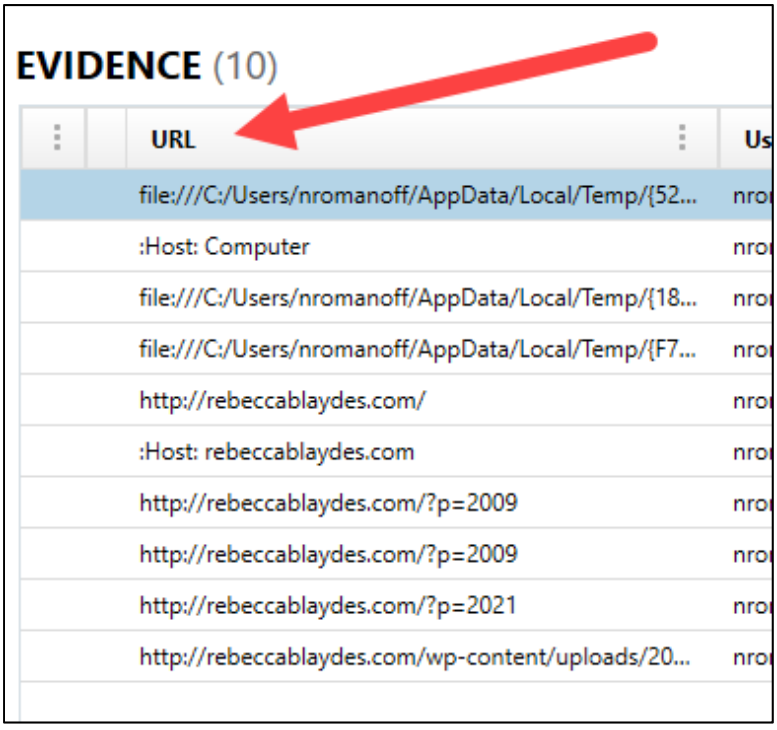
    Potentially, arguments, computer names (in NETBios Name), volume serial numbers, etc.

    This is illustrative of the balancing act that tools take between parsing and displaying what it can with damaged **lnk** files (**AXIOM**) even though the most relevant data may not be found vs. erroring out when incomplete or unexpected data is encountered (**LECmd**).

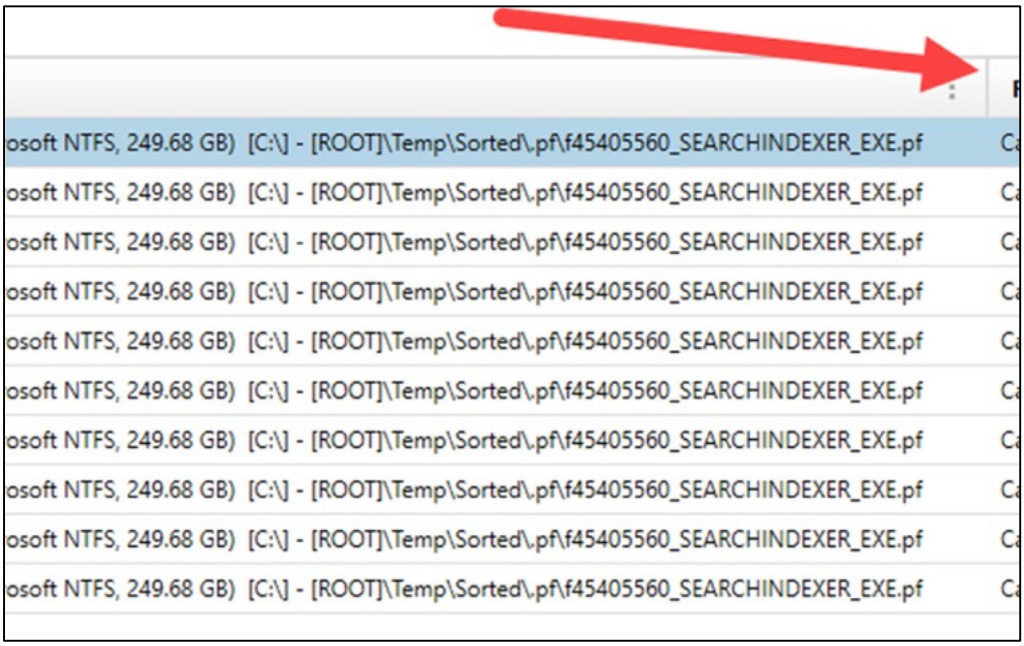
54. Under the **WEB RELATED** artifact list, click **Internet Explorer Daily History**



55. Take a moment to review the URLs under the **URL** column. Recall that we did not file carve for any browser history related artifacts in **PhotoRec**, yet **AXIOM** has found some!



56. Locate the **Source** column and expand it so the entire source is visible. To do this, hover over the column separator, then left click and drag to increase the column's width. You are in the right spot when the cursor changes to a horizontal, double sided arrow.



a. What is the full path and file name of the file where the URLs were all found?

Temp\Sorted\.pf\f45405560_SEARCHINDEXER_EXE.pf

57. Open a **PowerShell** window using the shortcut on the **Desktop**.

58. Use **PECmd** to parse **C:\Temp\Sorted\.pf\f45405560_SEARCHINDEXER_EXE.pf** with the following command:

```
.\PECmd.exe -f C:\Temp\Sorted\.pf\f45405560_SEARCHINDEXER_EXE.pf
```

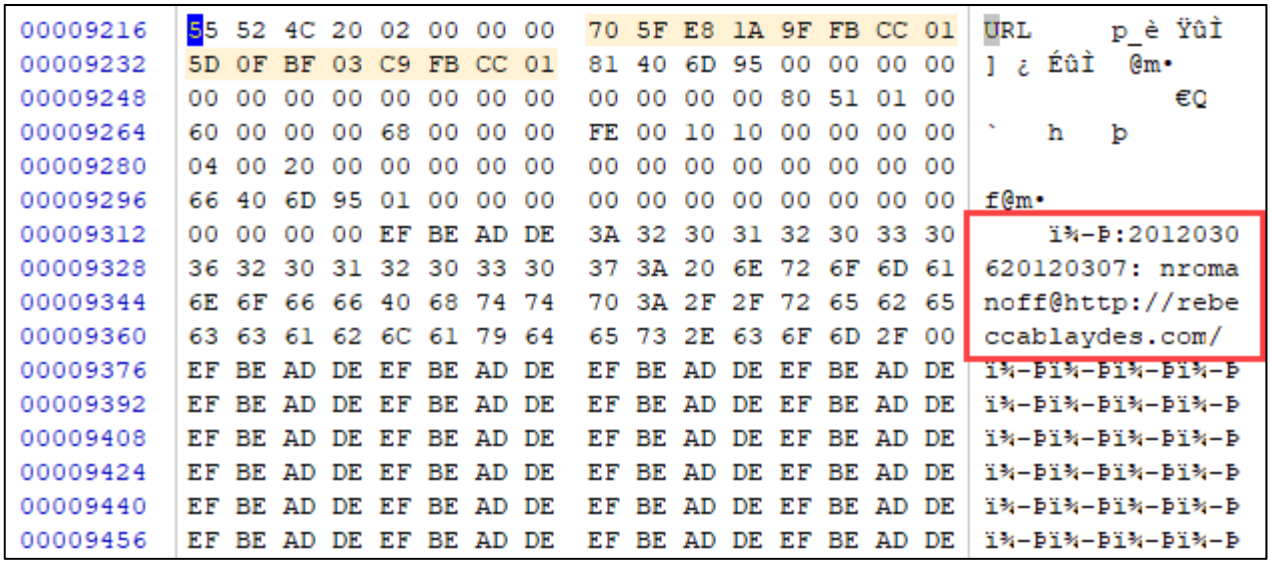
a. Did the command complete successfully?

No. The program indicated that only partial output was shown.

b. Was **PECmd** able to extract any data from the prefetch file, such as run count and last run timestamp?

Yes, a run count of 4, and a Last run timestamp of 2012-01-11 08:17:04.

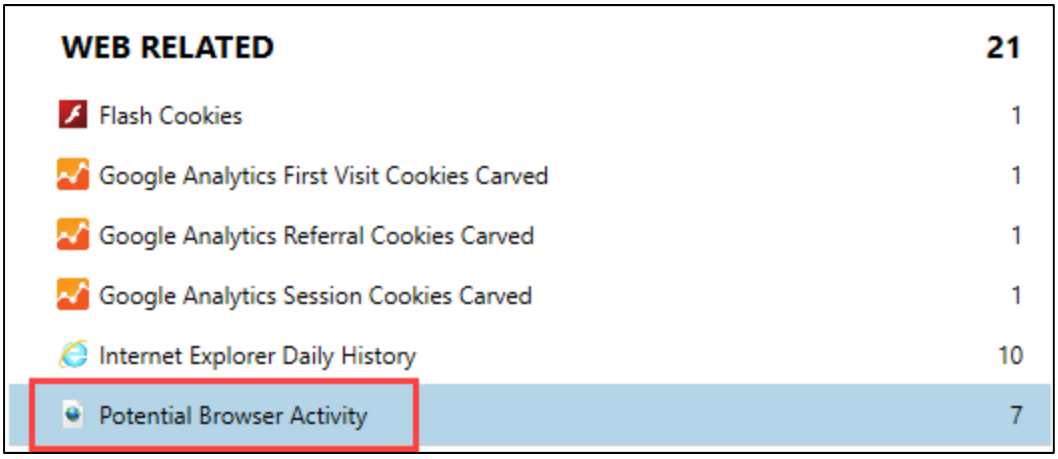
You would not normally expect to find URLs in a prefetch file, but **AXIOM** found some. How did it do this? Recall that **AXIOM** is looking for "data within data". If you were to open the .**pf** file where the URLs were found in a hex editor, you could visually see the URLs, like this:



But how did that data even get in this prefetch file in the first place? Well, as we said before, carving is a best effort endeavor. When **PhotoRec** was looking for prefetch files, it found a valid header, then did its best to recover the prefetch file contents. In this case, however, we have data from a non-prefetch file as well. Because **AXIOM** is looking for data within data, it found and displayed the URL.

This is another great example of stream carving (i.e. looking for data within data) vs. file carving. In many cases, stream carving can recover useful information inside files that may not parse with a native parser (like **PECmd** in this case).

59. Under the **WEB RELATED** artifact list, click **Potential Browser Activity**



60. Locate the **Source** column and expand it so the entire source is visible. To do this, hover over the column separator, then left click and drag to increase the column's width.

    a. How many unique files were these URLs found in (hint: sorting on the **Source** column will make this easier)

       Three files

61. In a **PowerShell** window, confirm you are in **C:\Tools**. Use **PECmd** to parse **C:\Temp\Sorted\.pf\f43410352_WMIPRVSE_EXE.pf** with the following command:

```
.\PECmd.exe -f C:\Temp\Sorted\.pf\f43410352_WMIPRVSE_EXE.pf
```

    a. Did the command complete successfully?

       No, PECmd said only partial output is shown.

    b. Was **PECmd** able to extract any data from the prefetch file?

       Yes, a run count of 378, and a Last run timestamp of 2012-02-29 07:05:20. There were also a count of directories referenced (5) and files referenced (443).

    c. How did parsing this file compare with the data exacted from **f45405560_SEARCHINDEXER_EXE.pf**?

       Looking through the Files referenced section, there are some readable values but most appear to have been corrupted. This is additional information that PECmd was able to recover from this prefetch file when compared to the previous one.

OPTIONAL HOMEWORK: Run **AXIOM** again, but against the entire **WindowsImage.E01**. It is recommended you do this outside of class and let it run overnight as it can take a while to run. Compare and contrast the data we looked at above with all the data that **AXIOM** finds from the entire image.

Exercise—Key Takeaways

- Data and stream carving can bring new leads and insight to your cases.
- There is no guarantee that files recovered via carving will be useful.
- Different tools can produce different results. Test your tools!
- Tools like Axiom can find data embedded inside files, such as URLs, email addresses, and more.

Exercise 6.2—Data Recovery

Background

Data recovery is a very misunderstood area of practice. A simple Google search will bring up all manner of data recovery companies that aren't really data recovery companies. In addition, they don't even exist where they say they exist, when they exist. Beyond this misleading shell game, there is an entire underbelly of the industry that is rife with people who think that data recovery is easy, and that you just need a freezer to save the day. YouTube has hundreds, if not thousands of videos on how to perform all manner of data recovery, from dead heads, to blown motors and burnt chips.

The reality is much different. There are several different facets to data recovery, with diagnosis being the most important of all of them. Without a proper diagnosis, any attempts are, at best, going to result in absolute failure, and at worst, are going to damage or destroy a drive that otherwise could have been recovered by someone with the appropriate skills. Part of the diagnosis is to determine whether the problem is a logical issue or a physical issue. A logical issue would be anything that has to do with the data and how it exists on the drive. A corrupt or overwritten MFT or partition table would be a good example. Physical issues are issues that no software can fix by itself. This would include things such as damaged or failed chips on a circuit board, or non-functional read/write head stack assemblies.

From a purely logistics perspective, we cannot create a lab environment where every student has a non-functional hard drive with exactly the same problem as everyone else. Because data recovery is a physical thing that happens in the physical world, we cannot create problems in a VM. What we can do is create a logical problem on a drive, then create an image of that drive to work with and try to fix.

In this exercise, a very common issue will be addressed, which is that of a damaged partition and MFT. Here then, are two real world scenarios that can cause this failure state.

Scenario 1
The user was using their computer when it 'blue-screened'. Upon restarting the computer, it went into recovery mode. They did not read the instructions carefully and made the wrong selection between repair and restore. Choosing “restore” caused the computer to format the volume and start re-installing windows.

Scenario 2
The user planned on copying data from one external hard drive to another so that they had a backup. During the first step of preparing the destination drive, they accidentally started wiping the wrong drive and destroyed the partition information and the MFT (Master File Table).

Exercise Objectives

- Use R-Studio data recovery software to recover files from a damaged volume and overwritten MFT

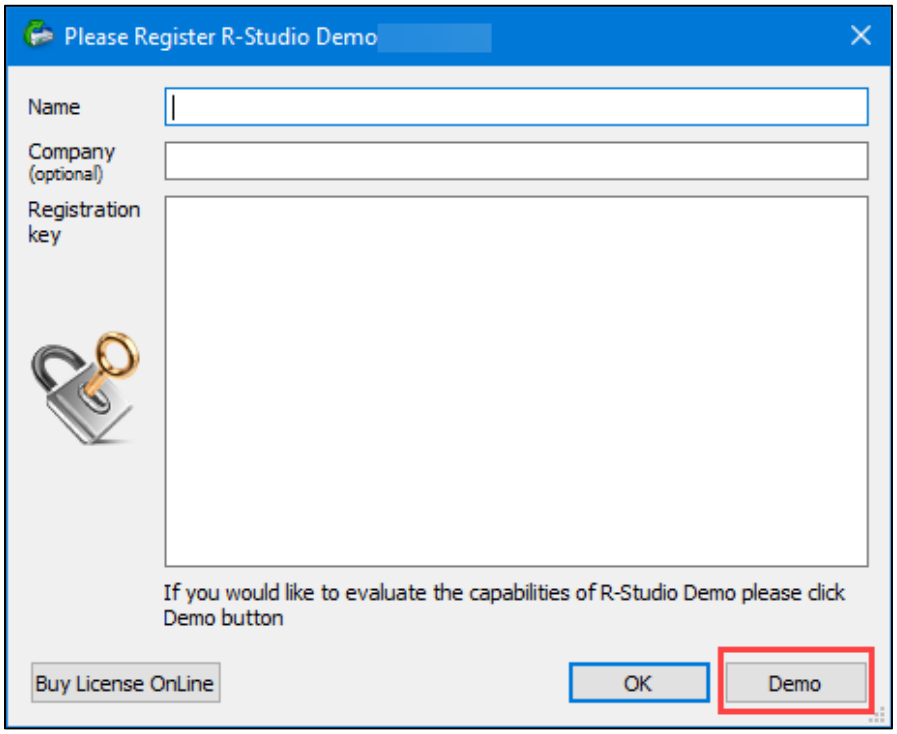
Exercise Preparation

1. Boot your **FOR498 Windows VM**

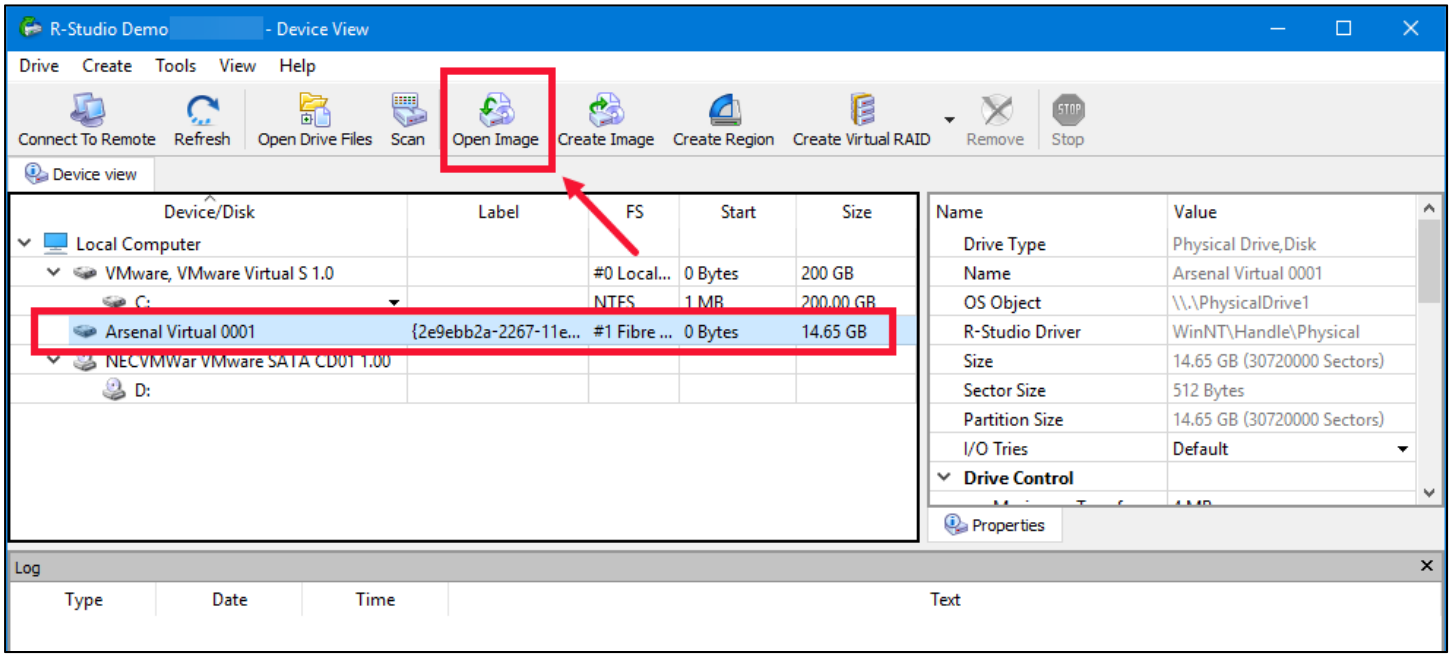
2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Using the techniques you learned in the image mounting exercise (**Exercise 3.2**), mount the **CARVEY.E01** found at **C:\Cases\Exercises\6.3** using **Arsenal Image Mounter**.

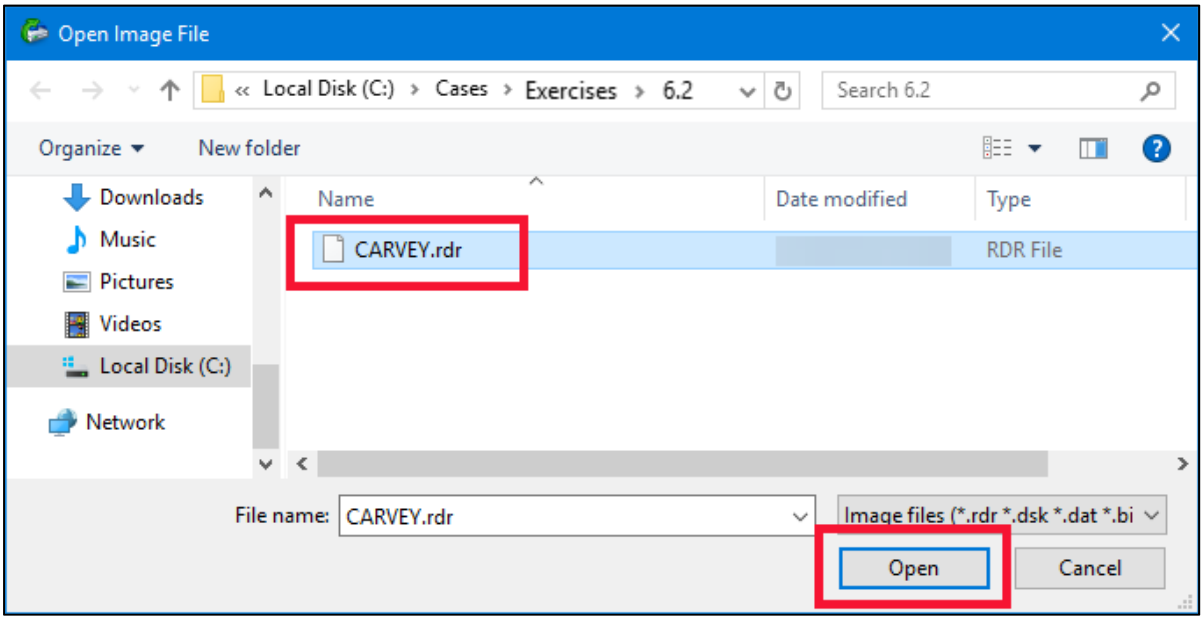
4. Locate the **R-Studio** shortcut in the **Utilities** fence on the **Desktop** and start it. The initial splash screen will indicate that you are working in **Demo** mode. There is no time constraint in **Demo** mode, however you are limited to the size of files you can recover (nothing bigger than 256 KB). Click on the **Demo** button.



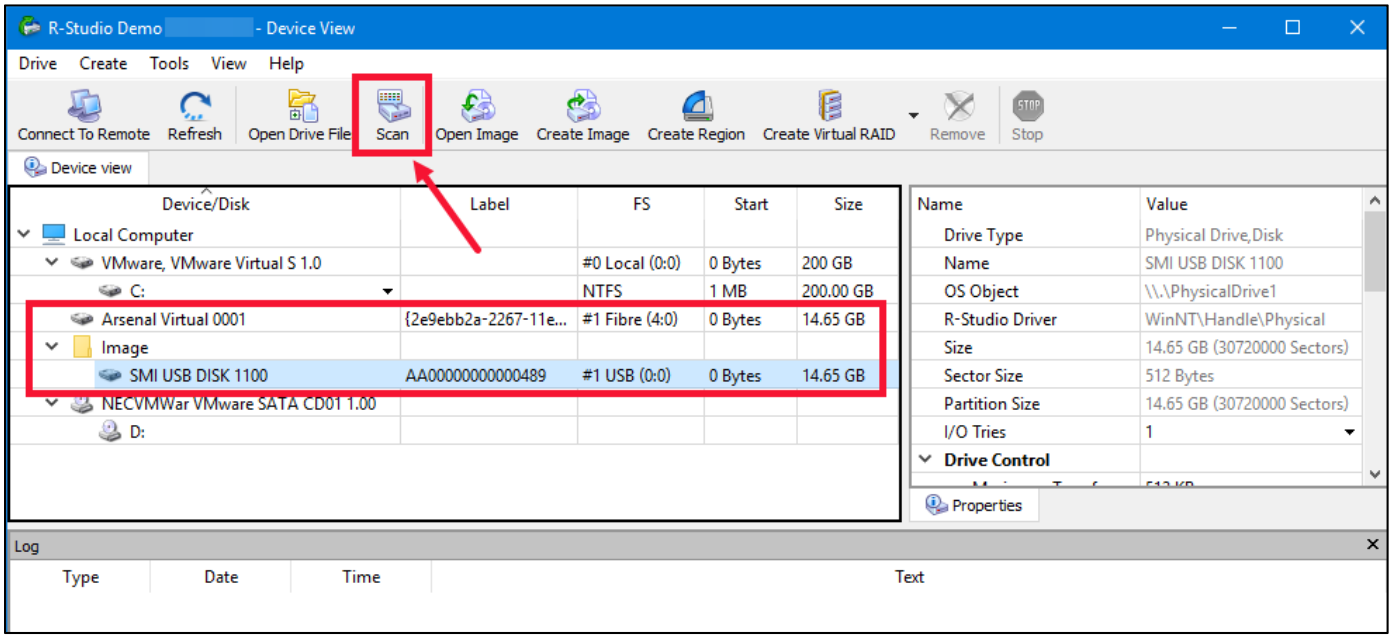
5. The program will open, and a listing of all detectable drives will be shown, irrespective of file system or file system integrity. Locate and select **Arsenal Virtual 0001**. Although we cannot provide a failed hard drive to every student and replicate the problem exactly across all of them, an image file of a damaged hard drive has been created, and we will work from this. Click on the **Open Image** icon.



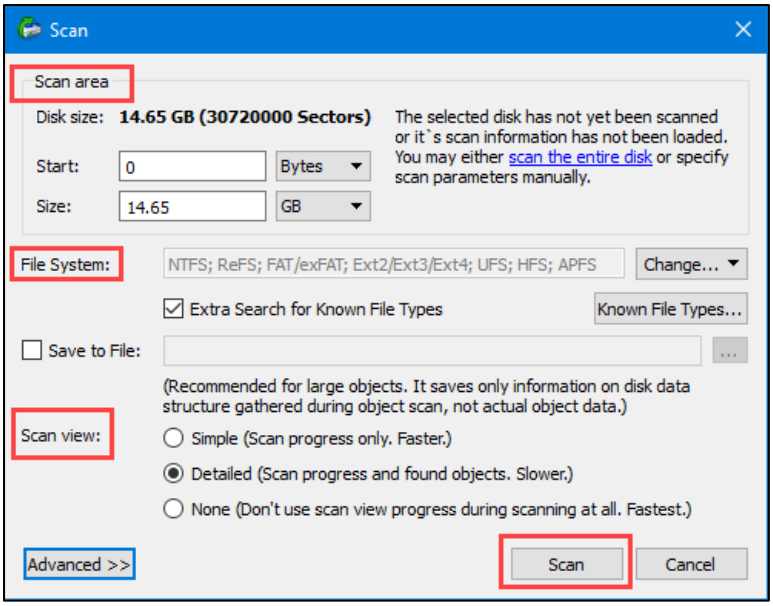
6. In the **Open Image File** dialog, navigate to **C:\Cases\Exercises\6.2** and select **Carvey.rdr** from the list, then click on the **Open** button.



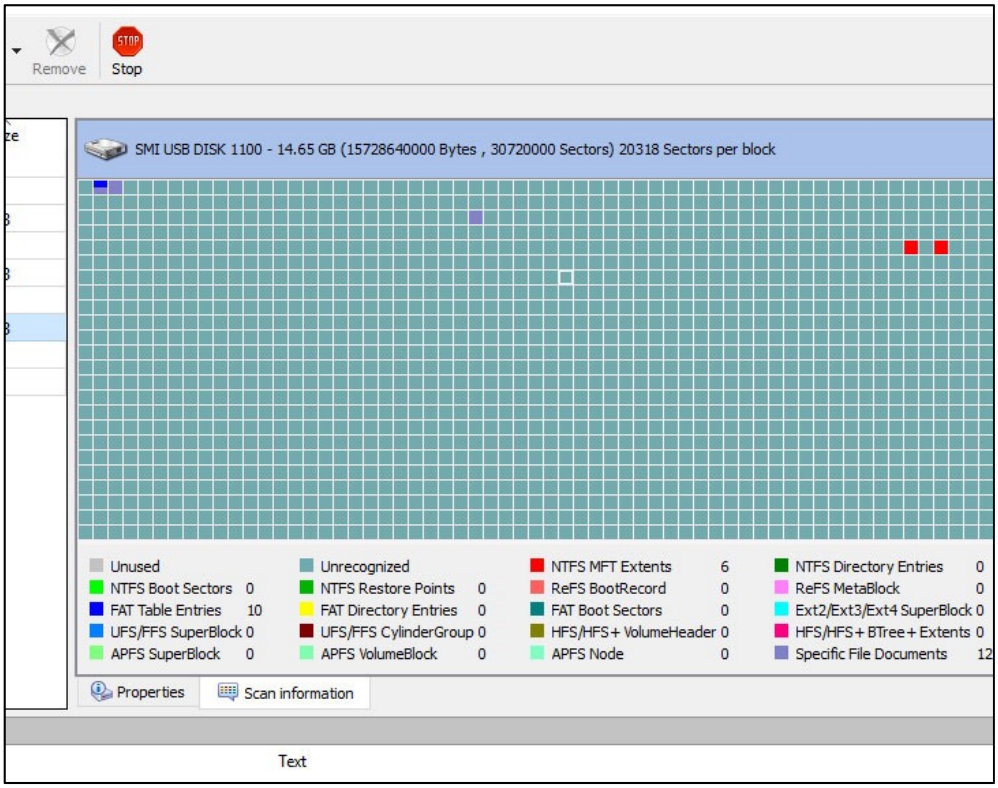
7. The **rdr** file is an **R-Studio** specific image of the same physical drive used to create the E01 you mounted with **AIM** earlier. Once the **rdr** is opened, it will be mounted and details about the image are shown, including a more accurate representation of the drive's space as well as the name and serial number of the physical drive. Click in the newly mounted image, then click on the **Scan** button in the top tool bar.



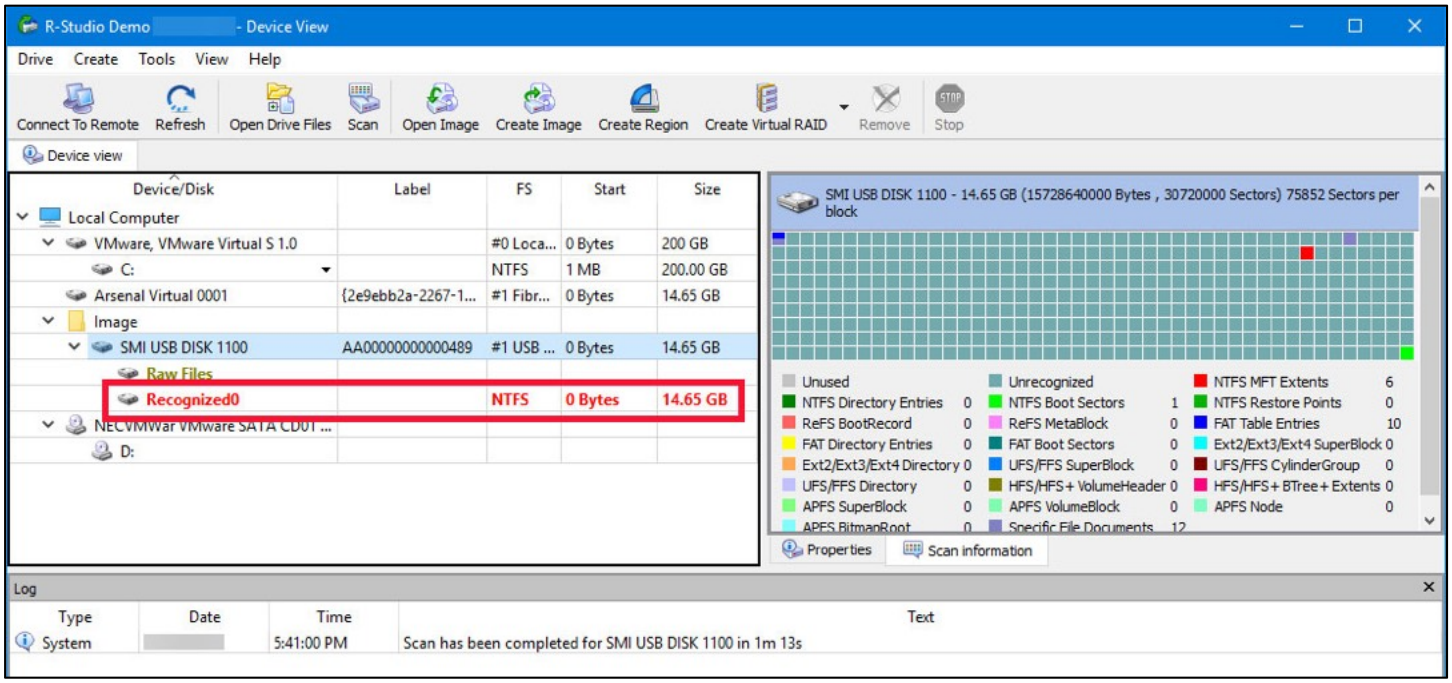
8. You will see a **Scan** dialog appear with several options. This window allows for a great deal of granularity. In the **Scan area**, you can select what physical part of the drive you want to scan. This is useful if you have a damaged area, and don't want **R-Studio** wasting time scanning an area it will never recover. Also, if you know that there is no data on the last half of your drive, you need not waste time scanning it. In the **File System** area, you can select from several file systems, as well as search merely by known file type. You also have the option of saving out the found files to a directory elsewhere. In the **Scan view** section, you can choose to see the scan progress or not, understanding that the more granularity you request, the slower the scan goes. For the purpose of our scan, we will leave all settings as default. Verify that your settings are as shown in the screen below, and click the scan button.



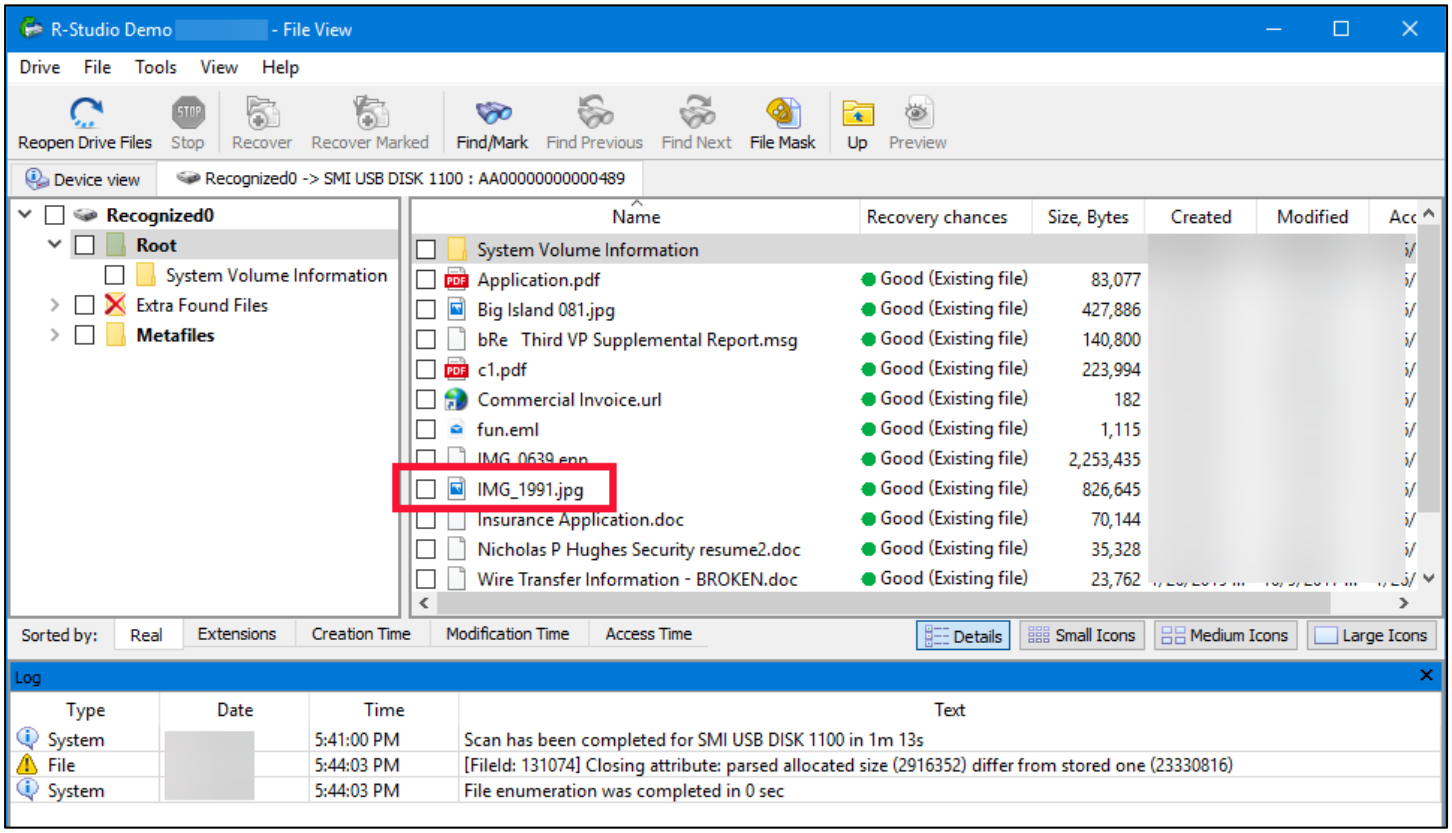
9. The scan will begin, and you will be able to watch its progress as it scans each block of data. A legend outlines the data as it is found.



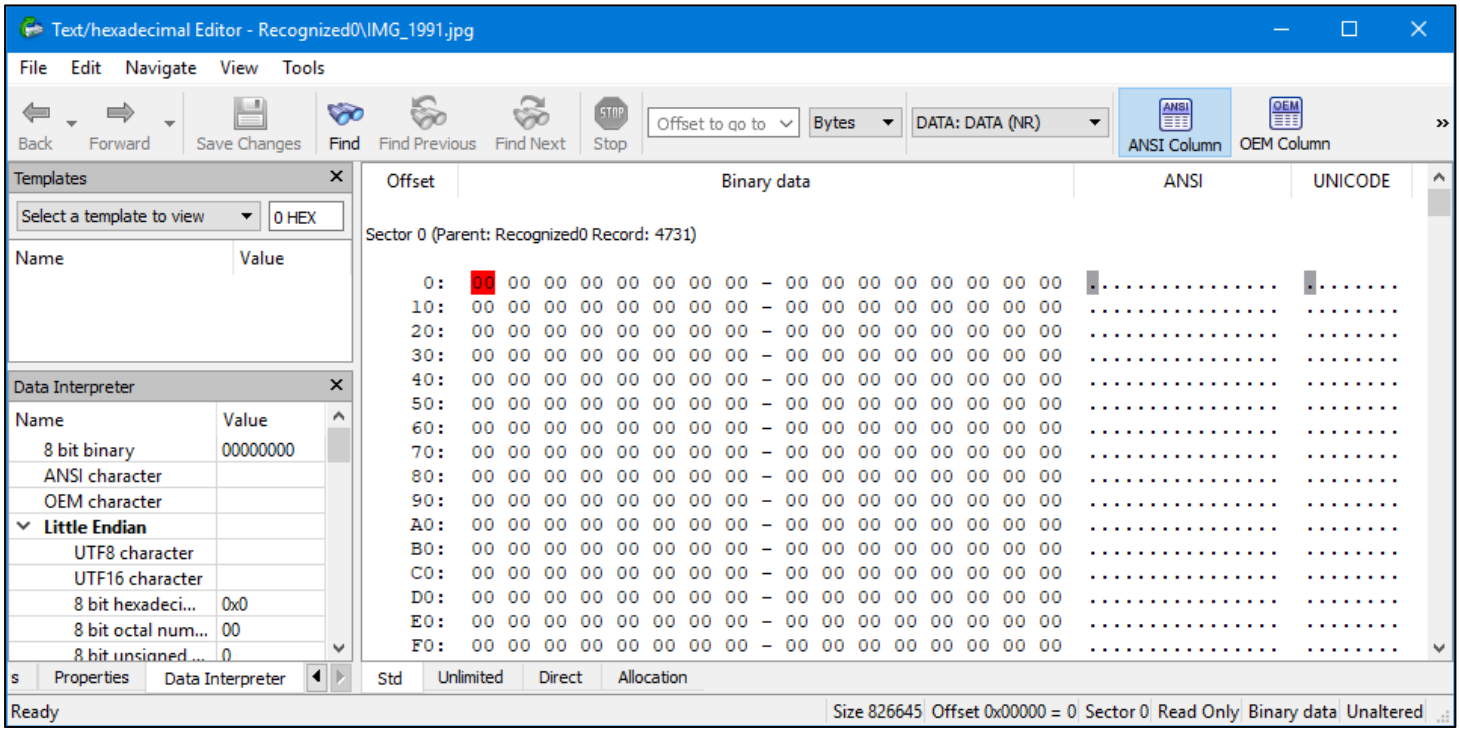
10. Once the scan is complete, you will see a new entry under the mounted **rdr** file. This is the 'best attempt' volume entry that the scan has been able to determine. In some cases, there may be multiples of entries. Typically, you want to select any green ones, but in this case, there is merely a single red one. Although this does not appear to be optimal, we will work with what we have. Double click on the **Recognized0** volume that **R-Studio** is presenting.



11. A new tab will open, and any files and/or folders that have been found will be listed. This looks very promising! You can see several files listed. The **MFT** and partition were destroyed (you can even check it out in **HxD**) and yet, even without a partition or **MFT**, you are somehow able to see actual file names, as well as sizes and other metadata. How is that possible? Also, unbeknownst to many, there is a second copy of the NTFS boot sector at the *end* of a volume, and if you know how, you can recover it and restore it to use for helping recover data on the drive. Under the column **Recovery chances**, you see all green dots. This would seem to be good news too, but this does not verify that a file is actually at the place on the drive that the **MFT** artifacts say so. In order to see the status of your recovery, double click on **IMG_1991.jpg**. If it is intact, it should open.



12. Double clicking on an intact file will cause it to open and show you the file contents. If it is not intact, the program will show you the hex representation of what it found at the address marker it was sent to by the **MFT**. In this case, the file does not exist at this address marker any more. Bad news for us. Close this box.



Exercise Questions

1. Take a moment and double click on all the files that you see in the list.

   a. Did any open?

   _____

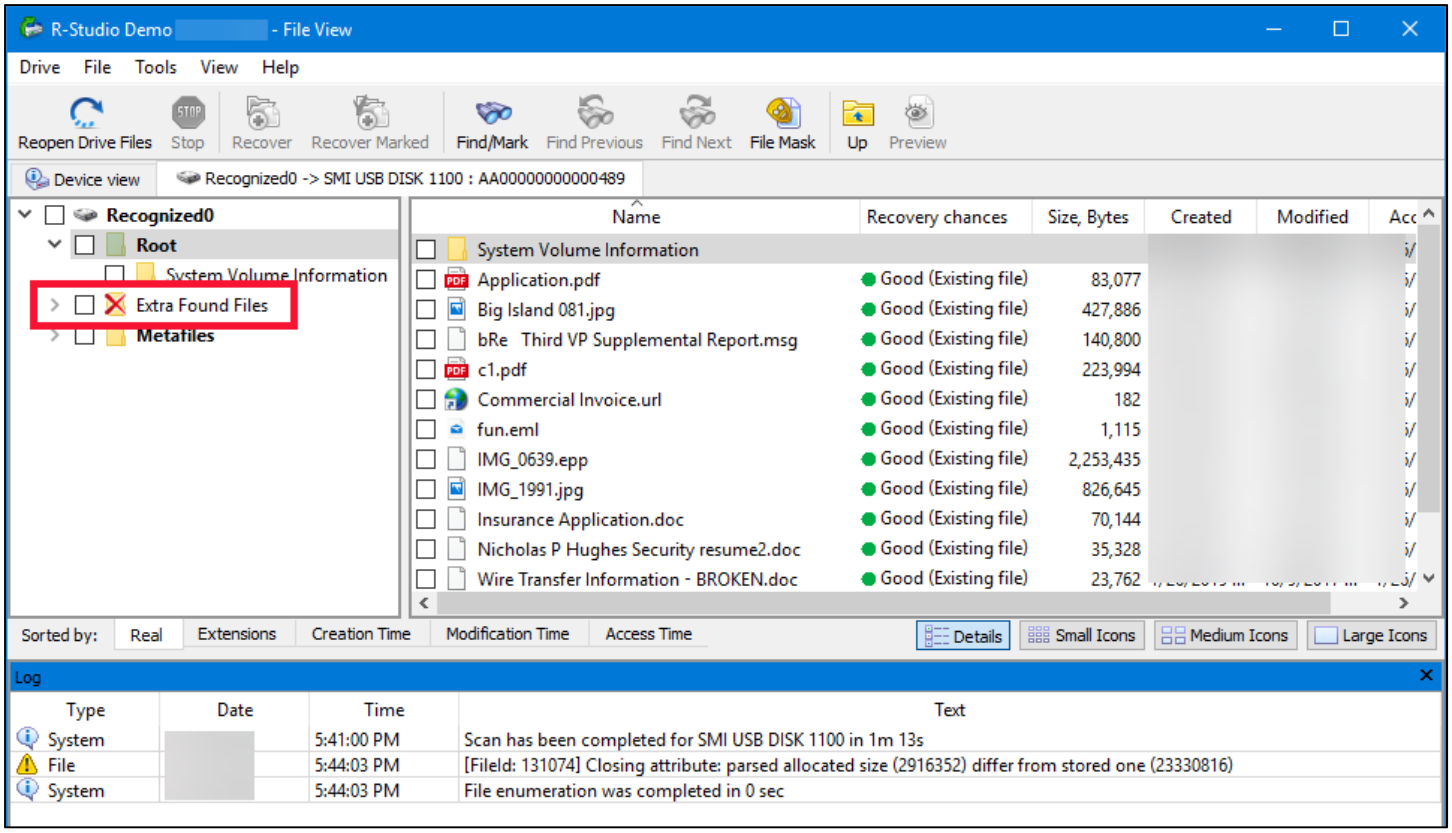
   b. If yes, which ones?

   _____

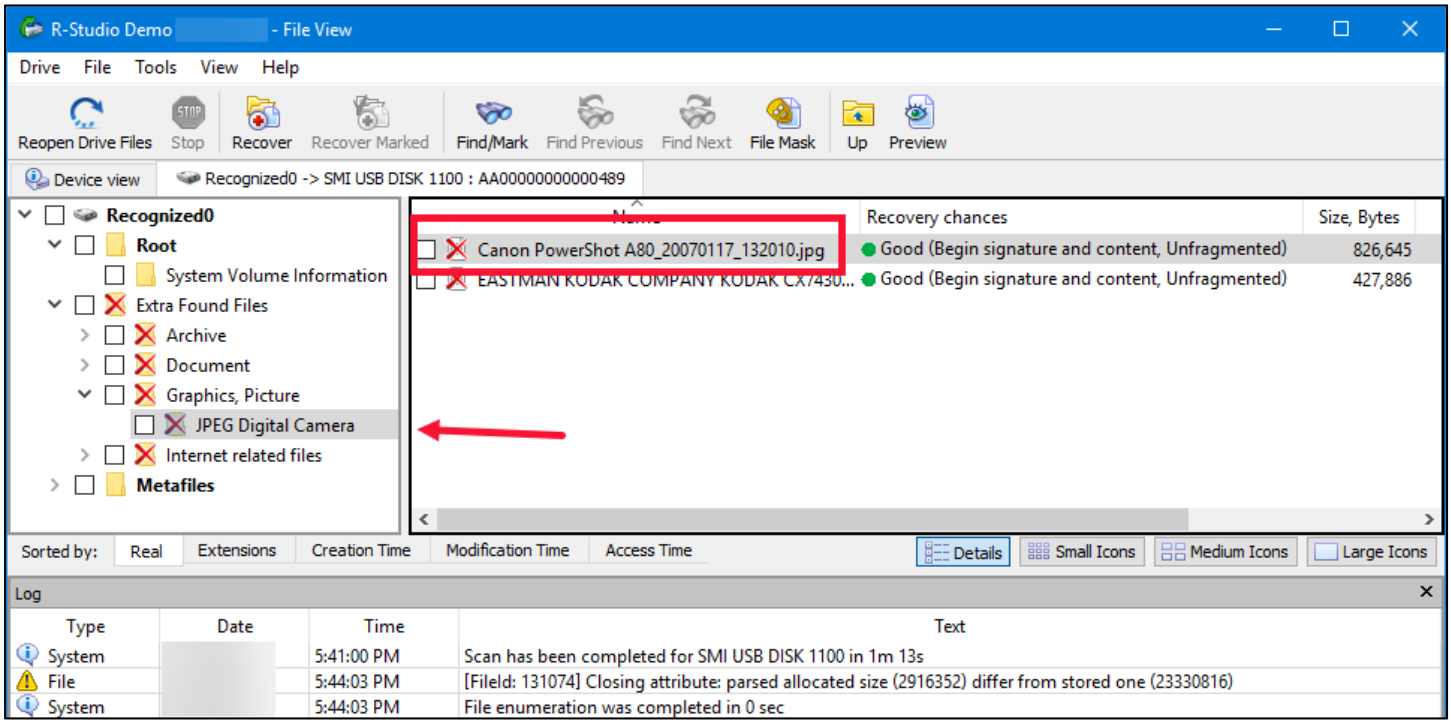
   c. If any opened, why did they open when others didn't?

   _____

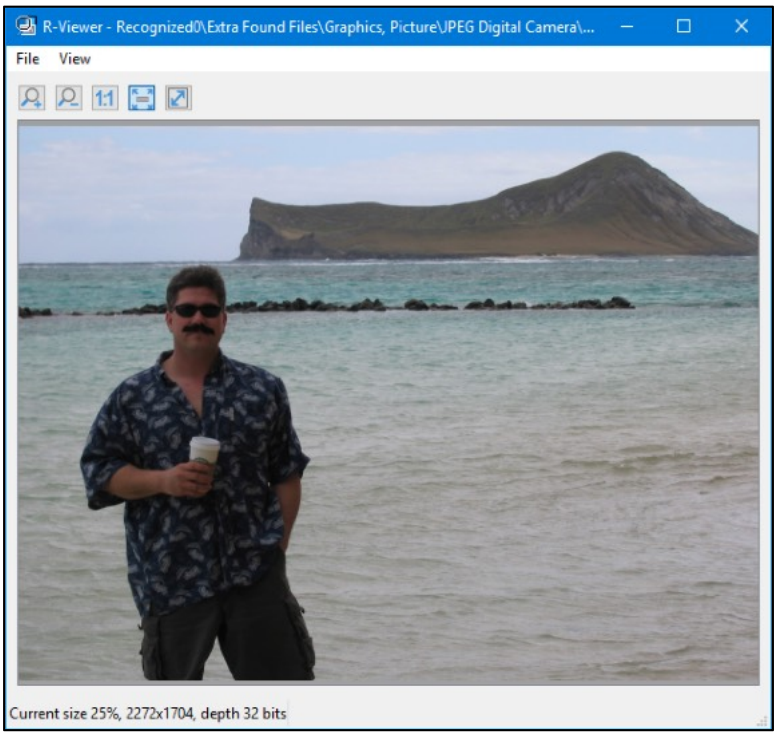
2. You didn't have much luck, even though it initially looked promising. But as we have been learning, just because it looks hopeless, does not mean it is! Note in the entries in the left box, a folder with an **X** across it named **Extra Found Files**. Click on the arrow immediately to the left of the **Extra Found Files** folder to reveal the file types that were found.



3. You will recall that during the scanning phase, you told **R-Studio** to also scan for known file types. Because of this, a file carving function was run. Let's see if it had any success. Look at the different folders that are shown and then expand the **Graphics, Picture** folder and click on the **JPEG Digital Camera** folder to show its contents.



4. Previously you saw in the list of files from the scan screen, there were two .**JPG** files, but they wouldn't open. When we look in the **JPEG Digital Camera** folder, we see two files listed. But they clearly don't have the same names as from the scan screen. Double click on **Canon PowerShot A80_20070117_132010.jpg**.



5. The photo opened and is clearly recoverable, although due to size limitations in the trial version of **R-Studio**, we cannot extract the file.

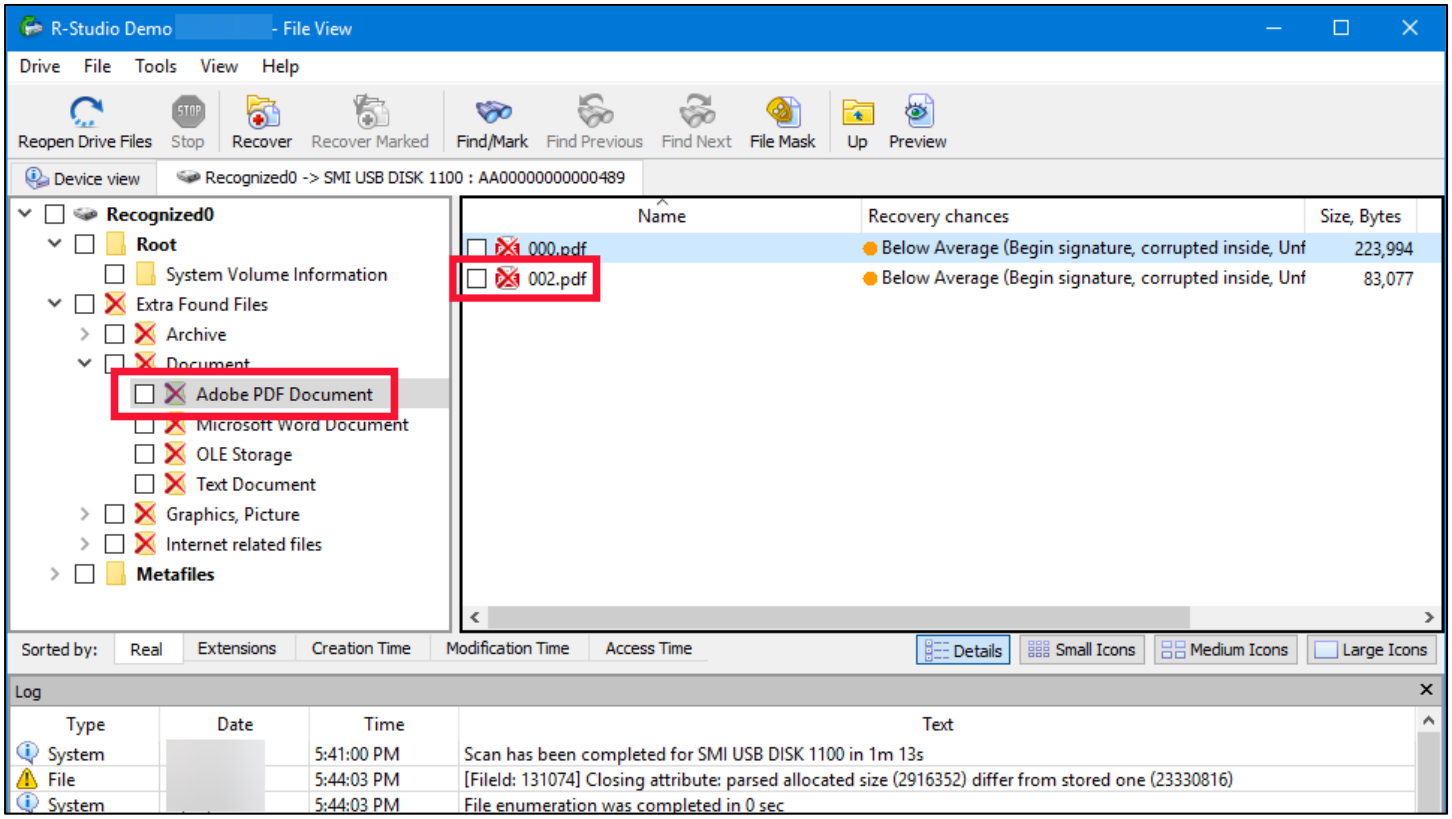
   a. Why were you able to recover the photo when you couldn't before?

   _____

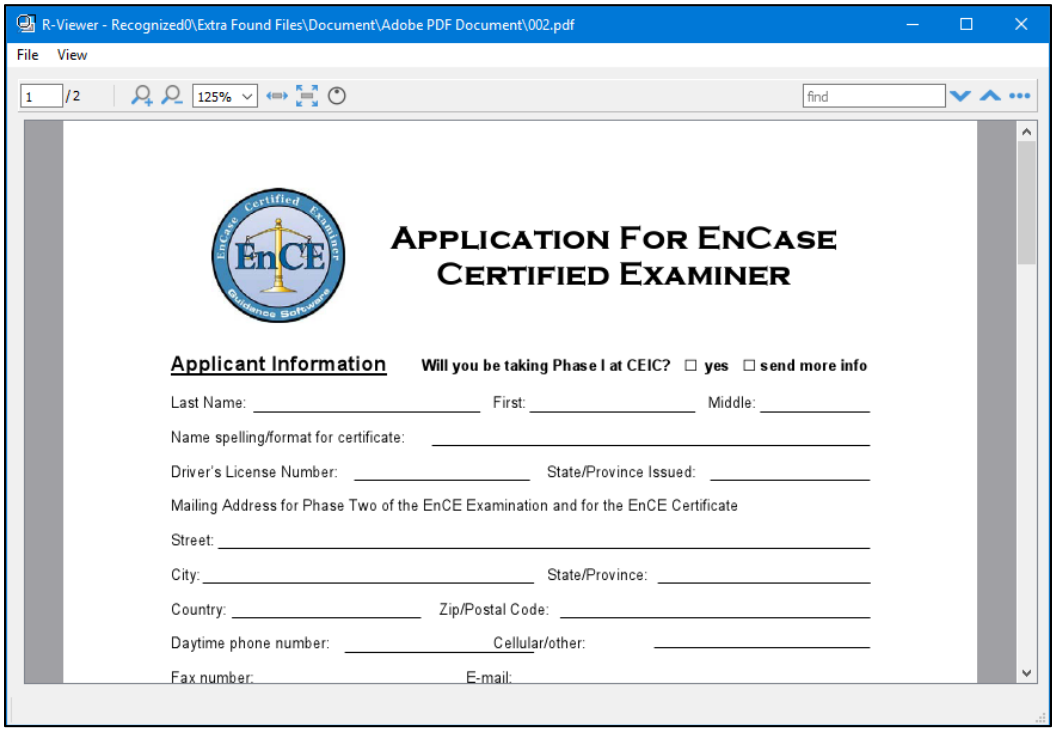
   b. Where does this new file name come from?

   _____

6. Close the photo and return to the **File View** screen. Expand the **Document** folder and click on the **Adobe PDF Document** folder to show its contents.



7. Let's double click on **002.pdf** and see if it opens.

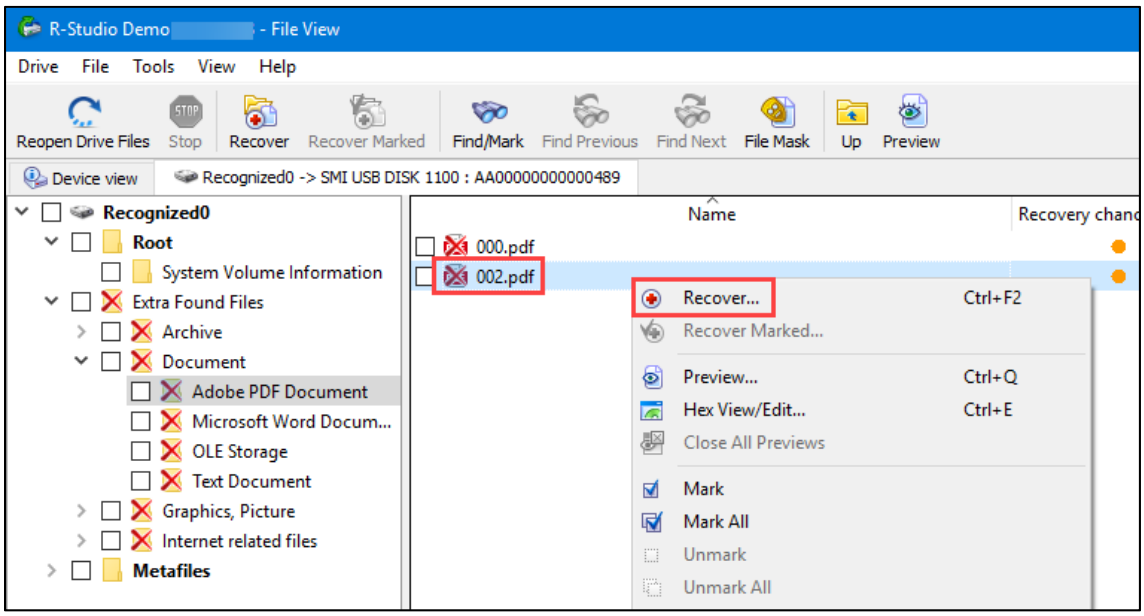


8. Success! However, note that the title of the .**pdf** is not specific and was given a rather generic name of **002.pdf**.

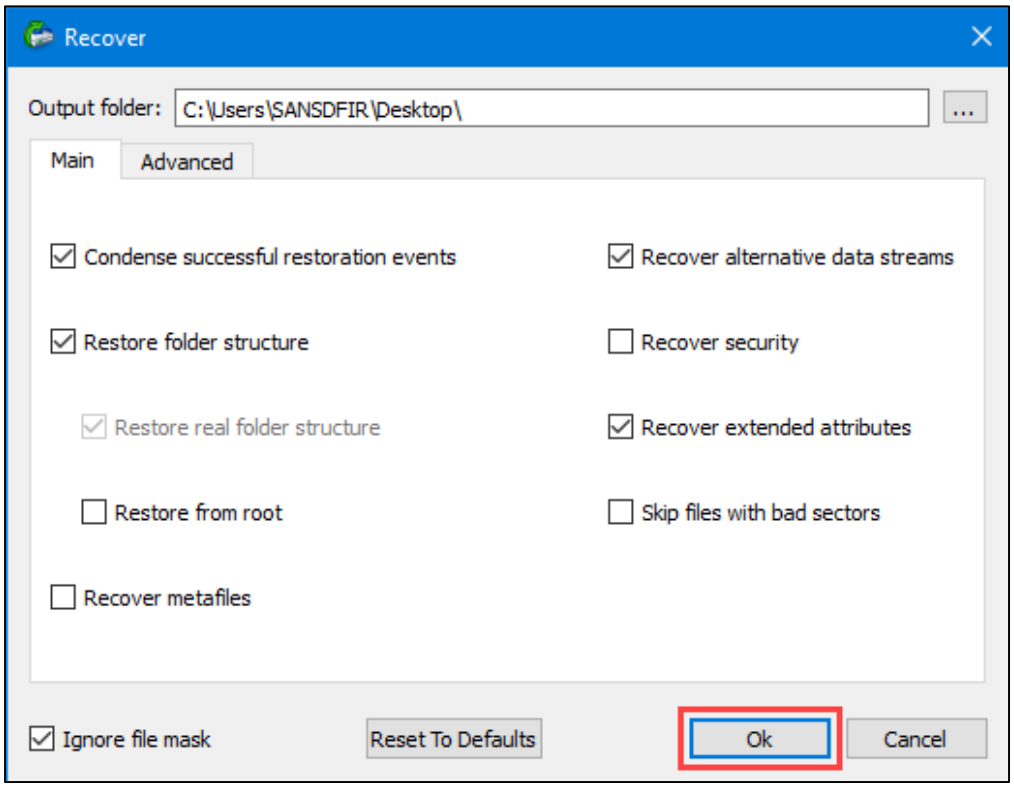
   a. Why does the .**pdf** not have a detailed name like the .**jpg** you previously opened?

   _____

9. Now you will try to recover the file. Right-click on the file and select **Recover…** from the drop-down menu.



10. You will be presented with the **Recover** window. Read over the options to see what they offer, but do not change anything other than the **Output** folder. Set this to your **VM Desktop**, and then click on the **Ok** button.



11. Navigate to your **VM Desktop** and note that the **PDF** is now there.

12. Spend some time navigating through the other folders and files to see what is available. When you are done, you can close the program and any other open windows.

Exercise Questions - Step-by-Step

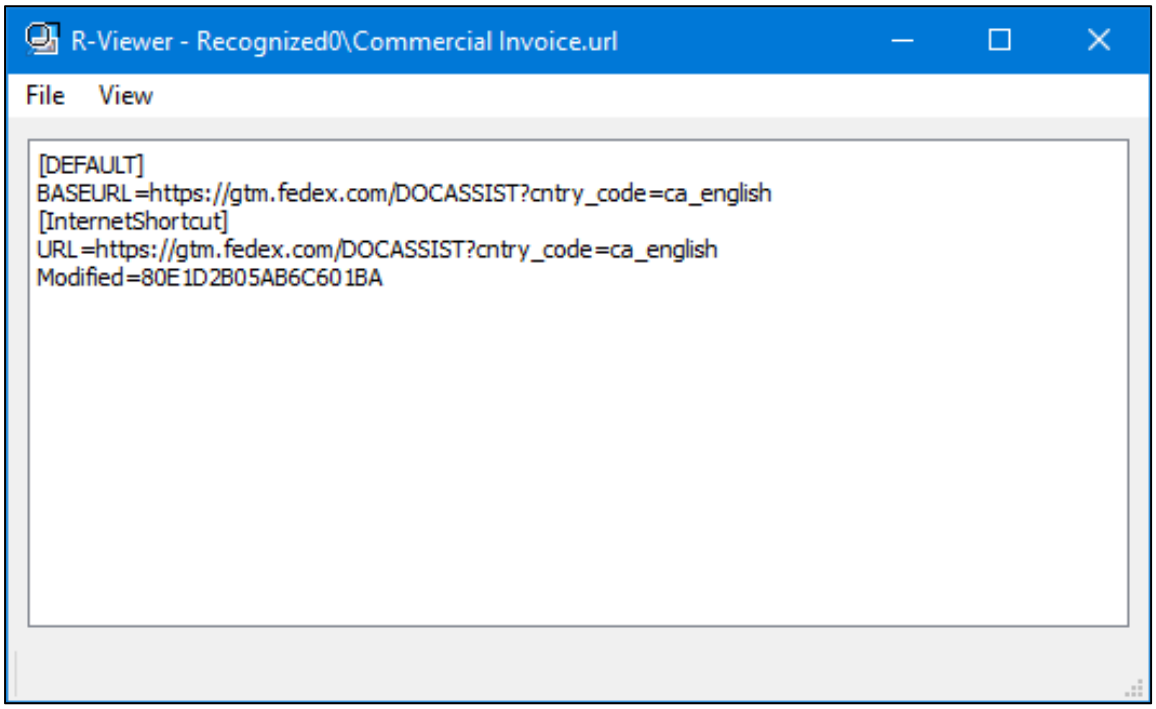
1. Take a moment and double click on all the files that you see in the list.

   a. Did any open?

   YES

   b. If yes, which ones?

   Commercial Invoice.url

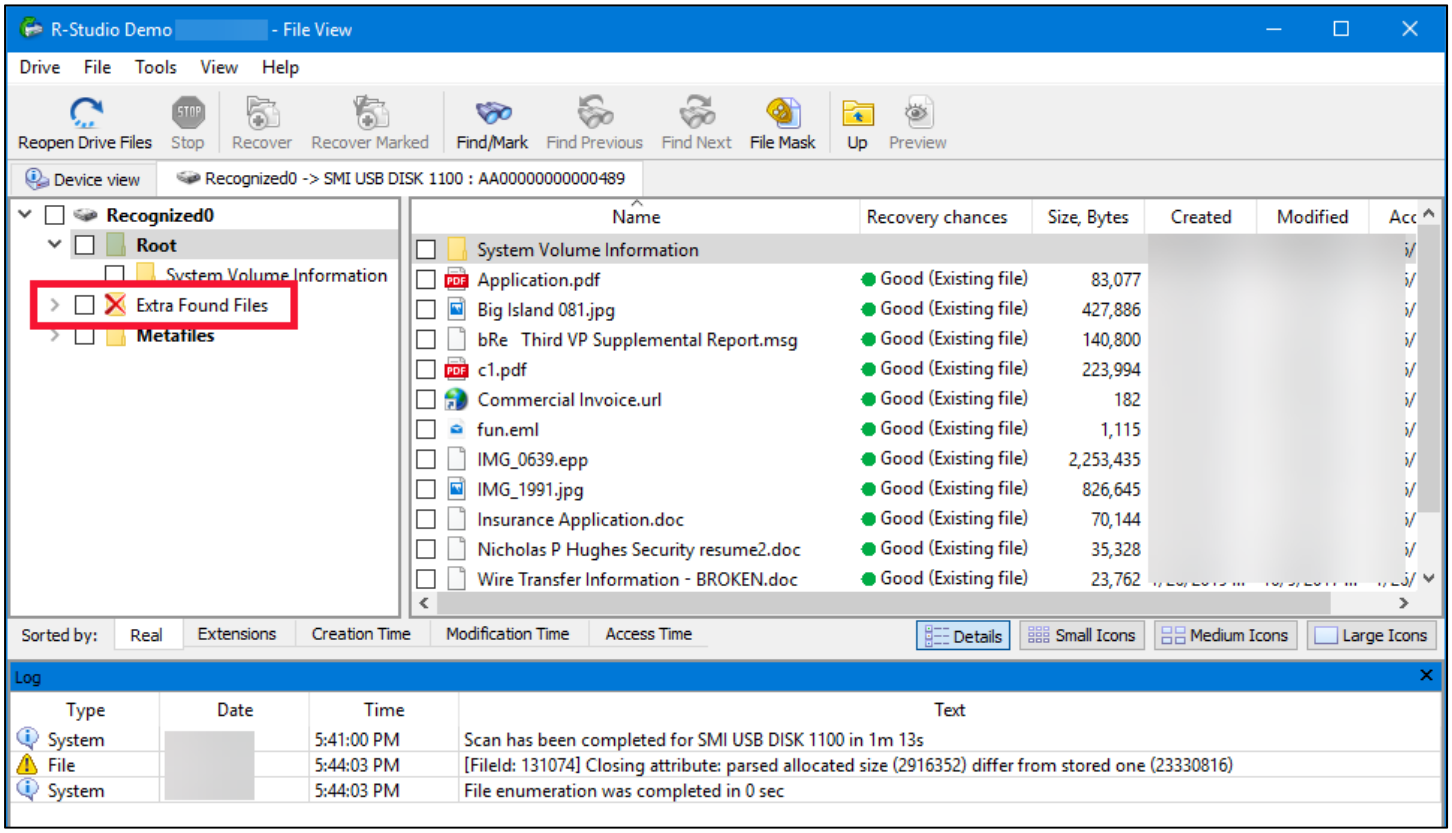


   c. If any opened, why did they open when others didn't?

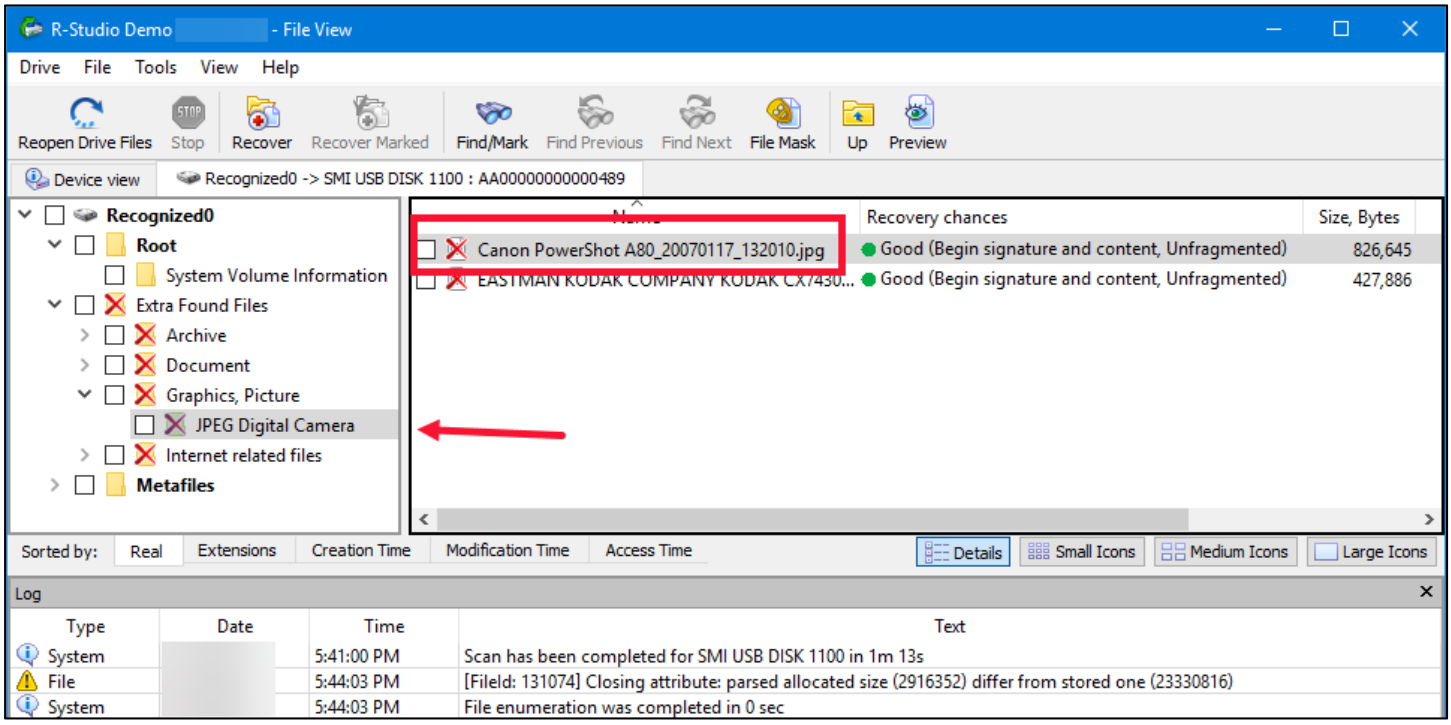
   Because it was actually resident inside an MFT entry, so did not have to go out to the drive to get the file.

   A typical MFT FILE entry is 1024 bytes in size. In most cases, the data in the entry does not use the entire 1024 bytes. If the remaining empty space in an MFT entry is larger than the file's logical size, then the file's data will actually reside inside the MFT entry. Since the data for the file being tracked is stored inside the entry itself, the content of these small files are easily recoverable. This is very common for shortcut files, url files, and small cookie files.

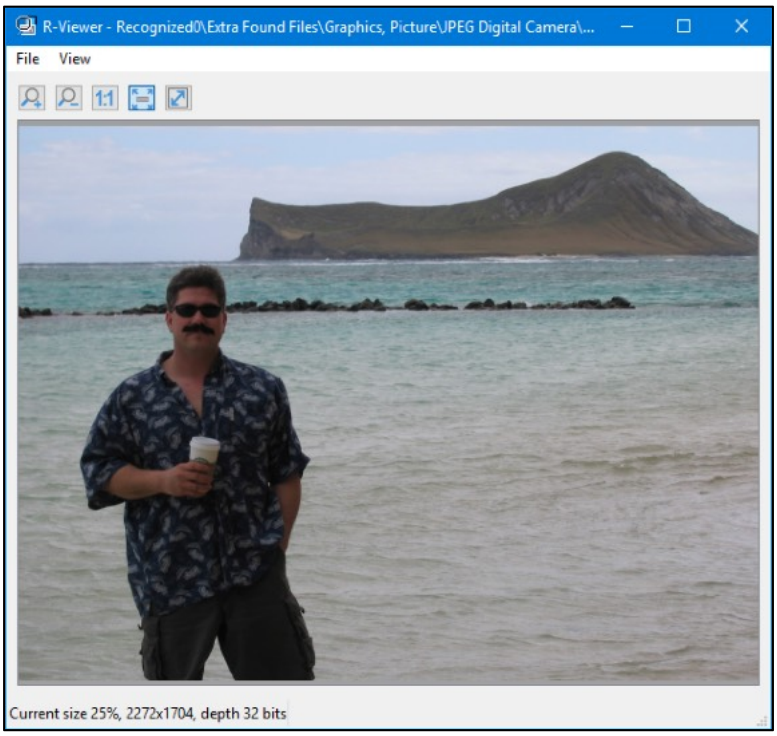
2. You didn't have much luck, even though it initially looked promising. But as we have been learning, just because it looks hopeless, does not mean it is! Note in the entries in the left box, a folder with an **X** across it named **Extra Found Files**. Click on the arrow immediately to the left of the **Extra Found Files** folder to reveal the file types that were found.



3. You will recall that during the scanning phase, you told **R-Studio** to also scan for known file types. Because of this, a file carving function was run. Let's see if it had any success. Look at the different folders that are shown and then expand the **Graphics, Picture** folder and click on the **JPEG Digital Camera** folder to show its contents.



4. Previously you saw in the list of files from the scan screen, there were two .**JPG** files, but they wouldn't open. When we look in the **JPEG Digital Camera** folder, we see two files listed. But they clearly don't have the same names as from the scan screen. Double click on **Canon PowerShot A80_20070117_132010.jpg**.



5. The photo opened and is clearly recoverable, although due to size limitations in the trial version of **R-Studio**, we cannot extract the file.

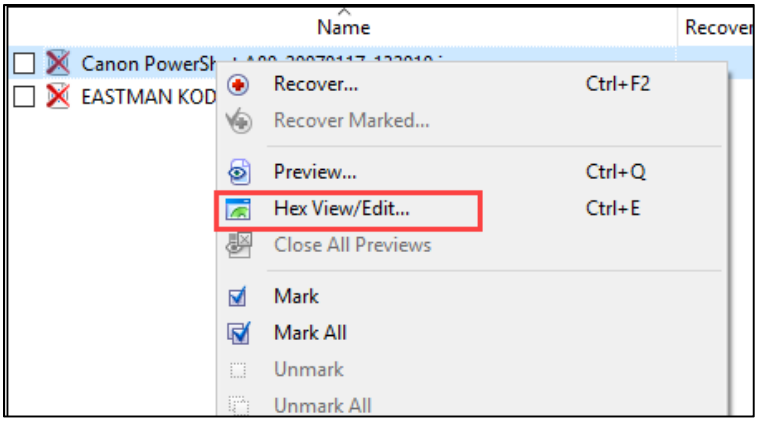
   a. Why were you able to recover the photo when you couldn't before?

   This file was found by file carving, which looks for file signatures, rather than locating a file based on the MFT pointer.

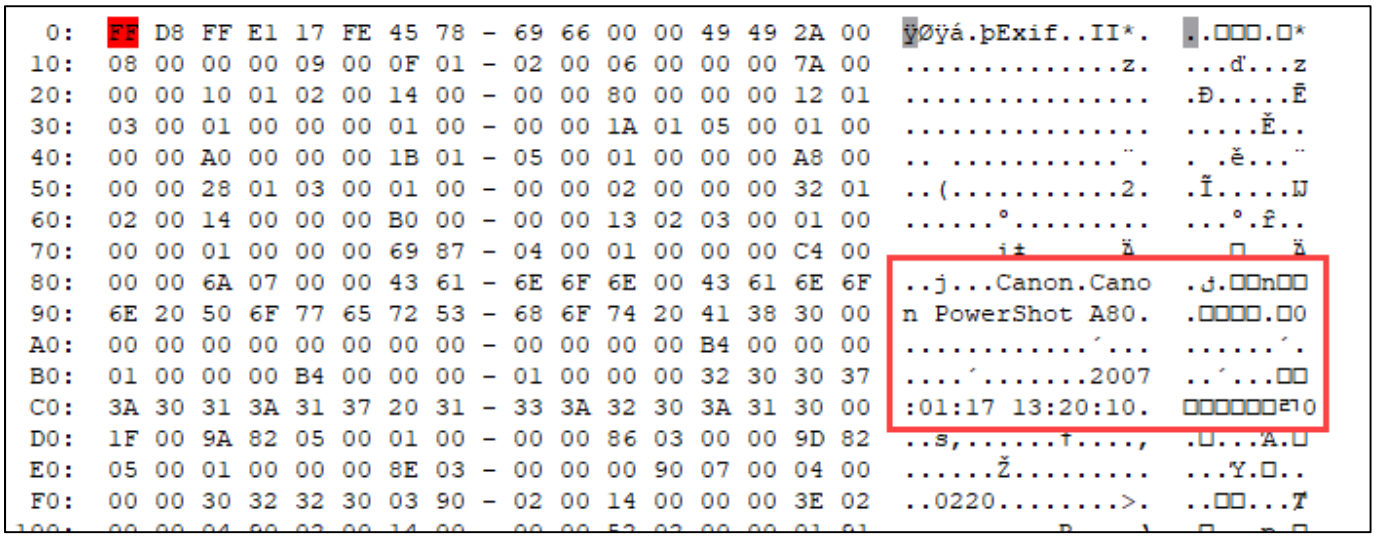
   b. Where does this new file name come from?

   It was extrapolated by R-Studio from data found within the file itself (metadata).

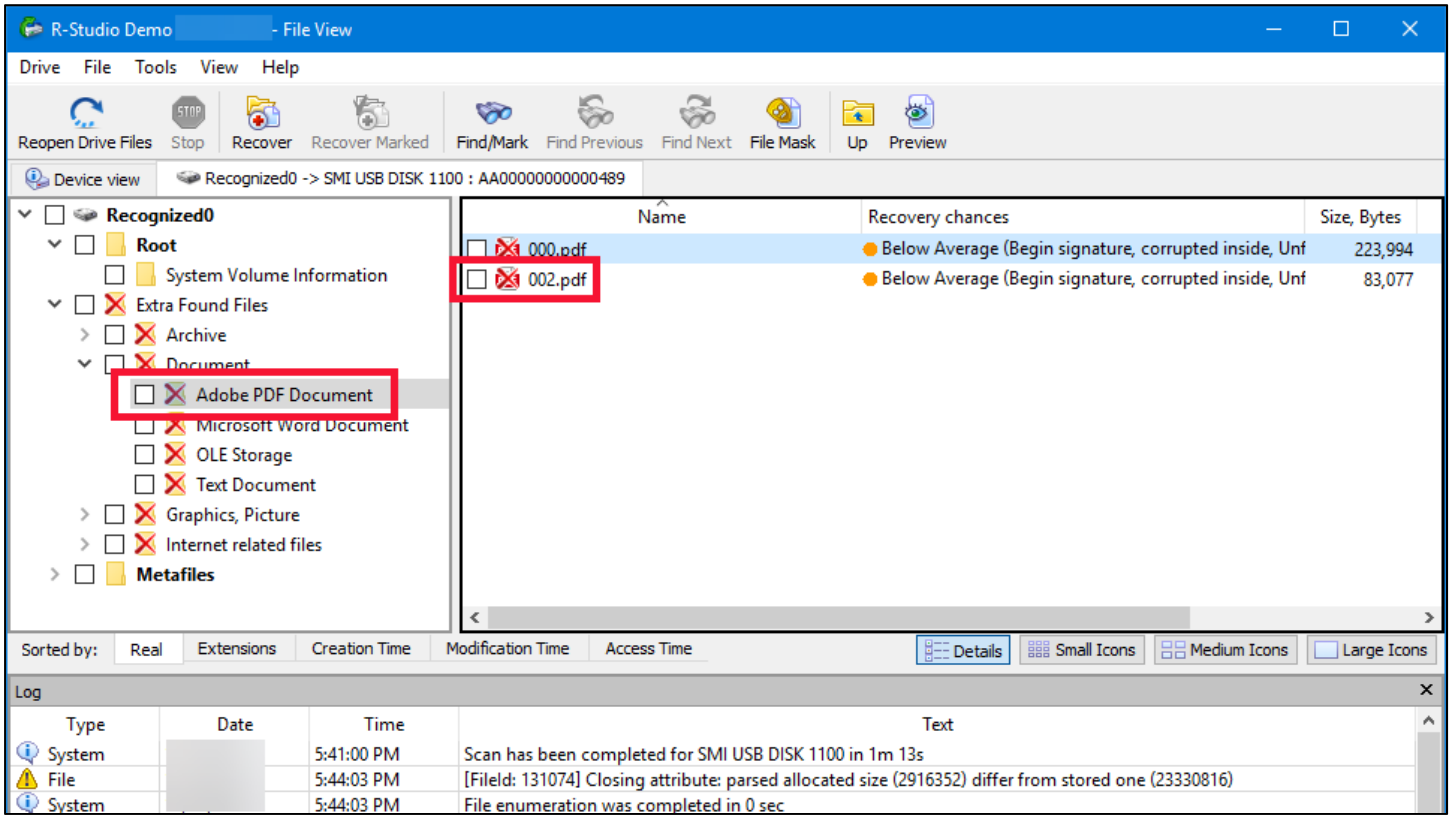
   You can see this by right clicking on the file name in the File View window, and selecting Hex View/Edit…



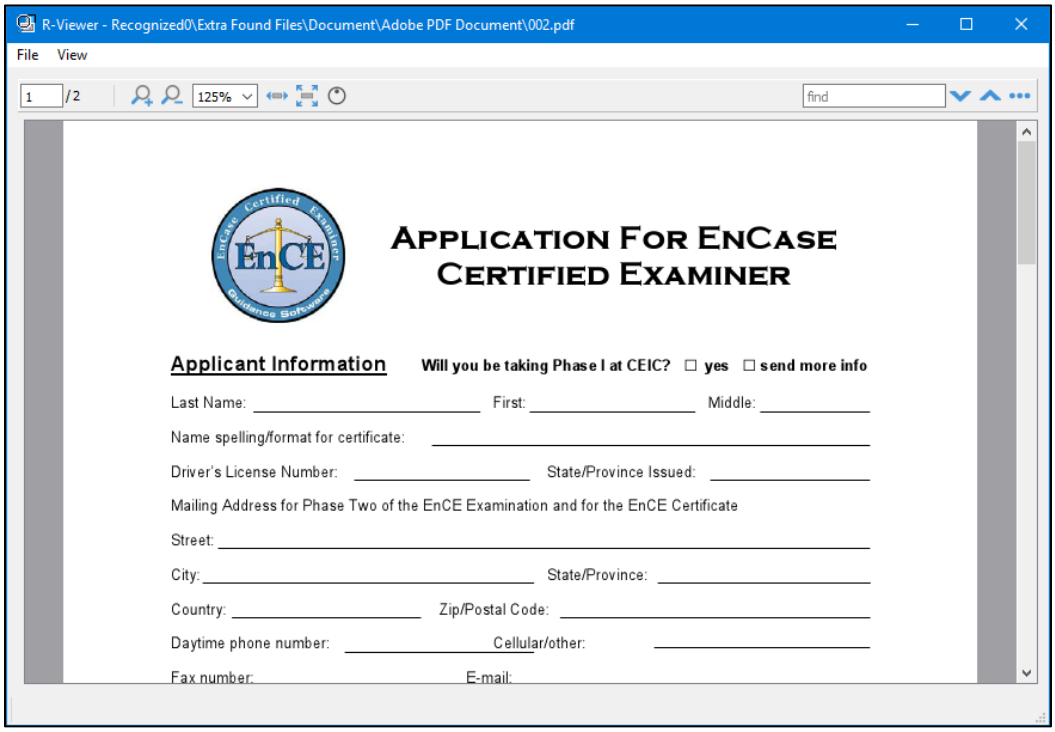
…and then viewing the file in R-Studio's built in hex viewer.



6. Close the photo and return to the **File View** screen. Expand the **Document** folder and click on the **Adobe PDF Document** folder to show its contents.



7. Let's double click on **002.pdf** and see if it opens.



8. Success! However, note that the title of the .**pdf** is not specific and was given a rather generic name of **002.pdf**.

    a. Why does the .**pdf** not have a detailed name like the .**jpg** you previously opened?

    Because there is no plain text metadata in readable areas for R-Studio to parse and/or R-Studio cannot understand the .pdf metadata within the file itself.

Exercise—Key Takeaways

- Data recovery can find data that was otherwise thought to be lost forever.
- Not all data recovery programs are equal in their recovery capabilities.

Exercise 6.3A—Data Carving & Rebuilding – Images and .doc

Background

Data rebuilding is a skill that has always been important, but increasingly so today. Once the reference to a file is destroyed, how can the data still be recovered? File carving tools will assist in this, but the examiner must understand the limitations of their tools. Without the proper pieces of the original file, a carver is useless. This is not a fast operation, and there are few quick wins, but this approach is much more surgical than simply scraping for files. The difference too is that if the data you seek exists, you will find it. In the traditional manner of carving, you could spend days or weeks going through carved data, and never find what you needed.

If the stakes are high enough, there are other methods at our disposal. How about data that has been partially overwritten by new data? No carving tool can touch this. Data rebuilding is not for people who dabble. It takes a considerable amount of patience, self-discipline, and expertise to master. It is not the type of skill that gets learned in the trenches. You must start honing this skill long before you ever need it in the real world. Either you know what you are looking for, or you know where to look to find what you are looking for. No "9-5" here. We are going into overtime!

Exercise Objectives

- Recognize various common file signatures
- Carve manually for a deleted .JPG file
- Understand OLECF and OpenXML Office formats
- Carve manually for a deleted document file

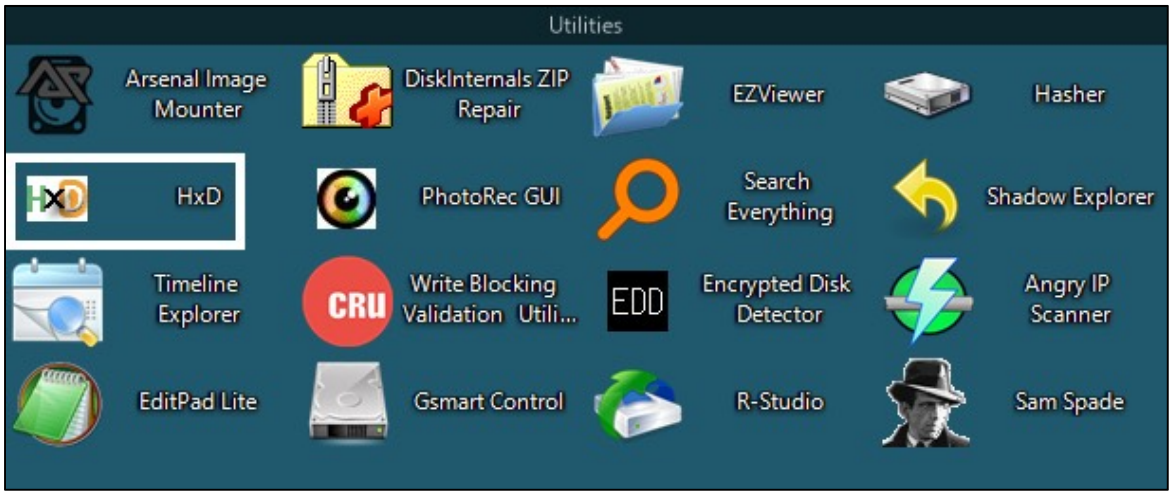
Exercise Preparation

1. Boot your **FOR498 Windows VM**

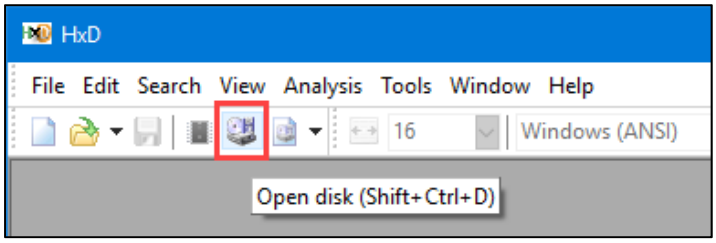
2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Using the techniques you learned in the image mounting exercise (**Exercise 3.2**), mount the **CARVEY.E01** found at **C:\Cases\Exercises\6.3** using **Arsenal Image Mounter**.

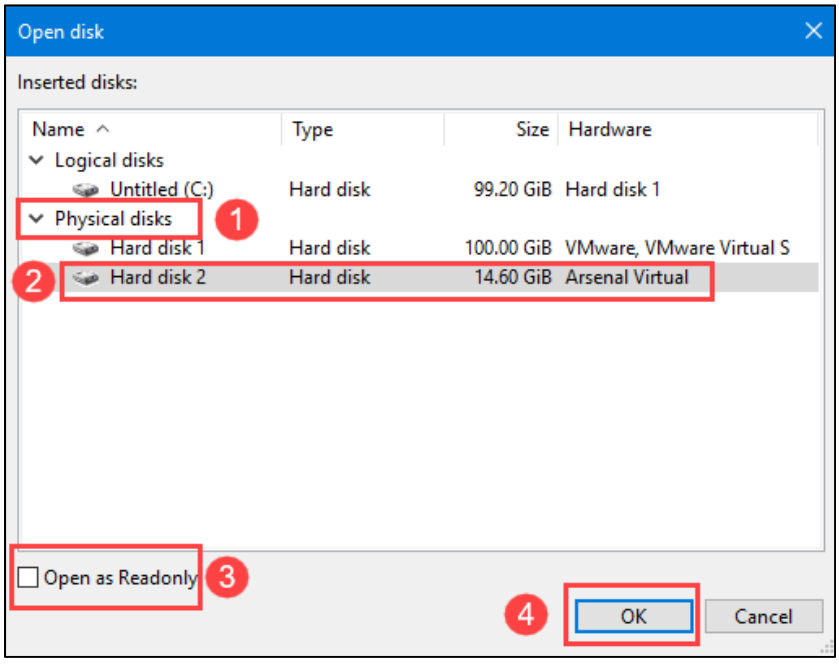
1. Once the drive is mounted, open **HxD** Hex Editor using the shortcut in the **Utilities** fence on the **Desktop**.



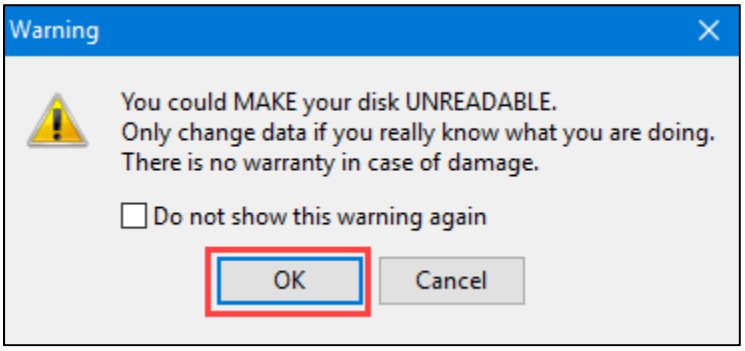
2. Click on the **Open disk** icon to reveal the **Physical disk** mounting options.



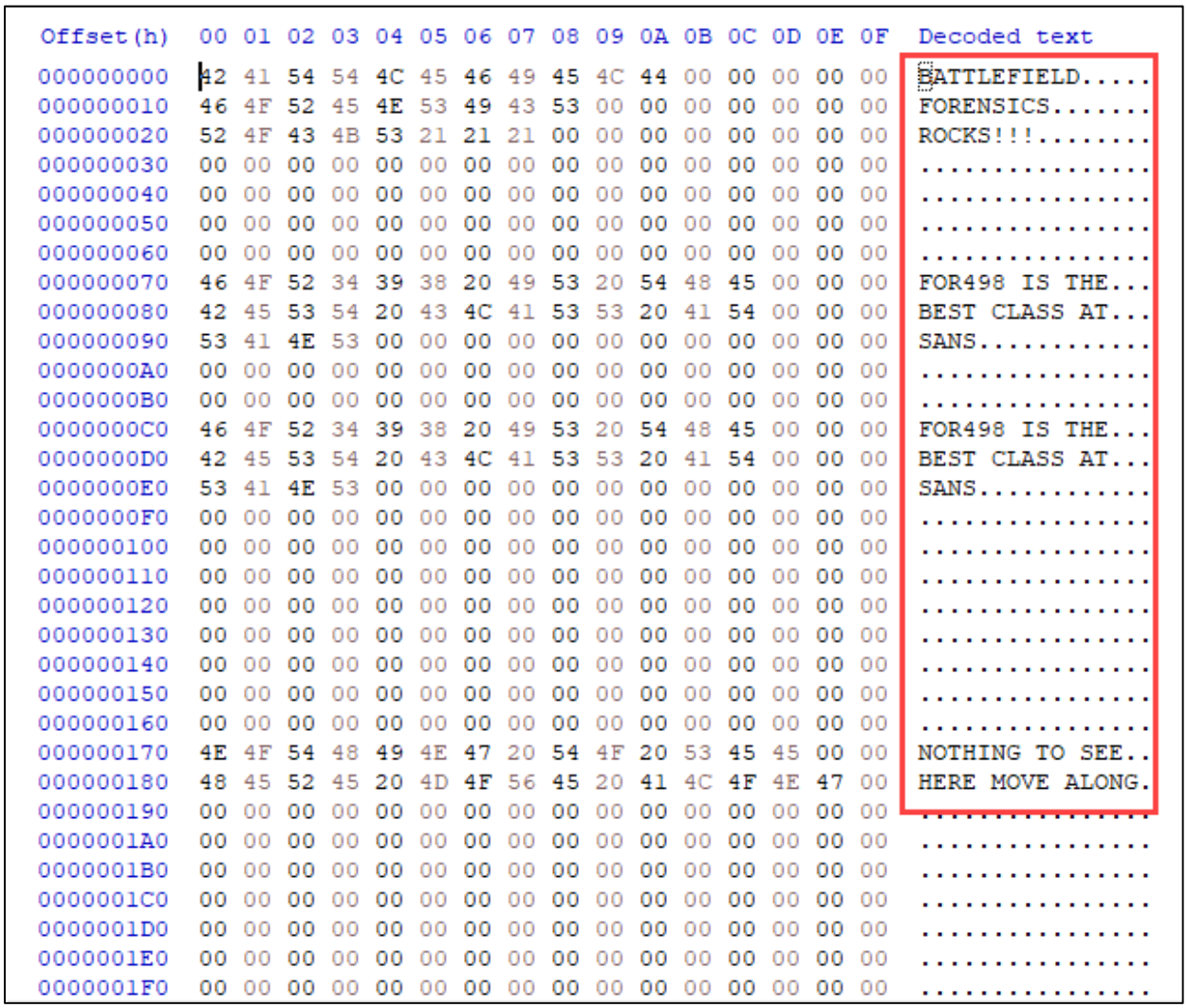
3. You will see both **Logical disks** and **Physical disks** listed. The **CARVEY.E01** image will not have a drive letter assigned. Select **Physical disks**, then select the **Hard disk** for which the **Hardware** name is **Arsenal Virtual**. Uncheck the **Open as Readonly** box and click **OK**.



4. You will receive a warning box. Read this carefully. Since you are working with a virtual drive, this is not so important. Click **OK**.

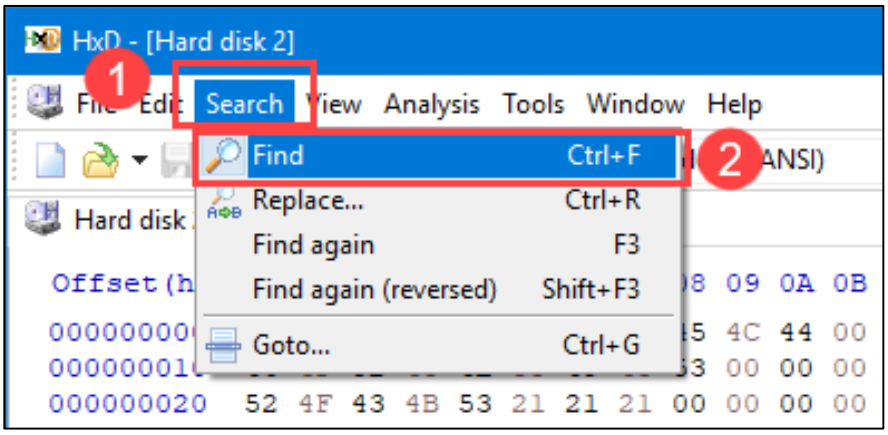


5. The first sector of the physical drive will appear. **WARNING!** If your screen does not show the same data as below, **STOP.** Chances are that you mounted the wrong drive and could destroy another volume on your system by continuing. If you see something different, close everything and start over. Your starting point for every carve in this exercise will be from zero sector. In other words, before starting a new carve or search, ensure you are starting from the very beginning of the drive, and can see the below.

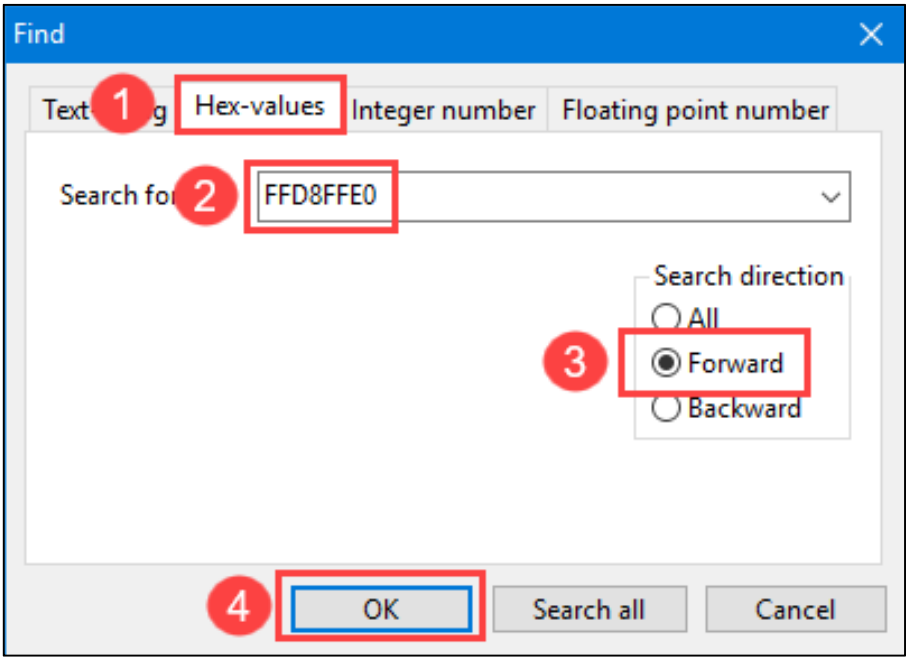


Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
000000000	42	41	54	54	4C	45	46	49	45	4C	44	00	00	00	00	00	BATTLEFIELD.....
000000010	46	4F	52	45	4E	53	49	43	53	00	00	00	00	00	00	00	FORENSICS.....
000000020	52	4F	43	4B	53	21	21	21	00	00	00	00	00	00	00	00	ROCKS!!!.....
000000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000070	46	4F	52	34	39	38	20	49	53	20	54	48	45	00	00	00	FOR498 IS THE...
000000080	42	45	53	54	20	43	4C	41	53	53	20	41	54	00	00	00	BEST CLASS AT...
000000090	53	41	4E	53	00	00	00	00	00	00	00	00	00	00	00	00	SANS.....
0000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000C0	46	4F	52	34	39	38	20	49	53	20	54	48	45	00	00	00	FOR498 IS THE...
0000000D0	42	45	53	54	20	43	4C	41	53	53	20	41	54	00	00	00	BEST CLASS AT...
0000000E0	53	41	4E	53	00	00	00	00	00	00	00	00	00	00	00	00	SANS.....
0000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000170	4E	4F	54	48	49	4E	47	20	54	4F	20	53	45	45	00	00	NOTHING TO SEE..
000000180	48	45	52	45	20	4D	4F	56	45	20	41	4C	4F	4E	47	00	HERE MOVE ALONG.
000000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

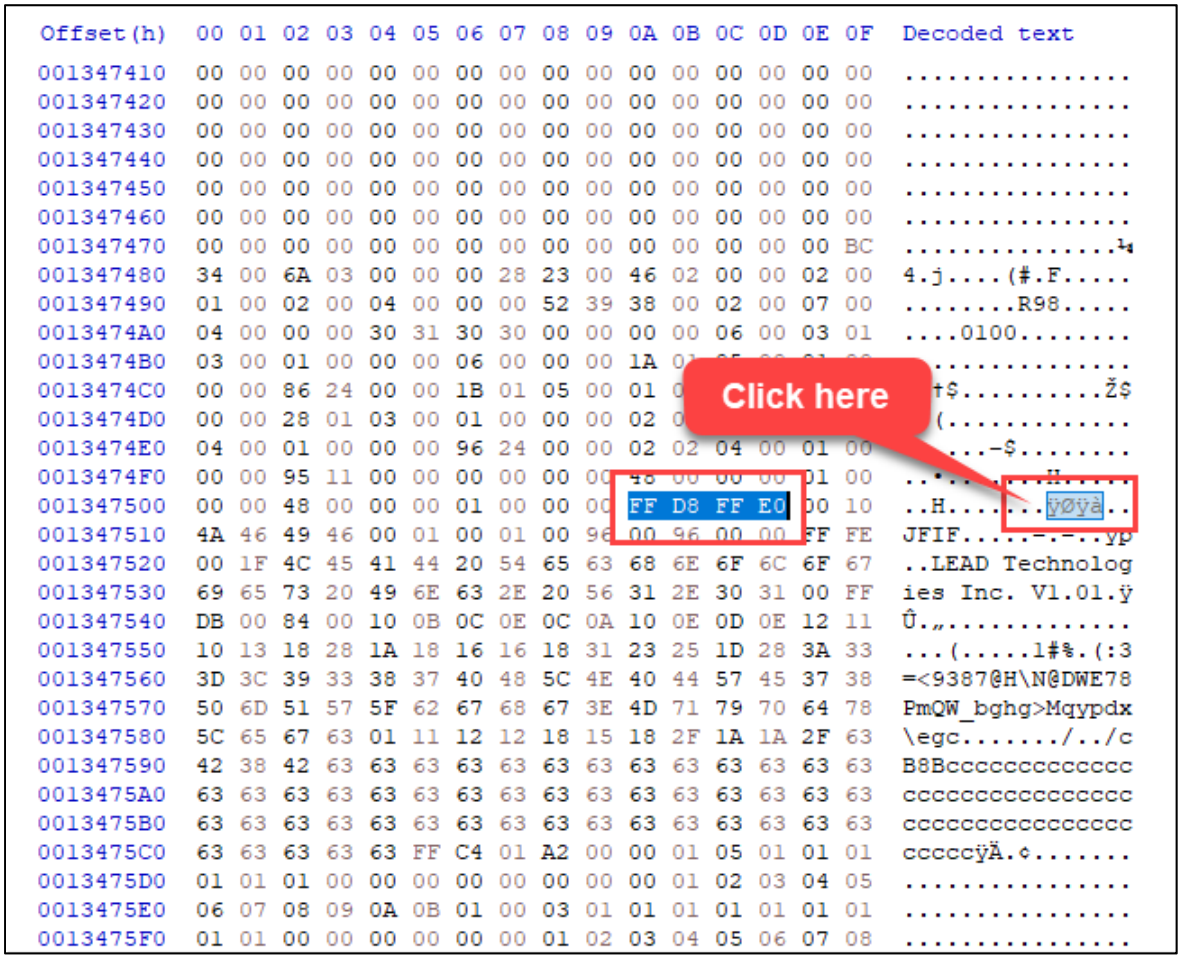
6. Let's start with an easy carve. We will carve a photo. It will be completely intact, and so normally would be picked up by a carver, but we must walk before we run. Click on **Search** and **Find**.



7. In the **Find** box, ensure that the **Hex-values** tab is selected, and in the **Search** for field, type **`FFD8FFE0`**. Set **Search direction** to **Forward** and click **OK**.

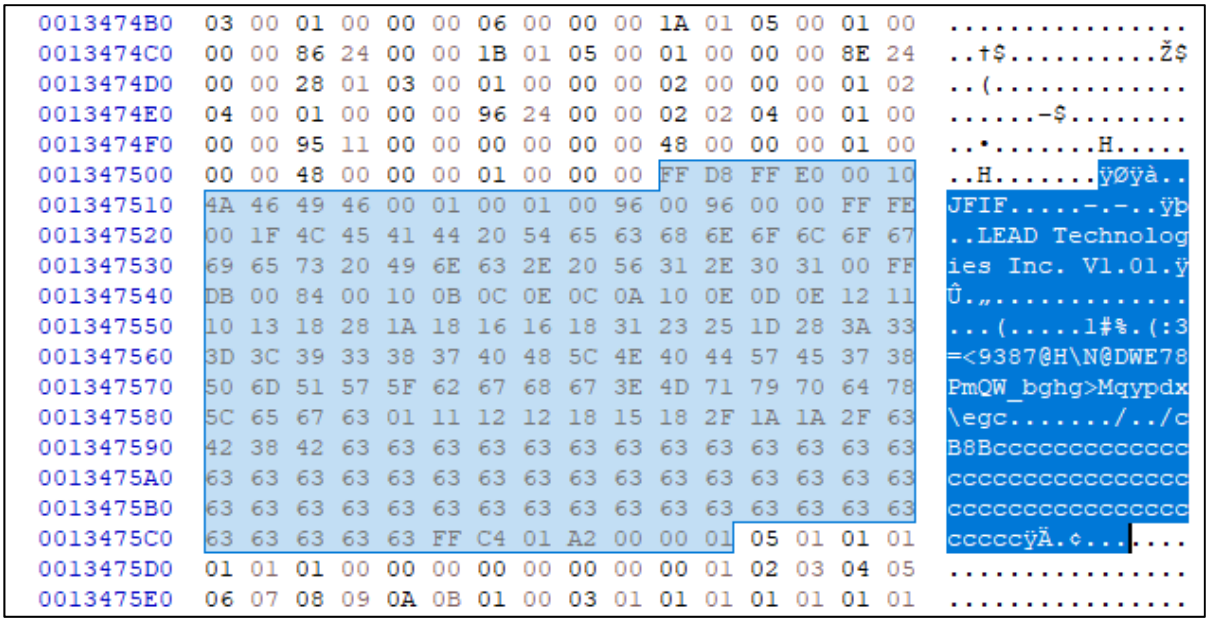


8. You will be taken to a location on the drive that appears as below. Place your cursor immediately in front of the first **ÿ** in the **Decoded text** column.



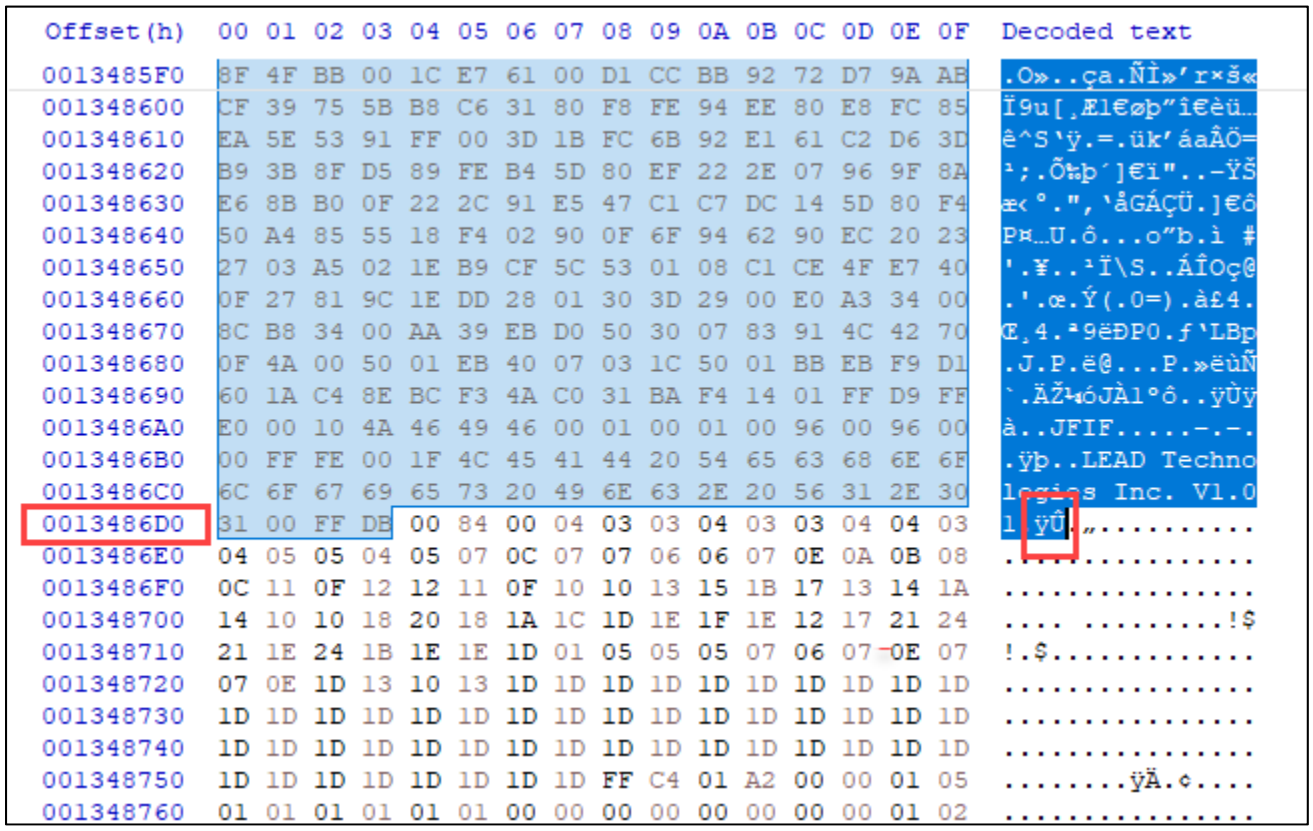
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
001347410	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001347420	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001347430	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001347440	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001347450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001347460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001347470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	BC	.....¼
001347480	34	00	6A	03	00	00	00	28	23	00	46	02	00	00	02	00	4.j....(#.F.....
001347490	01	00	02	00	04	00	00	00	52	39	38	00	02	00	07	00	........R98.....
0013474A0	04	00	00	00	30	31	30	30	00	00	00	00	06	00	03	01	....0100........
0013474B0	03	00	01	00	00	00	06	00	00	00	1A	01	05	00	01	00	.....
0013474C0	00	00	86	24	00	00	1B	01	05	00	01	0					†$.....Ž$
0013474D0	00	00	28	01	03	00	01	00	00	00	02	0					(.....
0013474E0	04	00	01	00	00	00	96	24	00	00	02	02	04	00	01	00	....–$.....
0013474F0	00	00	95	11	00	00	00	00	00	00	48	00	00	00	01	00	..•.....H.....
001347500	00	00	48	00	00	00	01	00	00	00	FF	D8	FF	E0	00	10	..H.....ÿØÿà..
001347510	4A	46	49	46	00	01	00	01	00	96	00	96	00	00	FF	FE	JFIF.....–.–..ÿþ
001347520	00	1F	4C	45	41	44	20	54	65	63	68	6E	6F	6C	6F	67	..LEAD Technolog
001347530	69	65	73	20	49	6E	63	2E	20	56	31	2E	30	31	00	FF	ies Inc. V1.01.ÿ
001347540	DB	00	84	00	10	0B	0C	0E	0C	0A	10	0E	0D	0E	12	11	Û.„.....
001347550	10	13	18	28	1A	18	16	16	18	31	23	25	1D	28	3A	33	...(.....1#%.(:3
001347560	3D	3C	39	33	38	37	40	48	5C	4E	40	44	57	45	37	38	=<9387@H\N@DWE78
001347570	50	6D	51	57	5F	62	67	68	67	3E	4D	71	79	70	64	78	PmQW_bghg>Mqypdx
001347580	5C	65	67	63	01	11	12	12	18	15	18	2F	1A	1A	2F	63	\egc...../../c
001347590	42	38	42	63	63	63	63	63	63	63	63	63	63	63	63	63	B8Bcccccccccccc
0013475A0	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	cccccccccccccc
0013475B0	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	cccccccccccccc
0013475C0	63	63	63	63	63	FF	C4	01	A2	00	00	01	05	01	01	01	cccccÿÄ.¢.....
0013475D0	01	01	01	00	00	00	00	00	00	00	00	01	02	03	04	05	.....
0013475E0	06	07	08	09	0A	0B	01	00	03	01	01	01	01	01	01	01	.....
0013475F0	01	01	00	00	00	00	00	00	01	02	03	04	05	06	07	08	.....

9. Press and hold the **Shift** key and press the **DOWN** arrow key a few times. You will see content starting to be highlighted. Ensure that the highlighting starts exactly at the **ÿ** and not one byte before or after.



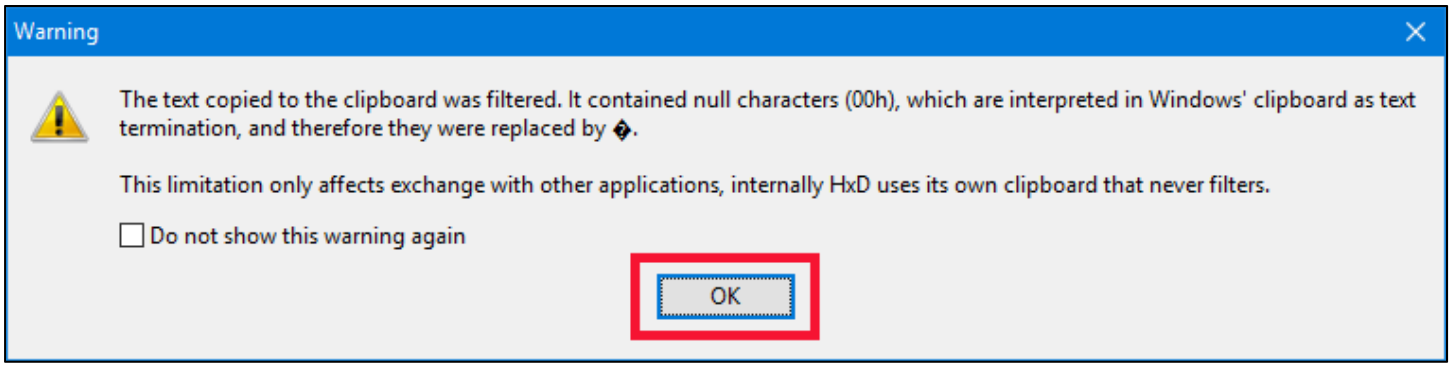
0013474B0	03	00	01	00	00	00	06	00	00	00	1A	01	05	00	01	00	.....
0013474C0	00	00	86	24	00	00	1B	01	05	00	01	00	00	00	8E	24	..†$.....Ž$
0013474D0	00	00	28	01	03	00	01	00	00	00	02	00	00	00	01	02	..(.....
0013474E0	04	00	01	00	00	00	96	24	00	00	02	02	04	00	01	00	.....–$.....
0013474F0	00	00	95	11	00	00	00	00	00	00	48	00	00	00	01	00	..•.....H.....
001347500	00	00	48	00	00	00	01	00	00	00	FF	D8	FF	E0	00	10	..H.....ÿØÿà..
001347510	4A	46	49	46	00	01	00	01	00	96	00	96	00	00	FF	FE	JFIF.....–.–..ÿþ
001347520	00	1F	4C	45	41	44	20	54	65	63	68	6E	6F	6C	6F	67	..LEAD Technolog
001347530	69	65	73	20	49	6E	63	2E	20	56	31	2E	30	31	00	FF	ies Inc. V1.01.ÿ
001347540	DB	00	84	00	10	0B	0C	0E	0C	0A	10	0E	0D	0E	12	11	Û.„.....
001347550	10	13	18	28	1A	18	16	16	18	31	23	25	1D	28	3A	33	...(.....1#%.(:3
001347560	3D	3C	39	33	38	37	40	48	5C	4E	40	44	57	45	37	38	=<9387@H\N@DWE78
001347570	50	6D	51	57	5F	62	67	68	67	3E	4D	71	79	70	64	78	PmQW_bghg>Mqypdx
001347580	5C	65	67	63	01	11	12	12	18	15	18	2F	1A	1A	2F	63	\egc...../../c
001347590	42	38	42	63	63	63	63	63	63	63	63	63	63	63	63	63	B8Bcccccccccccc
0013475A0	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	cccccccccccccc
0013475B0	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	cccccccccccccc
0013475C0	63	63	63	63	63	FF	C4	01	A2	00	00	01	05	01	01	01	cccccÿÄ.¢.....
0013475D0	01	01	01	00	00	00	00	00	00	00	00	01	02	03	04	05	.....
0013475E0	06	07	08	09	0A	0B	01	00	03	01	01	01	01	01	01	01	.....

10. Continue "sweeping" in this manner until you find the end of the file, indicated by **ÿÛ** and some open space following it. If you were watching closely, you may have seen this file footer earlier on, but it was surrounded by other data, so probably not the one we wanted. Having said that, there would be nothing wrong with stopping at the first one, carving the data out, and finding that you had nothing. You will already see that the dataset was so small that it was probably nothing. Just dive back in and collect more data until you find the next **ÿÛ** and try it again. Data carving is all about trial and error. Practice makes you better (and faster) over time.

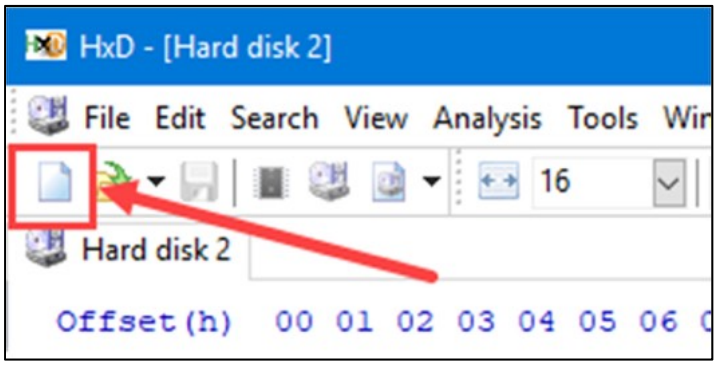


Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0013485F0	8F	4F	BB	00	1C	E7	61	00	D1	CC	BB	92	72	D7	9A	AB	.O»..ça.ÑÌ»'r×š«
001348600	CF	39	75	5B	B8	C6	31	80	F8	FE	94	EE	80	E8	FC	85	Ï9u[¸Æ1€øþ"î€èü…
001348610	EA	5E	53	91	FF	00	3D	1B	FC	6B	92	E1	61	C2	D6	3D	ê^S'ÿ.=.ük'áaÂÖ=
001348620	B9	3B	8F	D5	89	FE	B4	5D	80	EF	22	2E	07	96	9F	8A	¹;.Õ‰þ´]€ï".–Ÿš
001348630	E6	8B	B0	0F	22	2C	91	E5	47	C1	C7	DC	14	5D	80	F4	æ‹°.",'åGÁÇÜ.]€ô
001348640	50	A4	85	55	18	F4	02	90	0F	6F	94	62	90	EC	20	23	P¤…U.ô...o"b.ì #
001348650	27	03	A5	02	1E	B9	CF	5C	53	01	08	C1	CE	4F	E7	40	'.¥..¹Ï\S..ÁÎOç@
001348660	0F	27	81	9C	1E	DD	28	01	30	3D	29	00	E0	A3	34	00	.'.œ.Ý(.0=).à£4.
001348670	8C	B8	34	00	AA	39	EB	D0	50	30	07	83	91	4C	42	70	Œ¸4.ª9ëÐP0.ƒ'LBp
001348680	0F	4A	00	50	01	EB	40	07	03	1C	50	01	BB	EB	F9	D1	.J.P.ë@...P.»ëùÑ
001348690	60	1A	C4	8E	BC	F3	4A	C0	31	BA	F4	14	01	FF	D9	FF	`.ÄŽ¼óJÀ1ºô..ÿÙÿ
0013486A0	E0	00	10	4A	46	49	46	00	01	00	01	00	96	00	96	00	à..JFIF.....–.–.
0013486B0	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	68	6E	6F	.ÿþ..LEAD Techno
0013486C0	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	31	2E	30	logies Inc. V1.0
0013486D0	31	00	FF	DB	00	84	00	04	03	03	04	03	03	04	04	03	1.ÿÛ.„.....
0013486E0	04	05	05	04	05	07	0C	07	07	06	06	07	0E	0A	0B	08	.....
0013486F0	0C	11	0F	12	12	11	0F	10	10	13	15	1B	17	13	14	1A	.....
001348700	14	10	10	18	20	18	1A	1C	1D	1E	1F	1E	12	17	21	24	....!$
001348710	21	1E	24	1B	1E	1E	1D	01	05	05	05	07	06	07	0E	07	!.$.....
001348720	07	0E	1D	13	10	13	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	.....
001348730	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	.....
001348740	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	.....
001348750	1D	1D	1D	1D	1D	1D	1D	1D	FF	C4	01	A2	00	00	01	05	.....ÿÄ.¢....
001348760	01	01	01	01	01	01	00	00	00	00	00	00	00	00	01	02	.....

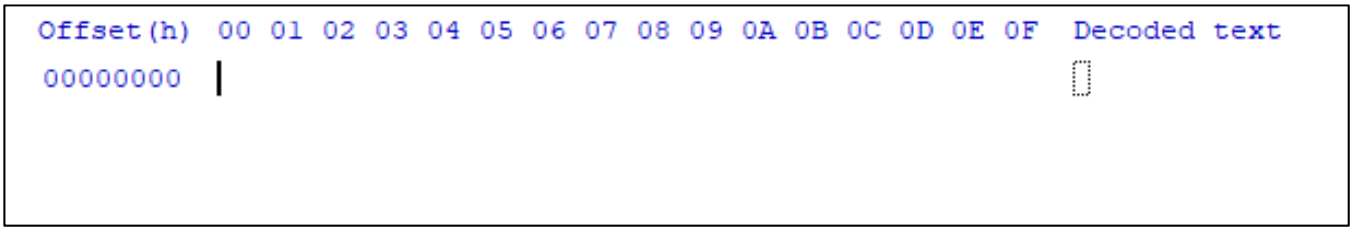
11. Once you have highlighted the content that you want, you can either right-click and **copy**, or you can **Ctrl+C**. We recommend **Ctrl+C** simply because right-clicking on the highlighted data, if not done perfectly, will act like a single click and all of your highlighting will be gone, which means you must start over. Not a big deal here, but if you have swept for a megabyte of data, you will be quite unhappy to lose it! After you have performed the copy, you will get a warning message. Click **OK**.



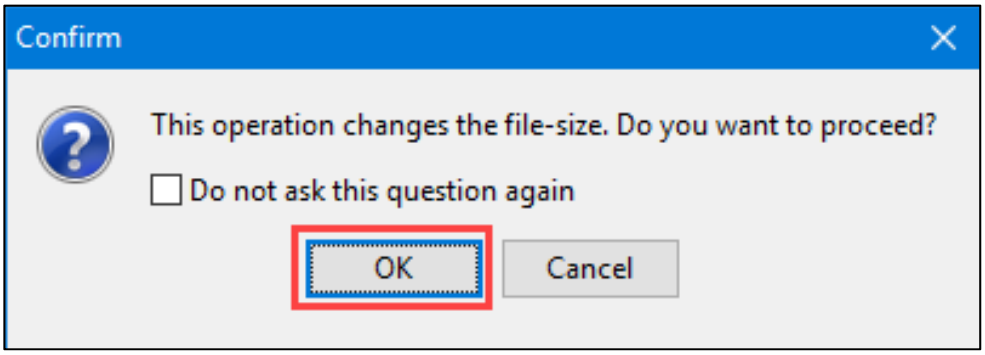
12. At the top left corner of the **HxD** program window, click the white page icon to create a new data space. You could also select **File → New.**



13. A new data space will open and will be blank.



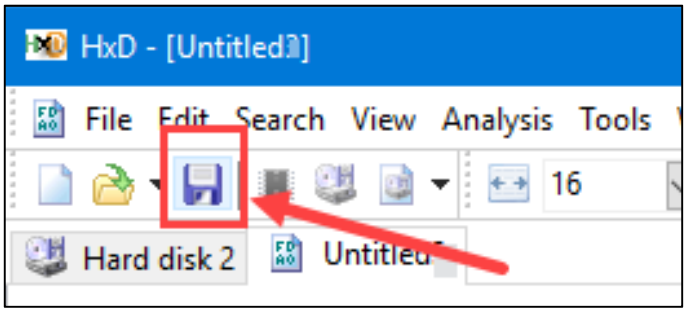
14. Now **Paste** the content you previously copied, by pressing **Ctrl+V**. You will first get a warning message. Click **OK**.



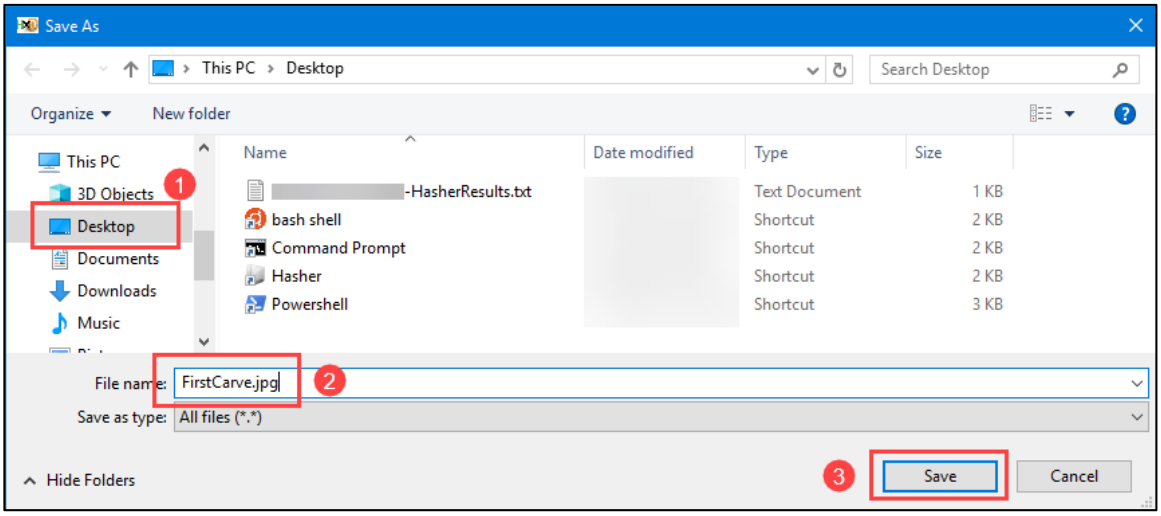
15. You will then see data in the window. All the data will be red, indicating an edit.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000FE0	31	64	61	9E	8E	1A	A9	09	D8	C0	B9	5B	98	26	29	16	1dažŽ.©.ØÀ¹[˜&).
00000FF0	E9	13	B3	07	03	F9	D6	8A	E4	0E	01	8F	56	AF	30	D0	é.³..ùÖŠä...V¯0Ð
00001000	70	5E	C7	3F	8D	02	0D	A0	0C	FF	00	23	40	0E	50	81	p^Ç?... .ÿ.#@.P.
00001010	72	31	EF	40	08	65	81	06	0B	AE	7E	A2	8B	00	0B	8B	r1ï@.e...®~¢‹..‹
00001020	70	3F	D6	26	7E	A2	9D	80	3E	D2	9E	E7	FE	02	4D	16	p?Ö&~¢.€>Òžçþ.M.
00001030	0B	8F	13	8C	64	23	FD	42	1F	F0	A2	C1	70	F3	B8	E2	...Œd#ýB.ð¢Ápó¸â
00001040	39	3F	14	34	58	2E	27	9C	C7	81	0C	A7	FE	02	05	16	9?.4X.'œÇ..§þ...
00001050	0B	8A	24	60	31	E5	48	3F	E0	4B	FE	34	AC	17	17	7B	.Š$`1åH?àKþ4¬..{
00001060	9E	91	FE	6F	40	5C	78	2F	D3	6A	83	FE	F1	A4	17	04	ž'þo@\x/Ójƒþñ¤..
00001070	92	E2	36	26	27	54	24	60	E0	E7	8F	CA	AE	33	71	D8	'â6&'T$`àç.Ê®3qØ
00001080	77	32	26	D0	23	B8	9D	A6	96	E9	D9	98	E4	83	D3	35	w2&Ð#¸.¦–éÙ˜äƒÓ5
00001090	5E	D5	85	C5	4D	16	68	86	23	D4	65	1E	E1	68	F6	CC	^Õ…ÅM.h†#Ôe.áhöÌ
000010A0	44	A9	0E	AD	6F	FE	A7	52	0D	FE	FA	9F	F1	AA	F6	DE	D©..oþ§R.þúŸñªöÞ
000010B0	40	4F	6F	35	F8	BE	82	E2	FB	6C	DE	50	DA	4C	7D	C7	@Oo5ø¾‚âûlÞPÚL}Ç
000010C0	D3	1E	E6	A9	55	4C	69	9A	D7	FA	E4	16	EF	6F	30	8A	Ó.æ©ULiš×úä.ïo0Š
000010D0	49	0A	36	5B	6C	64	90	08	E7	07	F3	AB	E6	8F	71	99	I.6[ld..ç.ó«æ.q™
000010E0	7A	A7	8B	45	C2	04	8F	4F	BB	00	1C	E7	61	00	D1	CC	z§‹EÂ..O»..ça.ÑÌ
000010F0	BB	92	72	D7	9A	AB	CF	39	75	5B	B8	C6	31	80	F8	FE	»'r×š«Ï9u[¸Æ1€øþ
00001100	94	EE	80	E8	FC	85	EA	5E	53	91	FF	00	3D	1B	FC	6B	"î€èü…ê^S'ÿ.=.ük
00001110	92	E1	61	C2	D6	3D	B9	3B	8F	D5	89	FE	B4	5D	80	EF	'áaÂÖ=¹;.Õ‰þ´]€ï
00001120	22	2E	07	96	9F	8A	E6	8B	B0	0F	22	2C	91	E5	47	C1	"..–ŸŠæ‹°.",'åGÁ
00001130	C7	DC	14	5D	80	F4	50	A4	85	55	18	F4	02	90	0F	6F	ÇÜ.]€ôP¤…U.ô...o
00001140	94	62	90	EC	20	23	27	03	A5	02	1E	B9	CF	5C	53	01	"b.ì #'.¥..¹Ï\S.
00001150	08	C1	CE	4F	E7	40	0F	27	81	9C	1E	DD	28	01	30	3D	.ÁÎOç@.'.œ.Ý(.0=
00001160	29	00	E0	A3	34	00	8C	B8	34	00	AA	39	EB	D0	50	30	).à£4.Œ¸4.ª9ëÐP0
00001170	07	83	91	4C	42	70	0F	4A	00	50	01	EB	40	07	03	1C	.ƒ'LBp.J.P.ë@...
00001180	50	01	BB	EB	F9	D1	60	1A	C4	8E	BC	F3	4A	C0	31	BA	P.»ëùÑ`.ÄŽ¼óJÀ1º
00001190	F4	14	01	FF	D9	FF	E0	00	10	4A	46	49	46	00	01	00	ô..ÿÙÿà..JFIF...
000011A0	01	00	96	00	96	00	00	FF	FE	00	1F	4C	45	41	44	20	..–.–..ÿþ..LEAD
000011B0	54	65	63	68	6E	6F	6C	6F	67	69	65	73	20	49	6E	63	Technologies Inc
000011C0	2E	20	56	31	2E	30	31	00	FF	DB							. V1.01.ÿÛ

16. Click on the Save icon at the top of the program window.



17. You will be asked for a location and name. Enter the info as shown and save to your **VM Desktop**.



Congratulations! You have just performed your first raw data carve.

Exercise Questions

1. Navigate to your **VM Desktop** and find the file you just carved. Try to open it.

   a. What is it a picture of?

   _____

   b. How big is the picture in KB?

   _____

   c. Is there anything we can deduce from the size?

   _____

Exercise Questions - Step-by-Step

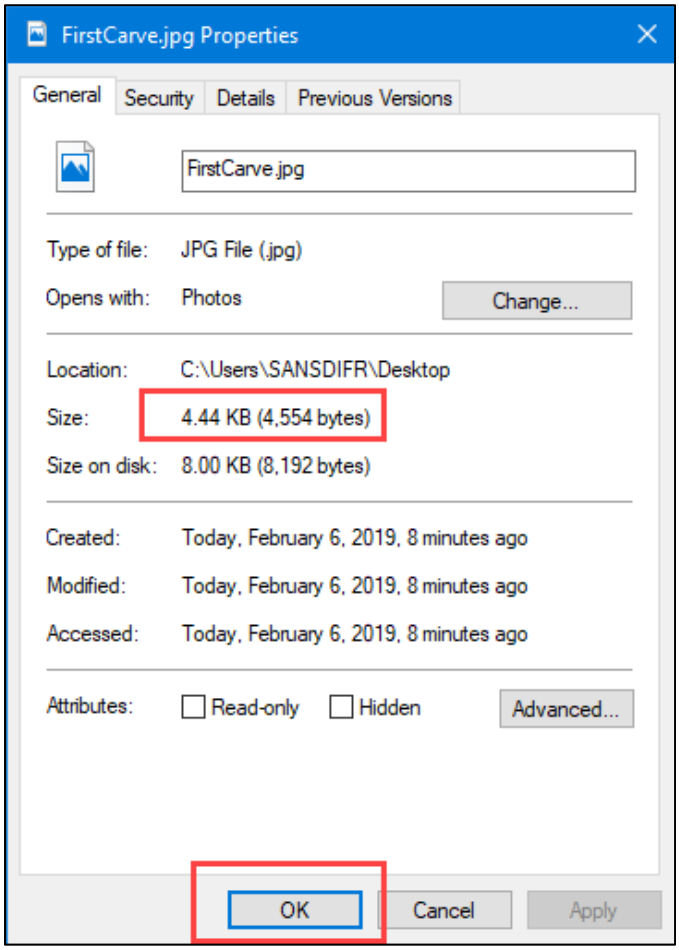
1. Navigate to your **VM Desktop** and find the file you just carved. Try to open it.

   a. What is it a picture of?

   A couple of houses and a car

   b. How big is the picture in KB?

   4.4 KB (8 KB on disk). Right-click on the icon and select Properties. Click OK to close it.



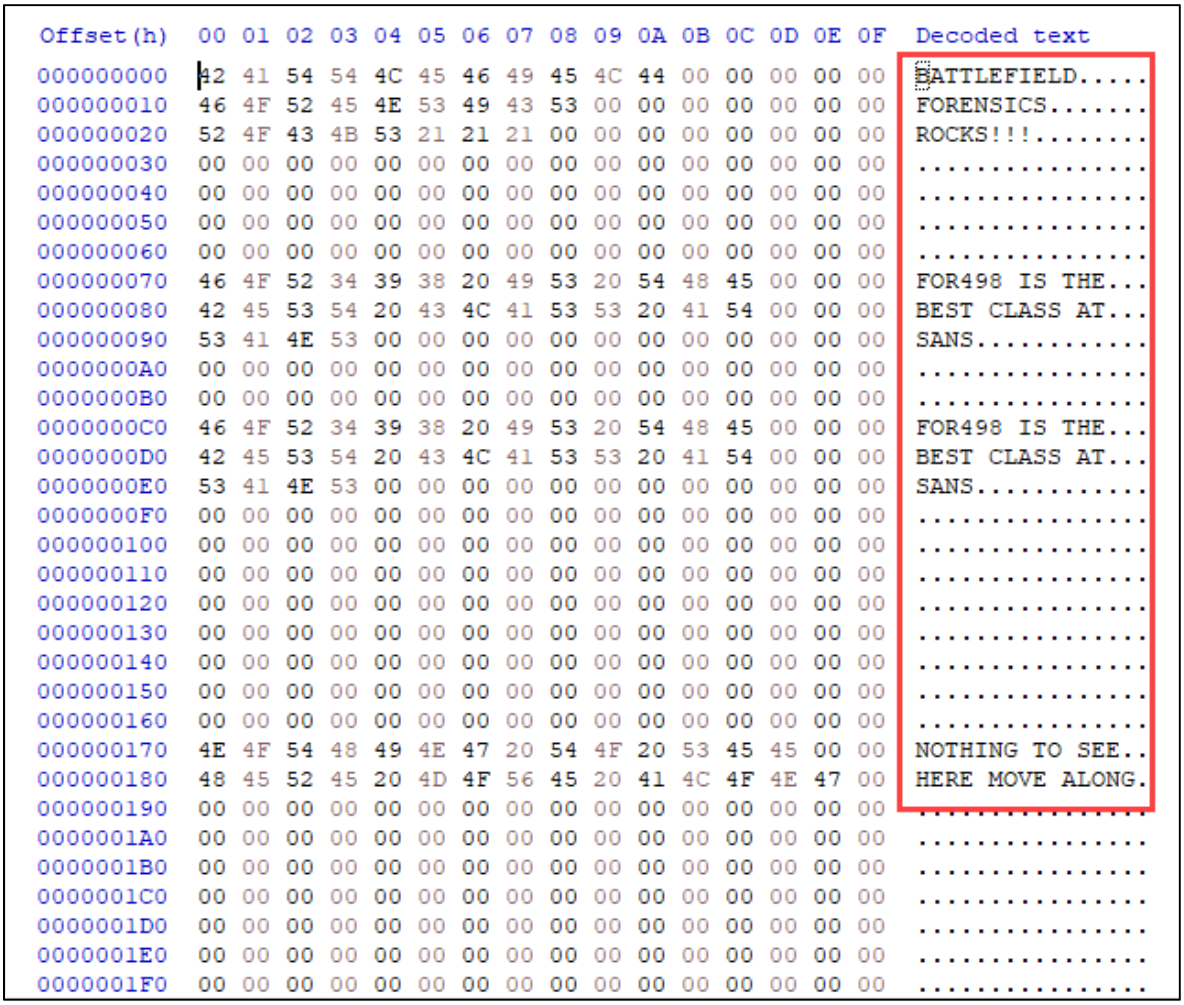
   c. Is there anything we can deduce from the size?

   This is a likely a thumbnail of a larger photo

Exercise - Manual Carving of .DOC File

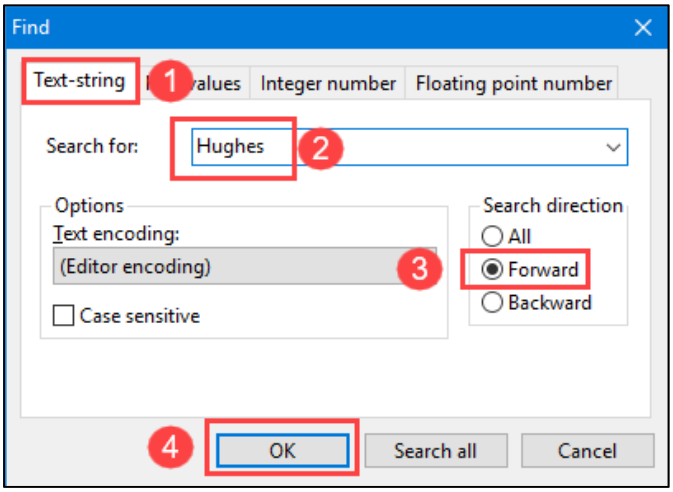
1. If you closed **HxD** and/or your forensic image, refer back to the first exercise in this lab to get started again.

2. If you are continuing directly from the last exercise, close all windows/tabs in **HxD** except for the original window that has the physical drive represented in it. Using the scroll bar on the right of the data pane, drag it all the way to the top so that you are back at zero sector, and then click anywhere in the hex output to place the cursor. Remember that you should be looking at the weird comments in the **Decoded text** column. If you do not see this, go back to the beginning of the first section and start again.

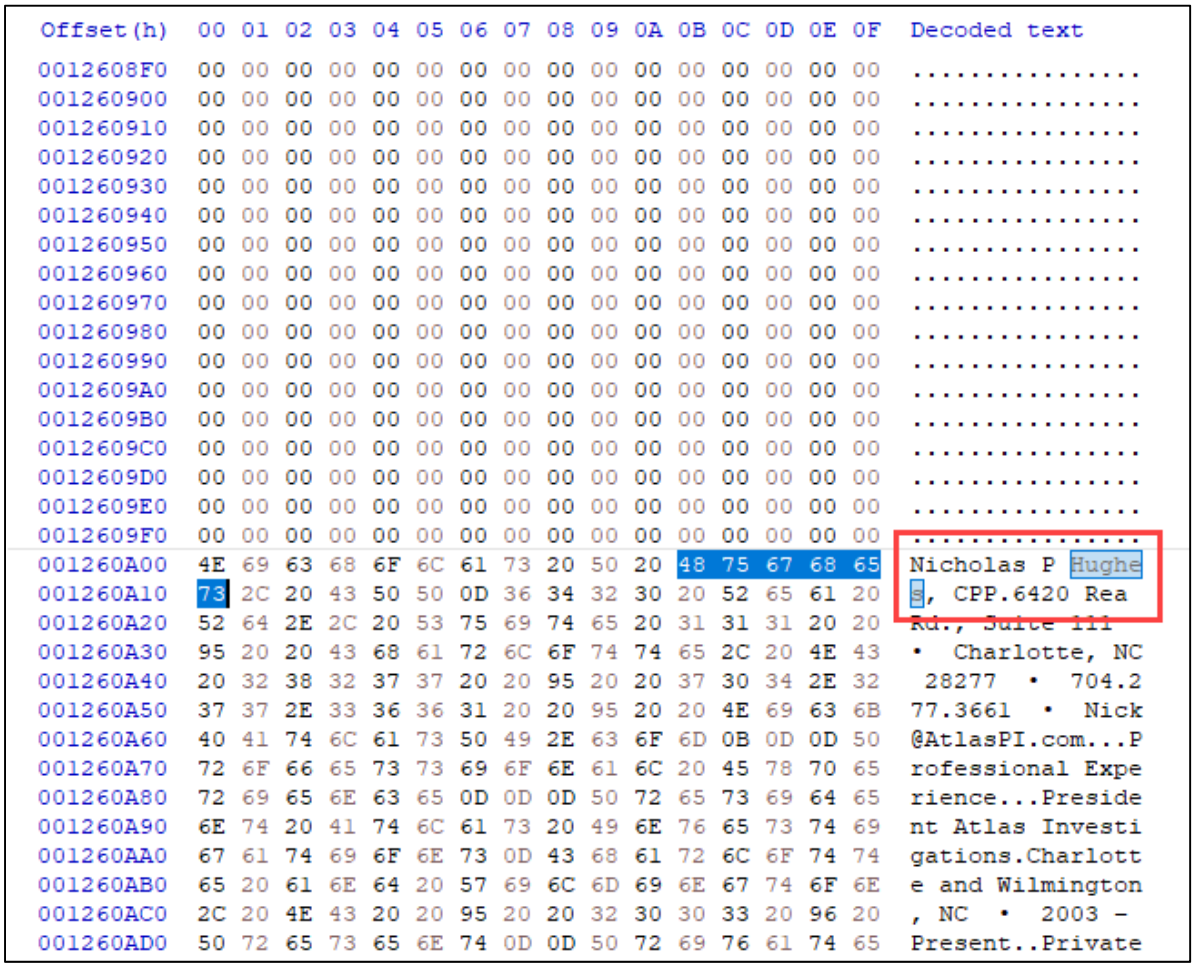


Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0000000000	42	41	54	54	4C	45	46	49	45	4C	44	00	00	00	00	00	BATTLEFIELD.....
0000000010	46	4F	52	45	4E	53	49	43	53	00	00	00	00	00	00	00	FORENSICS.....
0000000020	52	4F	43	4B	53	21	21	21	00	00	00	00	00	00	00	00	ROCKS!!!.....
0000000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000070	46	4F	52	34	39	38	20	49	53	20	54	48	45	00	00	00	FOR498 IS THE...
0000000080	42	45	53	54	20	43	4C	41	53	53	20	41	54	00	00	00	BEST CLASS AT...
0000000090	53	41	4E	53	00	00	00	00	00	00	00	00	00	00	00	00	SANS.....
00000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000000C0	46	4F	52	34	39	38	20	49	53	20	54	48	45	00	00	00	FOR498 IS THE...
00000000D0	42	45	53	54	20	43	4C	41	53	53	20	41	54	00	00	00	BEST CLASS AT...
00000000E0	53	41	4E	53	00	00	00	00	00	00	00	00	00	00	00	00	SANS.....
00000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000170	4E	4F	54	48	49	4E	47	20	54	4F	20	53	45	45	00	00	NOTHING TO SEE..
0000000180	48	45	52	45	20	4D	4F	56	45	20	41	4C	4F	4E	47	00	HERE MOVE ALONG.
0000000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

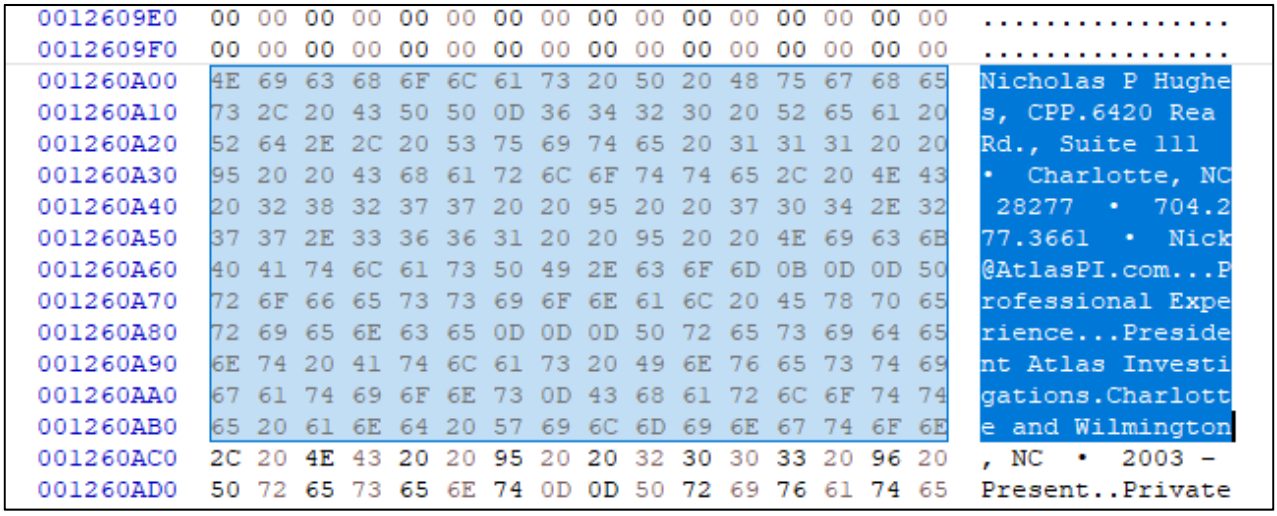
3. Using **Search → Find** again, your goal this time is to find a file with some text in it. Specifically, the name of **Nicholas Hughes**. This time use the **Text-string** tab, and in the **Search for** field, input **`Hughes`**. Ensure that the **Search direction** is set to **Forward**, and then click the **OK** button.



4. **HxD** will locate the search term, and due to what appears to be overwrite activity, there does not seem to be any data immediately prior to the search hit data. This is not uncommon in unallocated space. This is still a win, and if you were running an automated carving tool against a data set where the beginning of the file was overwritten, the carving tool would not have recovered what you see here.

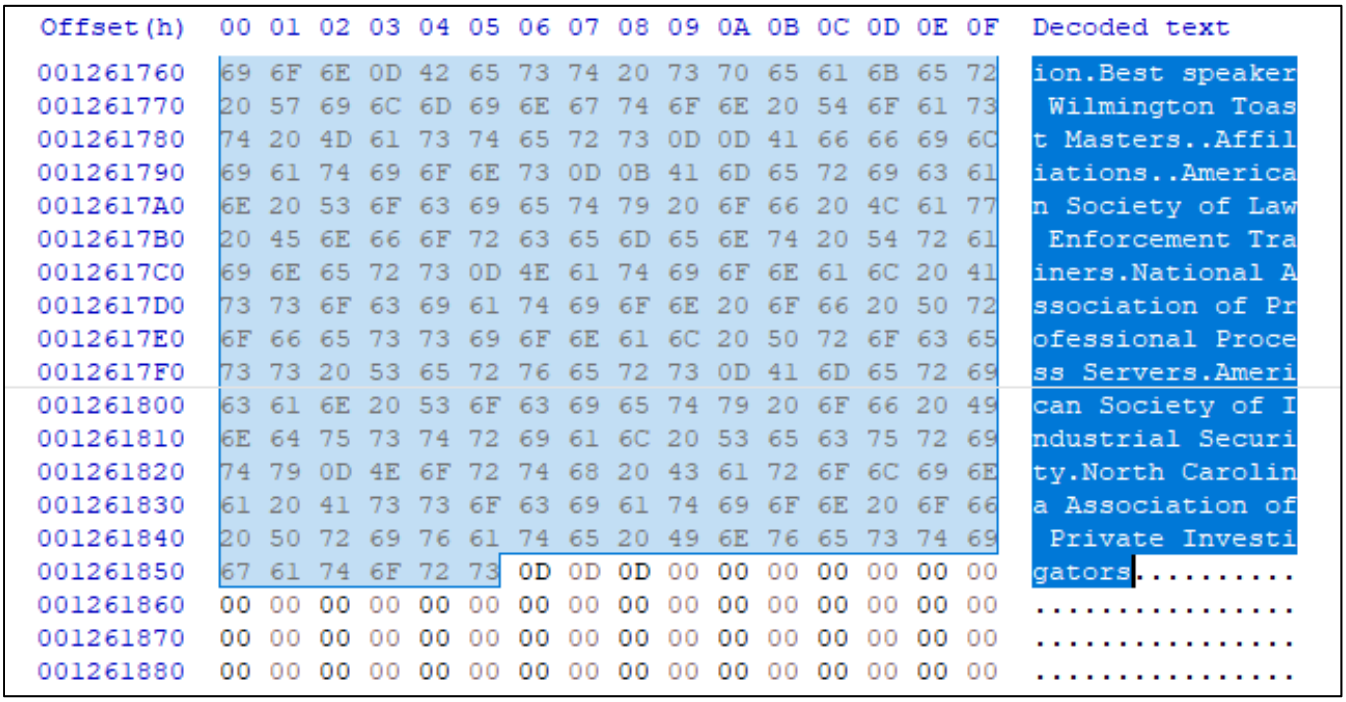


5. Plain text is better than no text, so using the technique from the previous section, place the cursor at the beginning of the text (**Nicholas**), hold down the **Shift** key, and use the **DOWN** arrow to start sweeping data.



```
0012609E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012609F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001260A00  4E 69 63 68 6F 6C 61 73 20 50 20 48 75 67 68 65  Nicholas P Hughe
001260A10  73 2C 20 43 50 50 0D 36 34 32 30 20 52 65 61 20  s, CPP.6420 Rea
001260A20  52 64 2E 2C 20 53 75 69 74 65 20 31 31 31 20 20  Rd., Suite 111
001260A30  95 20 20 43 68 61 72 6C 6F 74 74 65 2C 20 4E 43  •  Charlotte, NC
001260A40  20 32 38 32 37 37 20 20 95 20 20 37 30 34 2E 32   28277  •  704.2
001260A50  37 37 2E 33 36 36 31 20 20 95 20 20 4E 69 63 6B  77.3661  •  Nick
001260A60  40 41 74 6C 61 73 50 49 2E 63 6F 6D 0B 0D 0D 50  @AtlasPI.com...P
001260A70  72 6F 66 65 73 73 69 6F 6E 61 6C 20 45 78 70 65  rofessional Expe
001260A80  72 69 65 6E 63 65 0D 0D 0D 50 72 65 73 69 64 65  rience...Preside
001260A90  6E 74 20 41 74 6C 61 73 20 49 6E 76 65 73 74 69  nt Atlas Investi
001260AA0  67 61 74 69 6F 6E 73 0D 43 68 61 72 6C 6F 74 74  gations.Charlott
001260AB0  65 20 61 6E 64 20 57 69 6C 6D 69 6E 67 74 6F 6E  e and Wilmington
001260AC0  2C 20 4E 43 20 20 95 20 20 32 30 30 33 20 96 20  , NC  •  2003 –
001260AD0  50 72 65 73 65 6E 74 0D 0D 50 72 69 76 61 74 65  Present..Private
```

6. Continue sweeping until you arrive at what appears to be the end of the data.

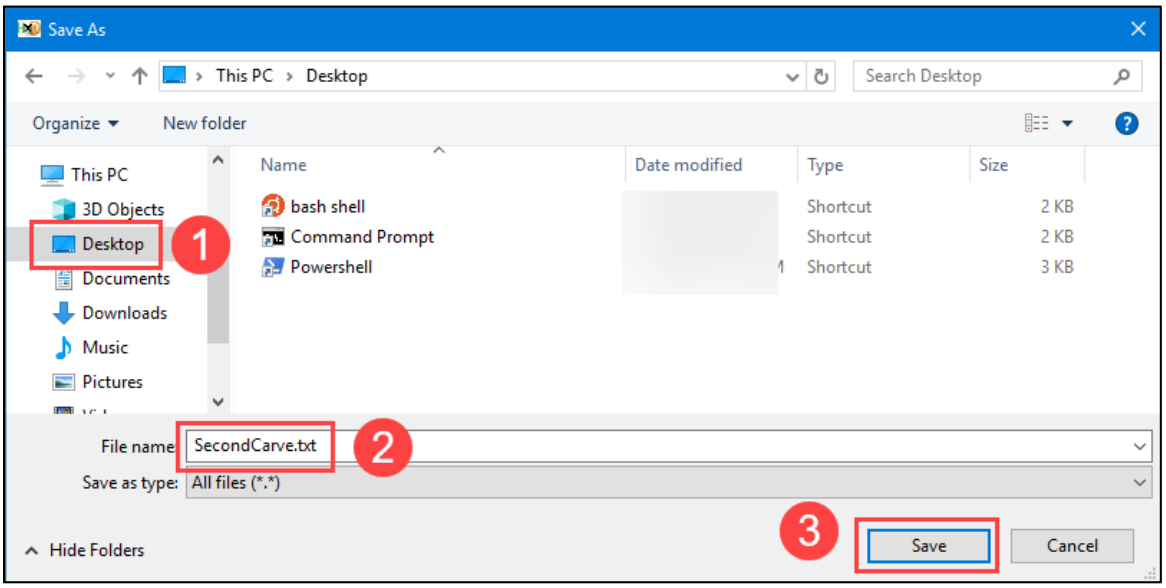


```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
001261760  69 6F 6E 0D 42 65 73 74 20 73 70 65 61 6B 65 72  ion.Best speaker
001261770  20 57 69 6C 6D 69 6E 67 74 6F 6E 20 54 6F 61 73   Wilmington Toas
001261780  74 20 4D 61 73 74 65 72 73 0D 0D 41 66 66 69 6C  t Masters..Affil
001261790  69 61 74 69 6F 6E 73 0D 0B 41 6D 65 72 69 63 61  iations..America
0012617A0  6E 20 53 6F 63 69 65 74 79 20 6F 66 20 4C 61 77  n Society of Law
0012617B0  20 45 6E 66 6F 72 63 65 6D 65 6E 74 20 54 72 61   Enforcement Tra
0012617C0  69 6E 65 72 73 0D 4E 61 74 69 6F 6E 61 6C 20 41  iners.National A
0012617D0  73 73 6F 63 69 61 74 69 6F 6E 20 6F 66 20 50 72  ssociation of Pr
0012617E0  6F 66 65 73 73 69 6F 6E 61 6C 20 50 72 6F 63 65  ofessional Proce
0012617F0  73 73 20 53 65 72 76 65 72 73 0D 41 6D 65 72 69  ss Servers.Ameri
001261800  63 61 6E 20 53 6F 63 69 65 74 79 20 6F 66 20 49  can Society of I
001261810  6E 64 75 73 74 72 69 61 6C 20 53 65 63 75 72 69  ndustrial Securi
001261820  74 79 0D 4E 6F 72 74 68 20 43 61 72 6F 6C 69 6E  ty.North Carolin
001261830  61 20 41 73 73 6F 63 69 61 74 69 6F 6E 20 6F 66  a Association of
001261840  20 50 72 69 76 61 74 65 20 49 6E 76 65 73 74 69   Private Investi
001261850  67 61 74 6F 72 73 0D 0D 0D 00 00 00 00 00 00 00  gators..........
001261860  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001261870  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001261880  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

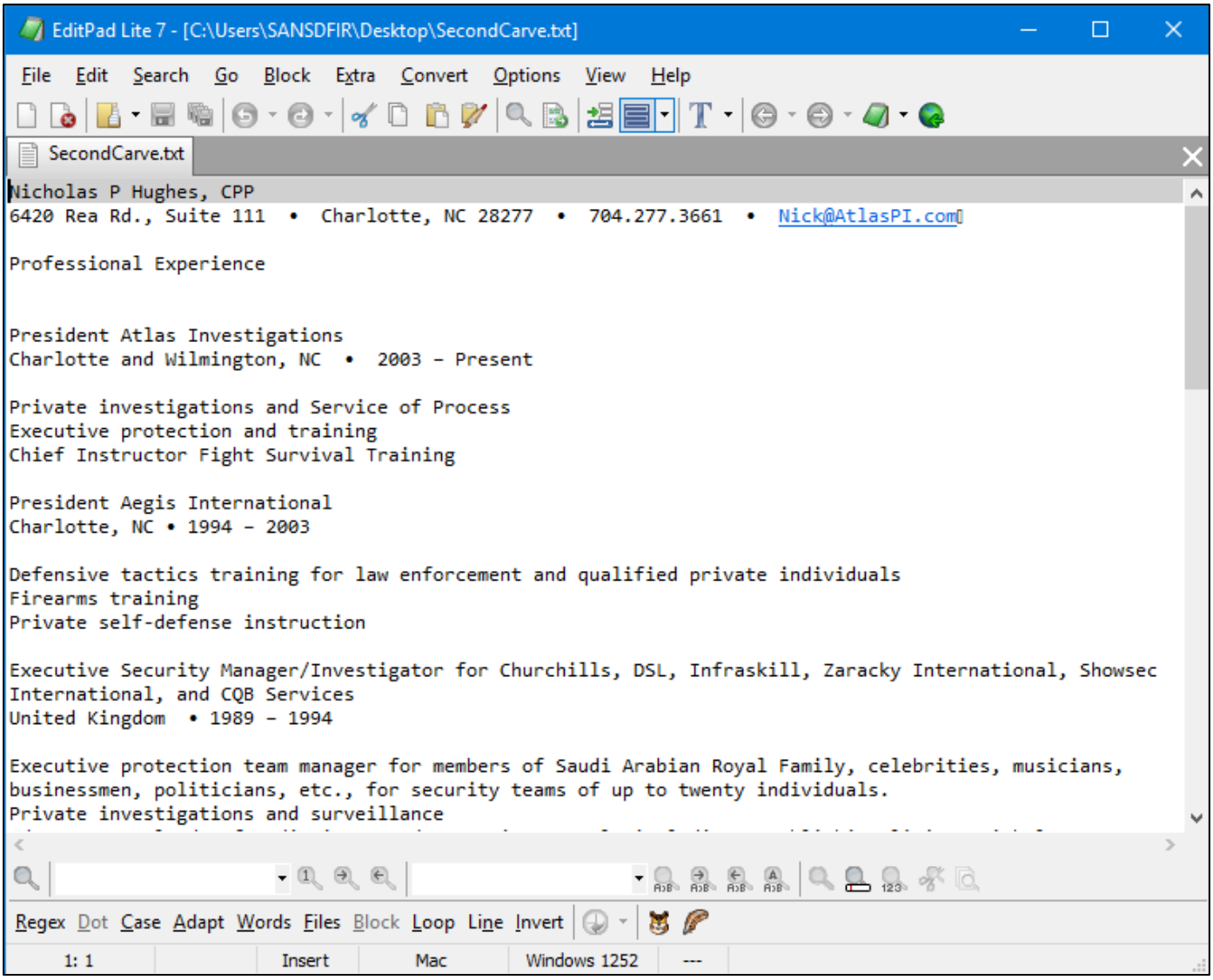
7. As in a previous example, press **Ctrl+C** to copy, and then create a new data area by clicking on the white paper icon in **HxD**. Accept any prompts that appear.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000C70   61 64 75 61 74 65 0D 53 70 65 61 6B 20 66 6C 75  aduate.Speak flu
00000C80   65 6E 74 20 46 72 65 6E 63 68 20 61 6E 64 20 74  ent French and t
00000C90   6F 75 72 69 73 74 2D 6C 65 76 65 6C 20 47 65 72  ourist-level Ger
00000CA0   6D 61 6E 20 61 6E 64 20 53 70 61 6E 69 73 68 0D  man and Spanish.
00000CB0   57 69 6C 64 65 72 6E 65 73 73 20 45 6D 65 72 67  Wilderness Emerg
00000CC0   65 6E 63 79 20 46 69 72 73 74 20 52 65 73 70 6F  ency First Respo
00000CD0   6E 64 65 72 92 73 20 43 6F 75 72 73 65 20 67 72  nder's Course gr
00000CE0   61 64 75 61 74 65 0D 0D 48 6F 6E 6F 72 73 0D 0D  aduate..Honors..
00000CF0   52 65 63 69 70 69 65 6E 74 20 6F 66 20 4C 69 66  Recipient of Lif
00000D00   65 74 69 6D 65 20 41 63 68 69 65 76 65 6D 65 6E  etime Achievemen
00000D10   74 20 41 77 61 72 64 20 4D 61 72 74 69 61 6C 20  t Award Martial
00000D20   41 72 74 73 20 4D 61 73 74 65 72 73 20 48 61 6C  Arts Masters Hal
00000D30   6C 20 6F 66 20 46 61 6D 65 0D 48 6F 6E 6F 72 61  l of Fame.Honora
00000D40   62 6C 65 20 64 69 73 63 68 61 72 67 65 20 46 72  ble discharge Fr
00000D50   65 6E 63 68 20 46 6F 72 65 69 67 6E 20 4C 65 67  ench Foreign Leg
00000D60   69 6F 6E 0D 42 65 73 74 20 73 70 65 61 6B 65 72  ion.Best speaker
00000D70   20 57 69 6C 6D 69 6E 67 74 6F 6E 20 54 6F 61 73   Wilmington Toas
00000D80   74 20 4D 61 73 74 65 72 73 0D 0D 41 66 66 69 6C  t Masters..Affil
00000D90   69 61 74 69 6F 6E 73 0D 0B 41 6D 65 72 69 63 61  iations..America
00000DA0   6E 20 53 6F 63 69 65 74 79 20 6F 66 20 4C 61 77  n Society of Law
00000DB0   20 45 6E 66 6F 72 63 65 6D 65 6E 74 20 54 72 61   Enforcement Tra
00000DC0   69 6E 65 72 73 0D 4E 61 74 69 6F 6E 61 6C 20 41  iners.National A
00000DD0   73 73 6F 63 69 61 74 69 6F 6E 20 6F 66 20 50 72  ssociation of Pr
00000DE0   6F 66 65 73 73 69 6F 6E 61 6C 20 50 72 6F 63 65  ofessional Proce
00000DF0   73 73 20 53 65 72 76 65 72 73 0D 41 6D 65 72 69  ss Servers.Ameri
00000E00   63 61 6E 20 53 6F 63 69 65 74 79 20 6F 66 20 49  can Society of I
00000E10   6E 64 75 73 74 72 69 61 6C 20 53 65 63 75 72 69  ndustrial Securi
00000E20   74 79 0D 4E 6F 72 74 68 20 43 61 72 6F 6C 69 6E  ty.North Carolin
00000E30   61 20 41 73 73 6F 63 69 61 74 69 6F 6E 20 6F 66  a Association of
00000E40   20 50 72 69 76 61 74 65 20 49 6E 76 65 73 74 69   Private Investi
00000E50   67 61 74 6F 72 73                               gators
```

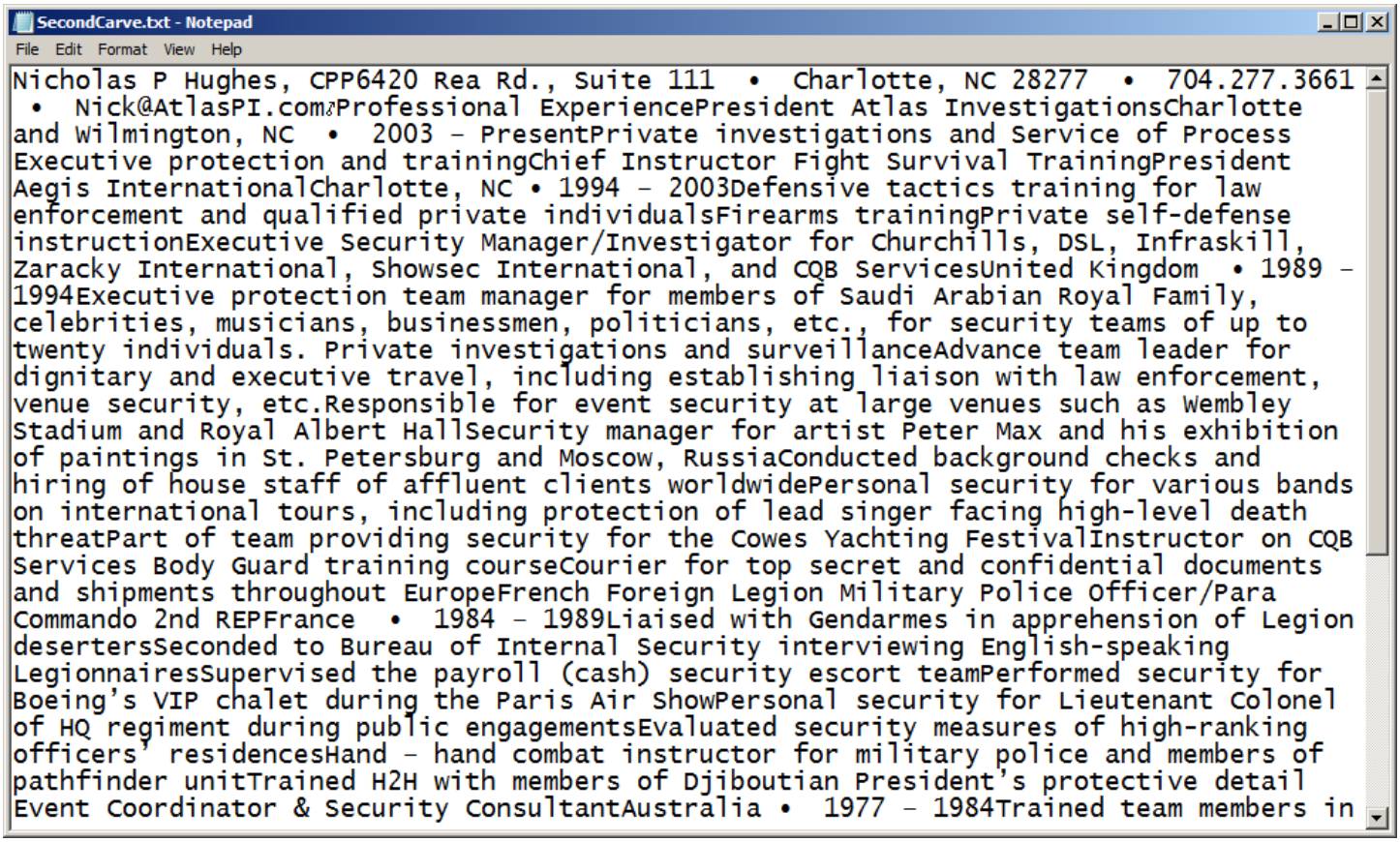
8. Click on the floppy disk image to save this data out to a file as per the information in the following screen. Since we don't know where this data came from, we will save it out as a text file for now. Save it to your **Desktop**.



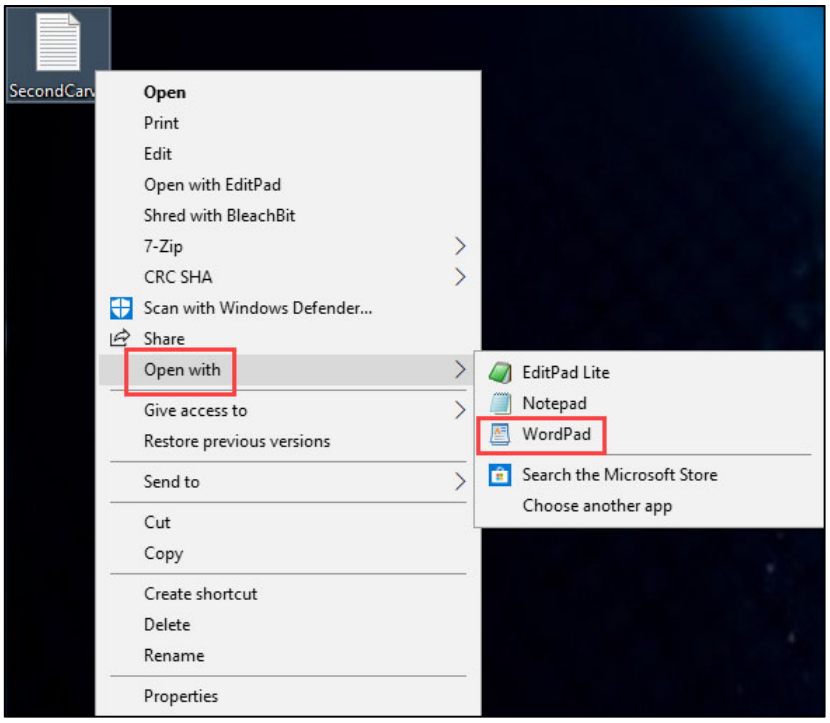
9. Navigate to your **Desktop** and locate and open the file. You now see the contents in a readable format that you can provide to a client.



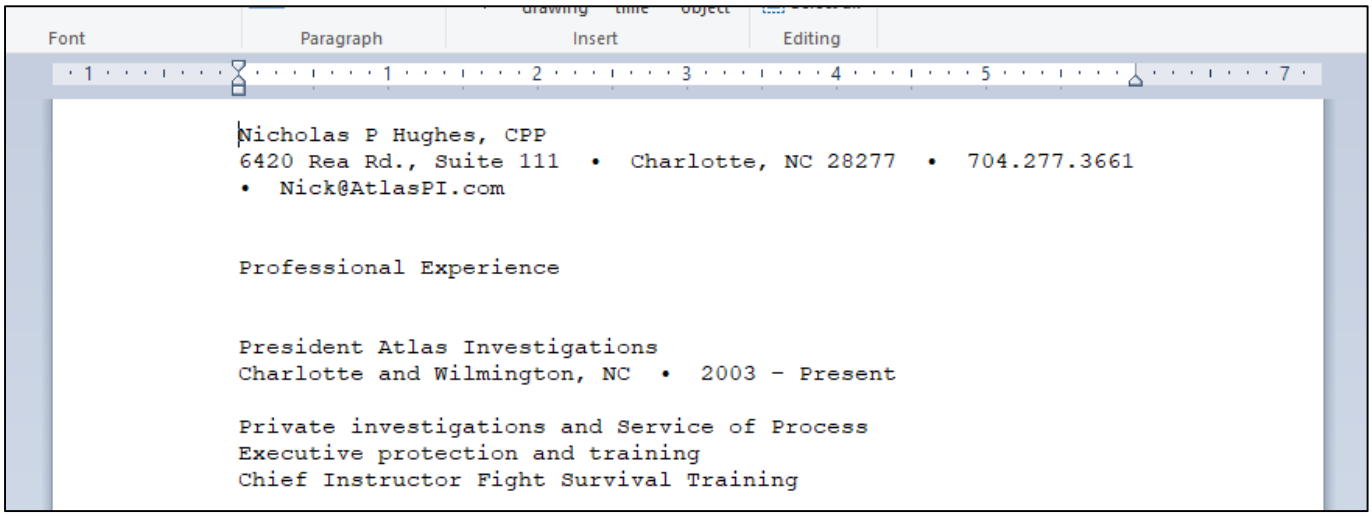
10. The only reason you see it laid out with clean formatting in the previous step is because of the default text reader in your **VM**. In reality, on the majority of computers, and indeed your clients' computers, they will be using the Windows default text reader of **Notepad**. Then the output would look like this (in anything but the latest version of Windows 10). Clearly this is not optimal.



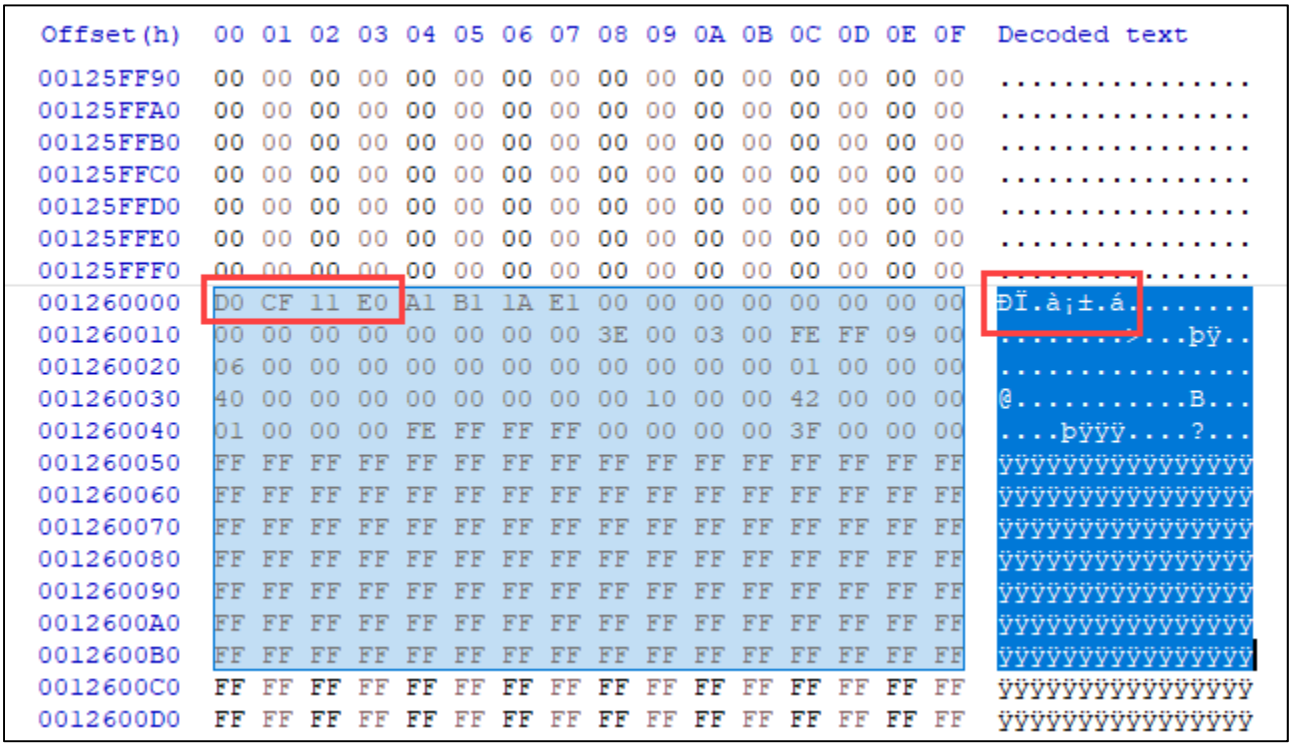
11. The file opens and we now have our carved text that would have been missed by an automated carving tool. But now we are greedy, and we want the data to look more organized than the continuous pile of text we just recovered. Close the .**txt** file, and then right click on the icon for the text file, select **Open with**, and then **WordPad**.



12. The recovered text should now look like this. It has structure and formatting! Which copy would you rather give to your client? If you were the client, which copy would you rather have?

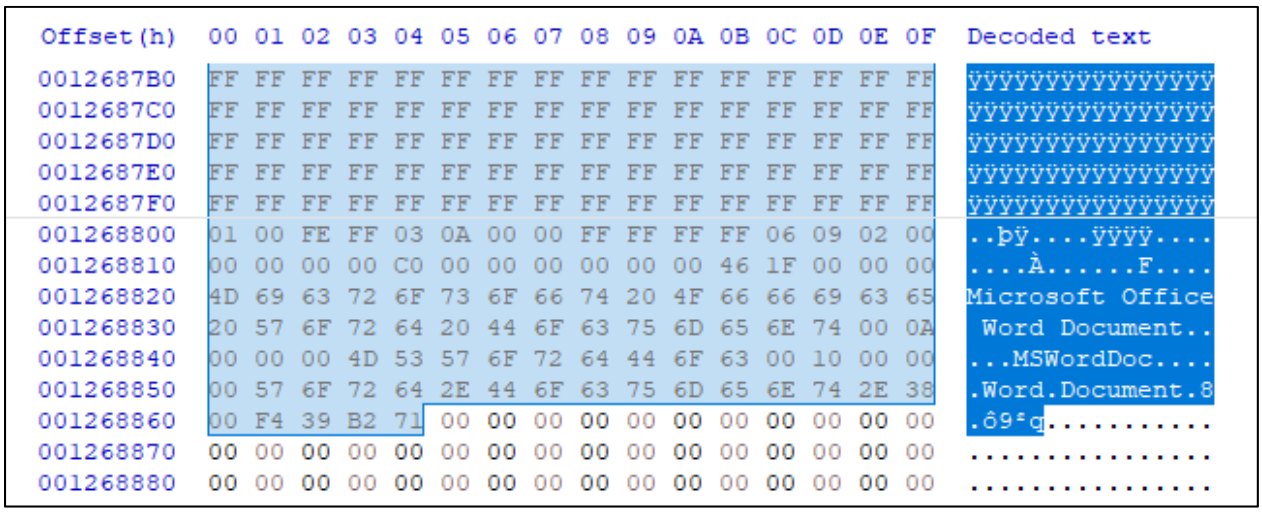


13. But we are even greedier yet. Are you sure that the file was overwritten? Did you bother to check? Or did you just grab the text you could see from the hex editor? Let's go back and look. Return to the original physical disk view in **HxD**, where you had found the hit on **Hughes**. Without actually sweeping any data yet, just scroll up from there to see if you can locate a file header.



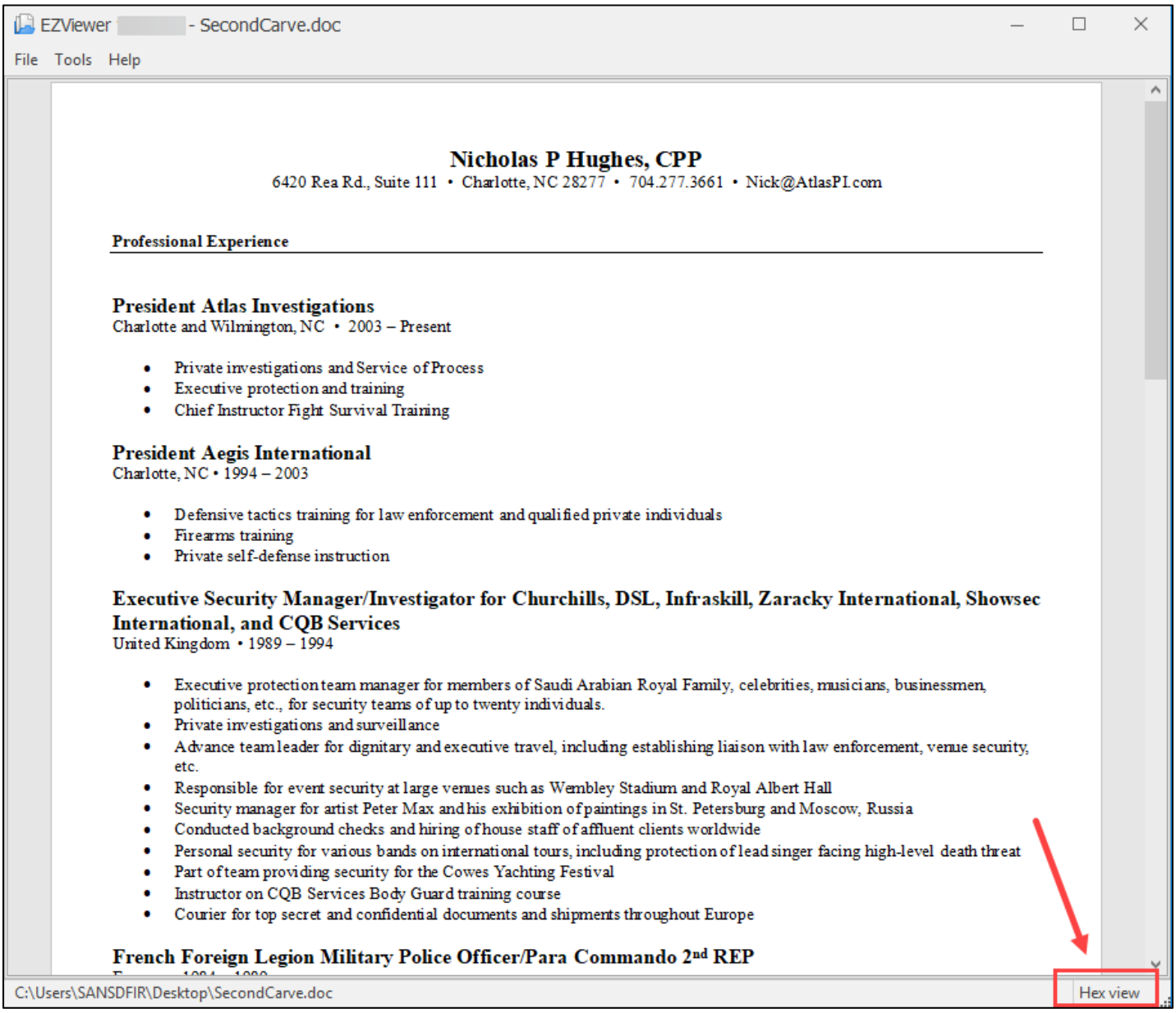
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00125FF90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00125FFA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00125FFB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00125FFC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00125FFD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00125FFE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00125FFF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001260000	D0	CF	11	E0	A1	B1	1A	E1	00	00	00	00	00	00	00	00	ÐÏ.à¡±.á.....
001260010	00	00	00	00	00	00	00	00	3E	00	03	00	FE	FF	09	00	........>...þÿ..
001260020	06	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	.....
001260030	40	00	00	00	00	00	00	00	00	10	00	00	42	00	00	00	@.....B...
001260040	01	00	00	00	FE	FF	FF	FF	00	00	00	00	3F	00	00	00	....þÿÿÿ....?...
001260050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
001260060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
001260070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
001260080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
001260090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012600A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012600B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012600C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012600D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

14. Indeed, we locate the file header for a **Word** document. Because the text was in plain view, we know this is a .**DOC** and not a .**DOCX**. Start sweeping at the beginning of the header as indicated above, and sweep data down past the data you already collected. Keep on going. Even when you think it is hopeless, keep on going. You are looking for the file footer of the .**DOC**. It looks like this: **ô9²q.** You will think that you must have passed it, or it has been overwritten. Here is the Kevin Ripa rule of thumb. If you haven't found the footer, then you keep sweeping, no matter how long until you either find it, or find data that is verifiably not it. For example, data in a new file that looks different. This is where experience comes in.

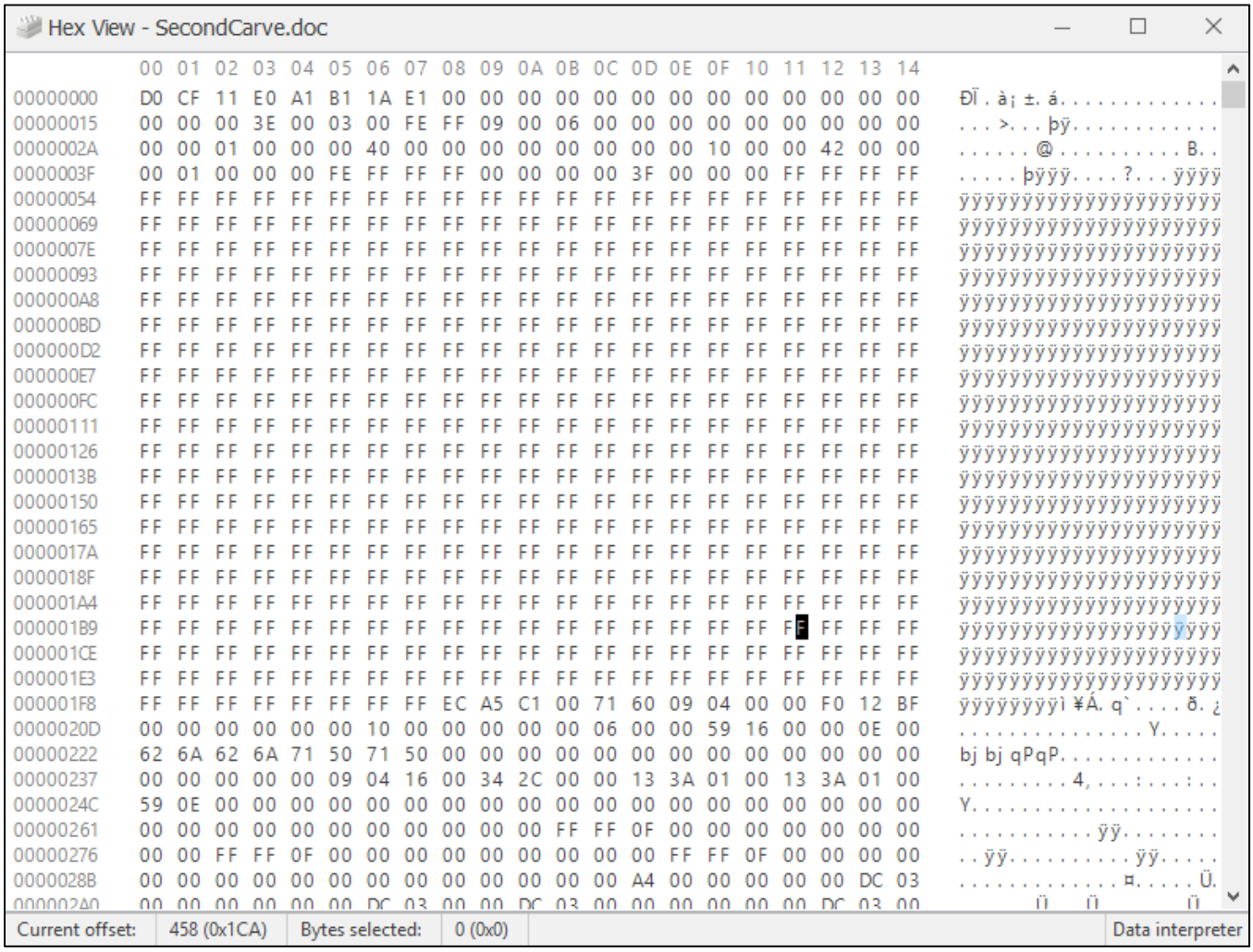


Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0012687B0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012687C0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012687D0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012687E0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012687F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
001268800	01	00	FE	FF	03	0A	00	00	FF	FF	FF	FF	06	09	02	00	..þÿ....ÿÿÿÿ....
001268810	00	00	00	00	C0	00	00	00	00	00	00	46	1F	00	00	00	....À.....F....
001268820	4D	69	63	72	6F	73	6F	66	74	20	4F	66	66	69	63	65	Microsoft Office
001268830	20	57	6F	72	64	20	44	6F	63	75	6D	65	6E	74	00	0A	Word Document..
001268840	00	00	00	4D	53	57	6F	72	64	44	6F	63	00	10	00	00	...MSWordDoc....
001268850	00	57	6F	72	64	2E	44	6F	63	75	6D	65	6E	74	2E	38	.Word.Document.8
001268860	00	F4	39	B2	71	00	00	00	00	00	00	00	00	00	00	00	.ô9²q.....
001268870	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
001268880	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

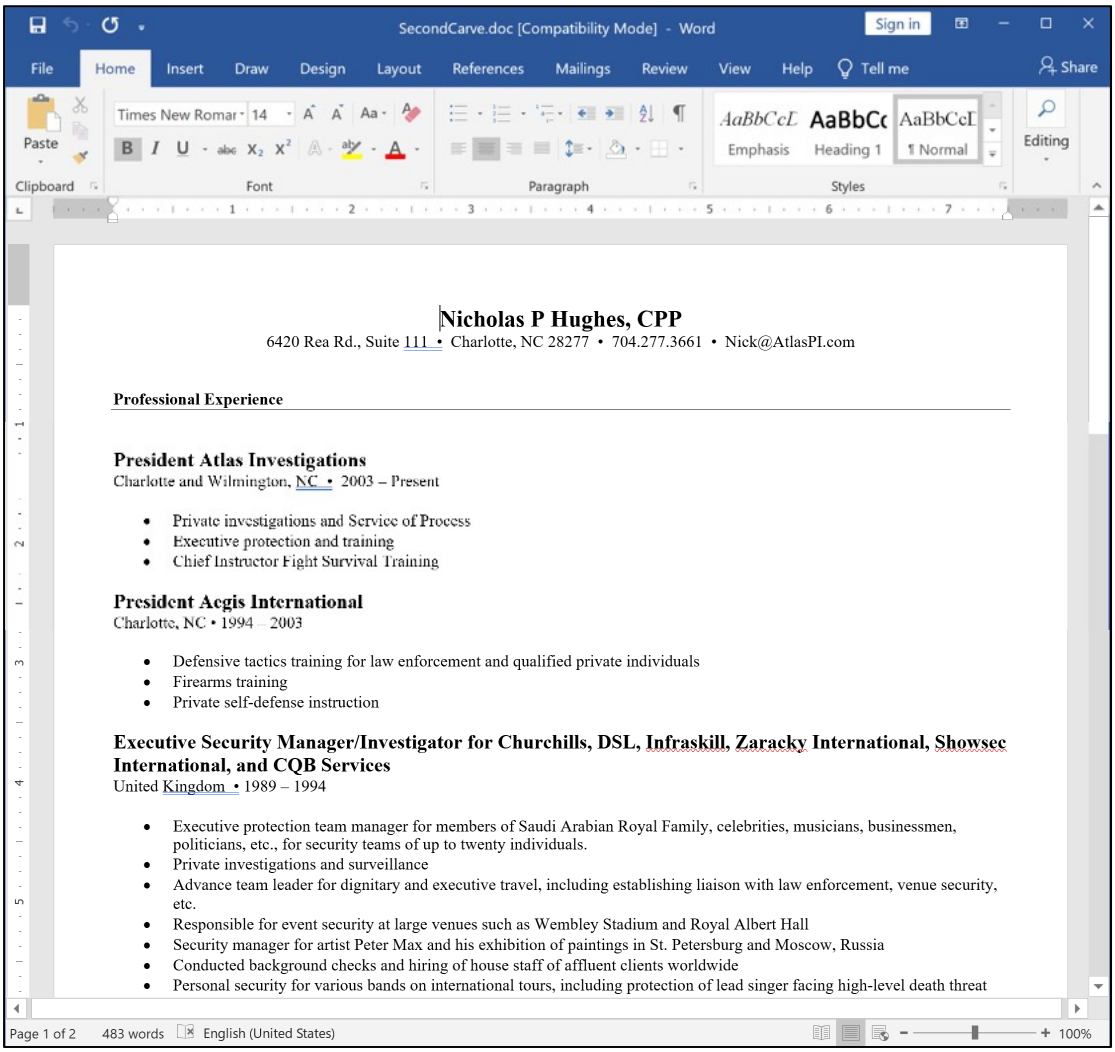
15. Using the same process as before, create a new workspace by clicking on the blank page icon, paste the data you swept, save it out to your desktop, but this time, name it **SecondCarve.doc**. Your **VM** does not have Microsoft Word to open the file you just created, but thanks to Eric Zimmerman, you have a viewer that will render it just like **Microsoft Word**. Right-click on **SecondCarve.doc** and select **Open with EZViewer**. You will see the window below.



16. There is no other viewer on the market today that can properly view such a wide range of files and do it for free. In fact, even the **Microsoft Word** viewer does not work as well. Note in the bottom right corner of the window in the previous step, a button called **Hex view**. Click on it, and you will see the window below, which is a hex representation of the file you have open. Note this looks exactly like the data you carved!



17. Let's try to view this file with **Microsoft Word** on your host machine (only if you have **Microsoft Office** installed). Close the **Hex View** window and the **EZViewer** window. Drag the **SecondCarve.doc** file from the **Desktop** of your **VM** to the **Desktop** of your **host** computer. To do this, you may need to take **VMware** out of full screen mode by clicking the relevant icon that you learned about in the first exercise on day 1. You want to be able to see your **VM Desktop** and your **host Desktop** at the same time. Click, hold, and drag **SecondCarve.doc** to your host **Desktop** and drop it. Then open it with **Microsoft Word**.



18. Out of the three versions you have seen (**Notepad, Wordpad, Word**), which one would you as the client want to see? Context and formatting mean everything. Although for our example we extracted a full file, we did this for the purpose of training and expediency. If you are missing the first few sectors, or the last few sectors, you can still rebuild the file to this state. It just takes practice and patience, and we will visit this in a later exercise.

Exercise—Key Takeaways

- Data is often available, even if it is not in a format that the examiner is familiar with.
- Data carving and rebuilding can be very time consuming and frustrating.
- If the data does not present itself in a manner that you are accustomed to, try something else.

This page intentionally left blank.

Exercise 6.3B—Data Carving & Rebuilding - Webpages & Images

Exercise 6.3 Objectives

- Recognize various common file signatures
- Understand OLECF and OpenXML Office formats
- Carve manually for a deleted file
- Deal with potentially unknown image formats

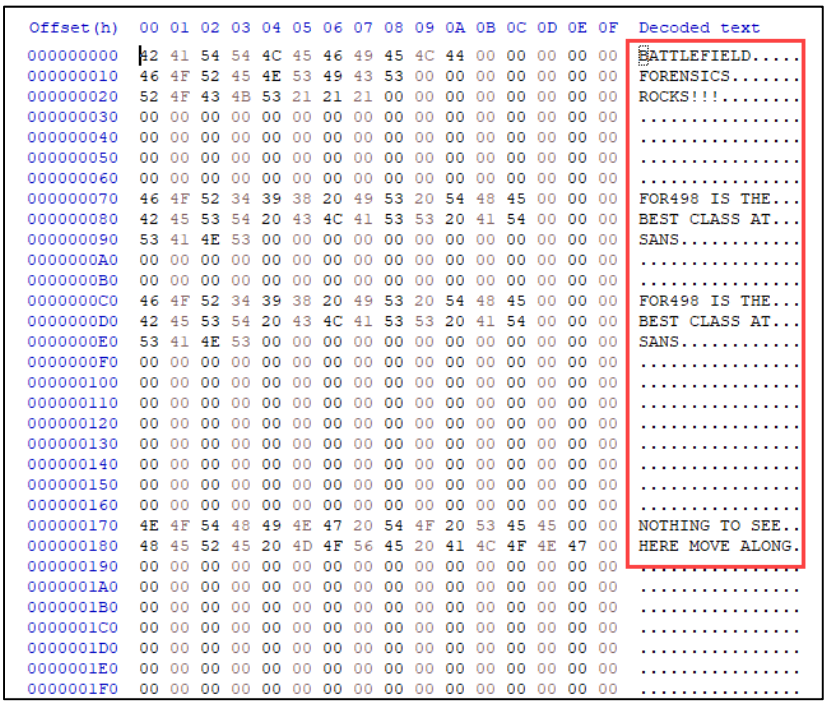
Exercise - Manual Carving of Webpage

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. If you closed **HxD** and/or your forensic image, refer back to exercise **6.3A** to get started again.

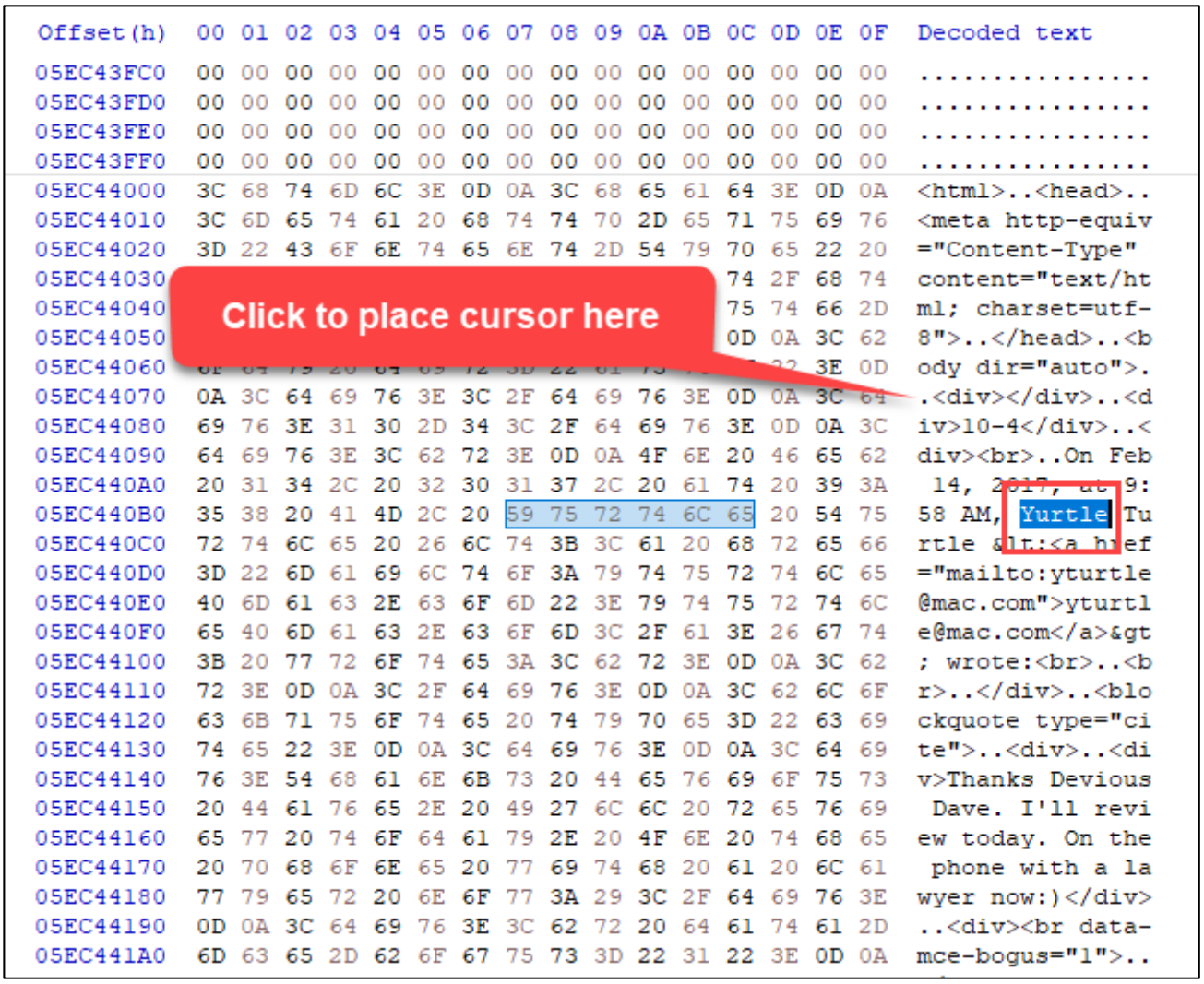
4. If you are continuing directly from the last exercise, close all windows/tabs in **HxD** except for the original window that has the physical drive represented in it. Using the scroll bar on the right of the data pane, drag it all the way to the top so that you are back at zero sector, and then click anywhere in the hex output to place the cursor. Remember that you should be looking at the weird comments in the **Decoded text** column. If you are not, go back to the beginning of the first section and start again.



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
000000000	42	41	54	54	4C	45	46	49	45	4C	44	00	00	00	00	00	BATTLEFIELD.....
000000010	46	4F	52	45	4E	53	49	43	53	00	00	00	00	00	00	00	FORENSICS.....
000000020	52	4F	43	4B	53	21	21	21	00	00	00	00	00	00	00	00	ROCKS!!!.....
000000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000070	46	4F	52	34	39	38	20	49	53	20	54	48	45	00	00	00	FOR498 IS THE...
000000080	42	45	53	54	20	43	4C	41	53	53	20	41	54	00	00	00	BEST CLASS AT...
000000090	53	41	4E	53	00	00	00	00	00	00	00	00	00	00	00	00	SANS.....
0000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000000C0	46	4F	52	34	39	38	20	49	53	20	54	48	45	00	00	00	FOR498 IS THE...
0000000D0	42	45	53	54	20	43	4C	41	53	53	20	41	54	00	00	00	BEST CLASS AT...
0000000E0	53	41	4E	53	00	00	00	00	00	00	00	00	00	00	00	00	SANS.....
0000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000170	4E	4F	54	48	49	4E	47	20	54	4F	20	53	45	45	00	00	NOTHING TO SEE..
000000180	48	45	52	45	20	4D	4F	56	45	20	41	4C	4F	4E	47	00	HERE MOVE ALONG.
000000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

5. Now let's try a different type of file: the webpage. This is a very forgiving file type. In many cases, we can find a very small portion of a webpage, and depending on its content, it will open and show us what it can. It takes very little manipulation to make it viewable. In this section, we will carve a portion of a web page and see if it will build.

6. Using **Search → Find** again, your goal is to find a file with some text in it. Specifically, the name of **Yurtle the Turtle**. This time, using the **Text-string** tab, input **`Yurtle`** into the **Search for** field. Ensure that the **Search direction** is set to **Forward**, and then click **OK**. When found, click the cursor part way into the data area as indicated.

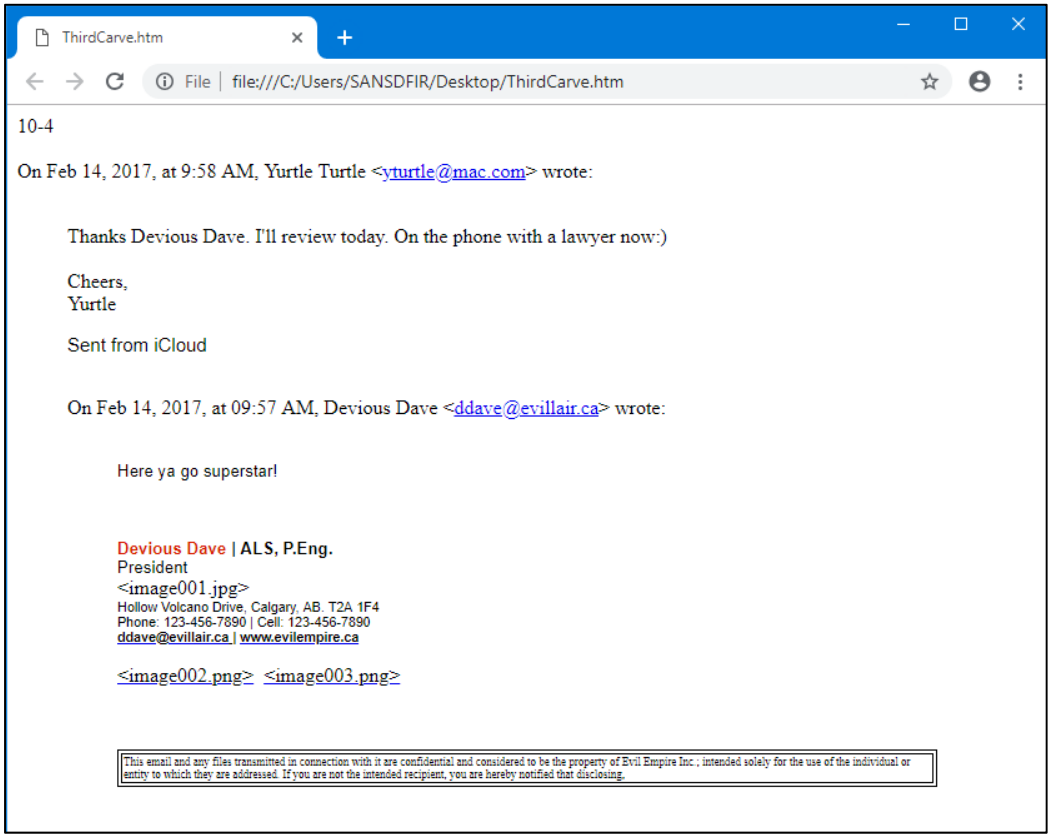



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
05EC43FC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
05EC43FD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
05EC43FE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
05EC43FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
05EC44000  3C 68 74 6D 6C 3E 0D 0A 3C 68 65 61 64 3E 0D 0A  <html>..<head>..
05EC44010  3C 6D 65 74 61 20 68 74 74 70 2D 65 71 75 69 76  <meta http-equiv
05EC44020  3D 22 43 6F 6E 74 65 6E 74 2D 54 79 70 65 22 20  ="Content-Type"
05EC44030                                    74 2F 68 74  content="text/ht
05EC44040                                    75 74 66 2D  ml; charset=utf-
05EC44050                                    0D 0A 3C 62  8">..</head>..<b
05EC44060                                       3E 0D     ody dir="auto">.
05EC44070  0A 3C 64 69 76 3E 3C 2F 64 69 76 3E 0D 0A 3C 64  .<div></div>..<d
05EC44080  69 76 3E 31 30 2D 34 3C 2F 64 69 76 3E 0D 0A 3C  iv>10-4</div>..<
05EC44090  64 69 76 3E 3C 62 72 3E 0D 0A 4F 6E 20 46 65 62  div><br>..On Feb
05EC440A0  20 31 34 2C 20 32 30 31 37 2C 20 61 74 20 39 3A   14, 2017, at 9:
05EC440B0  35 38 20 41 4D 2C 20 59 75 72 74 6C 65 20 54 75  58 AM, Yurtle Tu
05EC440C0  72 74 6C 65 20 26 6C 74 3B 3C 61 20 68 72 65 66  rtle &lt;<a href
05EC440D0  3D 22 6D 61 69 6C 74 6F 3A 79 74 75 72 74 6C 65  ="mailto:yturtle
05EC440E0  40 6D 61 63 2E 63 6F 6D 22 3E 79 74 75 72 74 6C  @mac.com">yturtl
05EC440F0  65 40 6D 61 63 2E 63 6F 6D 3C 2F 61 3E 26 67 74  e@mac.com</a>&gt
05EC44100  3B 20 77 72 6F 74 65 3A 3C 62 72 3E 0D 0A 3C 62  ; wrote:<br>..<b
05EC44110  72 3E 0D 0A 3C 2F 64 69 76 3E 0D 0A 3C 62 6C 6F  r>..</div>..<blo
05EC44120  63 6B 71 75 6F 74 65 20 74 79 70 65 3D 22 63 69  ckquote type="ci
05EC44130  74 65 22 3E 0D 0A 3C 64 69 76 3E 0D 0A 3C 64 69  te">..<div>..<di
05EC44140  76 3E 54 68 61 6E 6B 73 20 44 65 76 69 6F 75 73  v>Thanks Devious
05EC44150  20 44 61 76 65 2E 20 49 27 6C 6C 20 72 65 76 69   Dave. I'll revi
05EC44160  65 77 20 74 6F 64 61 79 2E 20 4F 6E 20 74 68 65  ew today. On the
05EC44170  20 70 68 6F 6E 65 20 77 69 74 68 20 61 20 6C 61   phone with a la
05EC44180  77 79 65 72 20 6E 6F 77 3A 29 3C 2F 64 69 76 3E  wyer now:)</div>
05EC44190  0D 0A 3C 64 69 76 3E 3C 62 72 20 64 61 74 61 2D  ..<div><br data-
05EC441A0  6D 63 65 2D 62 6F 67 75 73 3D 22 31 22 3E 0D 0A  mce-bogus="1">..
```

7. Start sweeping from the indicated location downwards. Keep sweeping until you reach the location indicated below. Even though the entire file you are trying to carve is present, you are only carving a portion of it to show that at least something will display.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
05EC458D0	63	6F	6E	6E	65	63	74	69	6F	6E	20	77	69	74	68	20	connection with
05EC458E0	69	74	20	61	72	65	20	63	6F	6E	66	69	64	65	6E	74	it are confident
05EC458F0	69	61	6C	20	61	6E	64	20	63	6F	6E	73	69	64	65	72	ial and consider
05EC45900	65	64	20	74	6F	20	62	65	20	74	68	65	20	70	72	6F	ed to be the pro
05EC45910	70	65	72	74	79	20	6F	66	20	45	76	69	6C	20	45	6D	perty of Evil Em
05EC45920	70	69	72	65	20	49	6E	63	2E	3B	20	69	6E	74	65	6E	pire Inc.; inten
05EC45930	64	65	64	20	73	6F	6C	65	6C	79	20	66	6F	72	20	74	ded solely for t
05EC45940	68	65	20	75	73	65	20	6F	66	20	74	68	65	20	69	6E	he use of the in
05EC45950	64	69	76	69	64	75	61	6C	20	6F	72	20	65	6E	74	69	dividual or enti
05EC45960	74	79	20	74	6F	20	77	68	69	63	68	20	74	68	65	79	ty to which they
05EC45970	20	61	72	65	20	61	64	64	72	65	73	73	65	64	2E	20	are addressed.
05EC45980	49	66	20	79	6F	75	20	61	72	65	20	6E	6F	74	20	74	If you are not t
05EC45990	68	65	20	69	6E	74	65	6E	64	65	64	20	72	65	63	69	he intended reci
05EC459A0	70	69	65	6E	74	2C	20	79	6F	75	20	61	72	65	20	68	pient, you are h
05EC459B0	65	72	65	62	79	20	6E	6F	74	69	66	69	65	64	20	74	ereby notified t
05EC459C0	68	61	74	20	64	69	73	63	6C	6F	73	69	6E	67	2C	20	hat disclosing,
05EC459D0	63	6F	70	79	69	6E	67	2C	20	6D	6F	64	69	66	79	69	copying, modifyi
05EC459E0	6E	67	2C	20	64	69	73	74	72	69	62	75	74	69	6E	67	ng, distributing
05EC459F0	20	6F	72	20	74	61	6B	69	6E	67	20	61	6E	79	20	61	or taking any a
05EC45A00	63	74	69	6F	6E	20	69	6E	20	72	65	6C	69	61	6E	63	ction in relianc

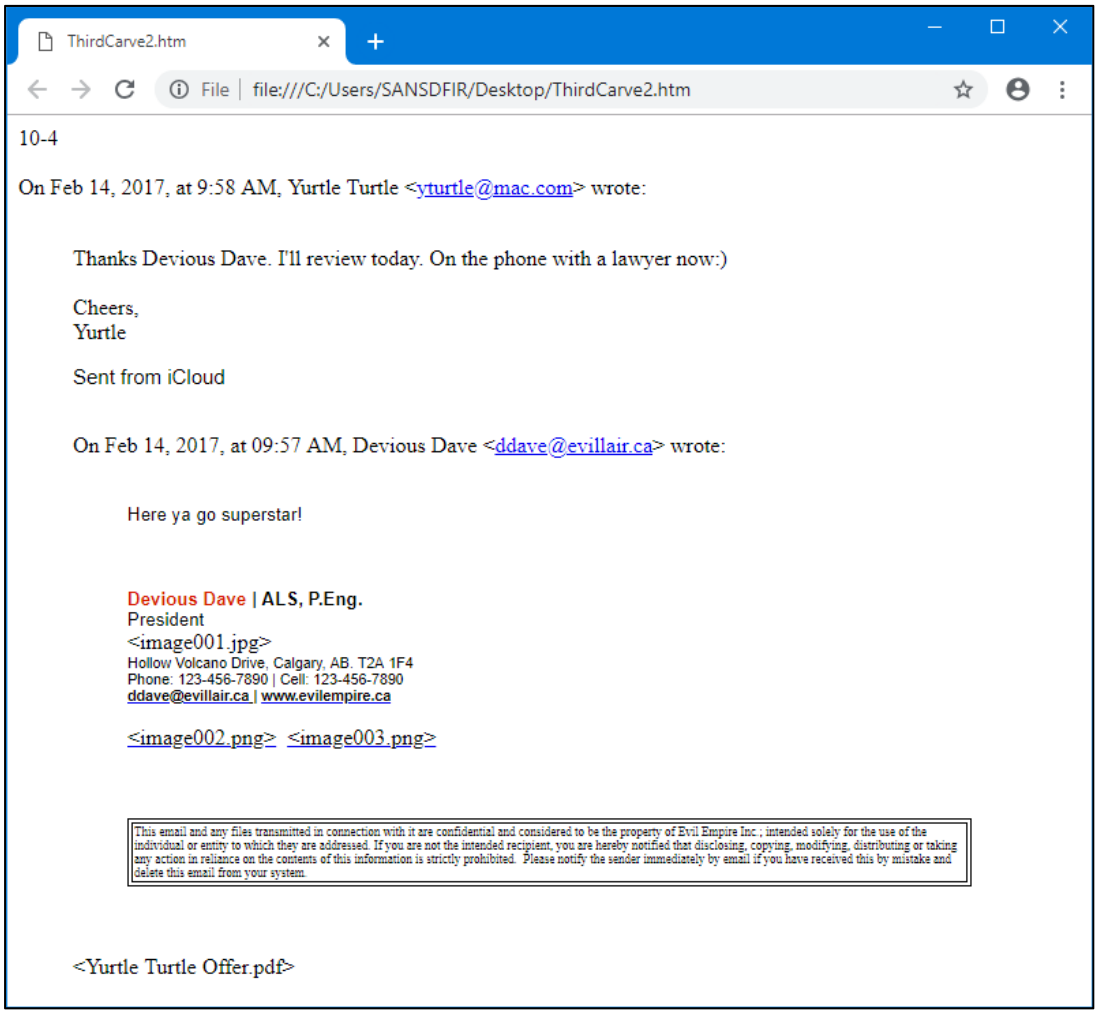
8. Using the same process as before, save this out to your **Desktop**, naming it **`ThirdCarve.htm`**. We decided to give it the file extension of **.htm** because our experience told us from viewing the data in the hex editor, that it was a part of a webpage. If you are ever unsure, simply save it out as a .**txt** file, and then you can test various different extensions from there. Navigate to the **Desktop** and open the file you just carved.



9. Even though you left the beginning of the file and a lot of the end of the file behind, the portion you carved still opened.

Exercise Questions

1. Go back to the hex editor, find the beginning and end of the file you just carved, carve out the entire .html file and save it out. Compare it to the first carve you performed, and answer the questions.



   a. Is there anything more in this complete portion?

      _____

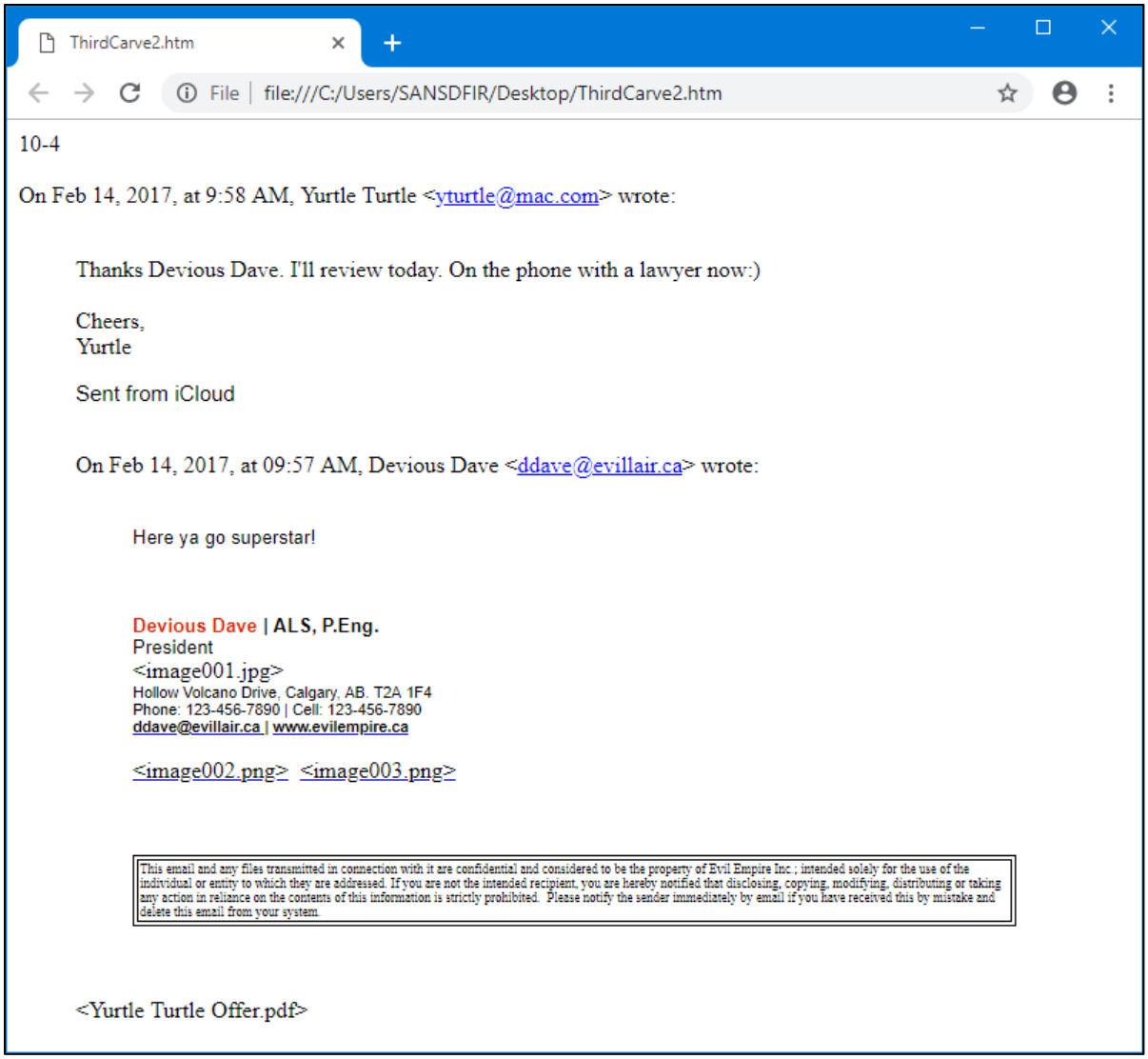
   b. If yes, what?

      _____

2. You can now close **HxD** and **Arsenal Image Mounter**, as well as any other files you had open.

Exercise Questions - Step by Step

1. Go back to the hex editor, find the beginning and end of the file you just carved, and carve out the entire .html file and save it out. Compare it to the first carve you did and answer the questions.



a. Is there anything more in this complete portion?

Yes

b. If yes, what?

Besides the rest of the disclaimer in the footer, we see that there was an attachment on the original email called Yurtle Turtle Offer.pdf.

We now potentially have another data point to pursue.

Exercise - A Photo from Nothing

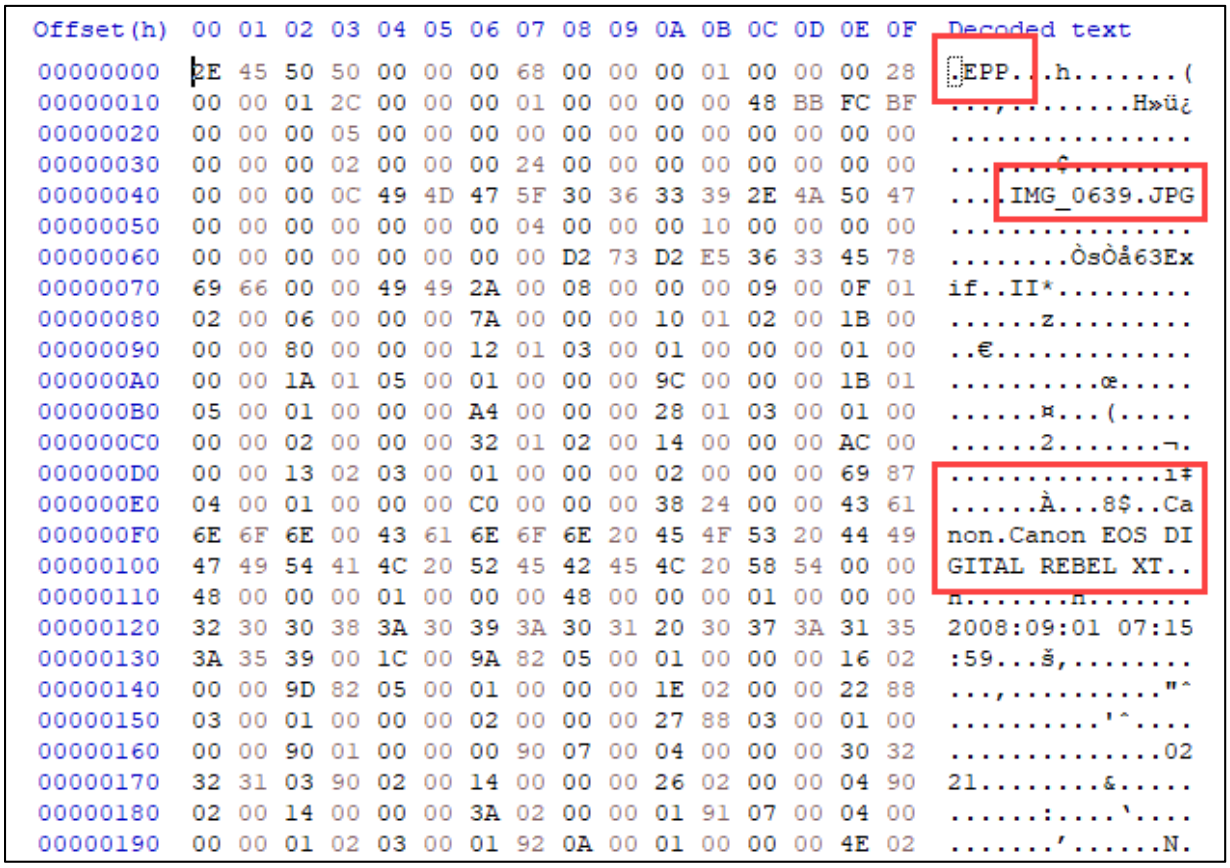
1. **BEFORE DOING ANTHING WITH ANY INDIVIDUAL FILES, IT WOULD BE BEST TO CREATE A COPY AND WORK ON THE COPY. YOU ARE MAKING CHANGES TO THESE FILES THAT YOU MAY NOT BE ABLE TO REVERSE**. As a last resort, you can always re-extract an original copy from your SANS provided USB drive.

2. From your **VM**, navigate to **C:\Cases\Exercises\6.3** and locate a file named **IMG_0639.epp**, and double click to open it. You will be asked what program you want to use to open the file, and you are welcome to try **Paint**, or **Internet Explorer**. Neither of them will open the file, so this is an expected result.

3. Looking at the file name though, it follows the nomenclature for a photo. Let's take a closer look.

4. Open **HxD**.

5. Click on **File → Open**, navigate to **C:\Cases\Exercises\6.3\IMG_0639.epp** and select it to open. There are certainly indicators that this file is a photo or picture, but looking at the first few bytes, or file header, it shows .**EPP**. This is certainly not a common format of a photo that opens with common tools.



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	2E	45	50	50	00	00	00	68	00	00	00	01	00	00	00	28	.EPP...h.......(
00000010	00	00	01	2C	00	00	00	01	00	00	00	00	48	BB	FC	BF	...,........H»ü¿
00000020	00	00	00	05	00	00	00	00	00	00	00	00	00	00	00	00	....
00000030	00	00	00	02	00	00	00	24	00	00	00	00	00	00	00	00	........$........
00000040	00	00	00	0C	49	4D	47	5F	30	36	33	39	2E	4A	50	47	....IMG_0639.JPG
00000050	00	00	00	00	00	00	00	04	00	00	00	10	00	00	00	00	....
00000060	00	00	00	00	00	00	00	00	D2	73	D2	E5	36	33	45	78	........ÒsÒå63Ex
00000070	69	66	00	00	49	49	2A	00	08	00	00	00	09	00	0F	01	if..II*.........
00000080	02	00	06	00	00	00	7A	00	00	00	10	01	02	00	1B	00	......z.........
00000090	00	00	80	00	00	00	12	01	03	00	01	00	00	00	01	00	..€.............
000000A0	00	00	1A	01	05	00	01	00	00	00	9C	00	00	00	1B	01	..........œ.....
000000B0	05	00	01	00	00	00	A4	00	00	00	28	01	03	00	01	00	......¤...(.....
000000C0	00	00	02	00	00	00	32	01	02	00	14	00	00	00	AC	00	......2.......¬.
000000D0	00	00	13	02	03	00	01	00	00	00	02	00	00	00	69	87	..............i‡
000000E0	04	00	01	00	00	00	C0	00	00	00	38	24	00	00	43	61	......À...8$..Ca
000000F0	6E	6F	6E	00	43	61	6E	6F	6E	20	45	4F	53	20	44	49	non.Canon EOS DI
00000100	47	49	54	41	4C	20	52	45	42	45	4C	20	58	54	00	00	GITAL REBEL XT..
00000110	48	00	00	00	01	00	00	00	48	00	00	00	01	00	00	00	H.......H.......
00000120	32	30	30	38	3A	30	39	3A	30	31	20	30	37	3A	31	35	2008:09:01 07:15
00000130	3A	35	39	00	1C	00	9A	82	05	00	01	00	00	00	16	02	:59...š,........
00000140	00	00	9D	82	05	00	01	00	00	00	1E	02	00	00	22	88	...,..........."^
00000150	03	00	01	00	00	00	02	00	00	00	27	88	03	00	01	00	..........'^....
00000160	00	00	90	01	00	00	00	90	07	00	04	00	00	00	30	32	02
00000170	32	31	03	90	02	00	14	00	00	00	26	02	00	00	04	90	21........&.....
00000180	02	00	14	00	00	00	3A	02	00	00	01	91	07	00	04	00	......:....'....
00000190	00	00	01	02	03	00	01	92	0A	00	01	00	00	00	4E	02	.......'......N.

6. But maybe there is something we can do about it. With most photos taken by SLR cameras, they have a proprietary format, but there is still a .**JPG** in there somewhere. Typically, the camera is merely adding proprietary data in front of the photo information.

7. Scroll down the data until you find a **.JPG** file signature. (**HINT**: use the **Offset** numbers in the left column of the screenshot to help you find the header) Remember that it looks like this: **ÿØÿà**. Once you find it, start sweeping the data.

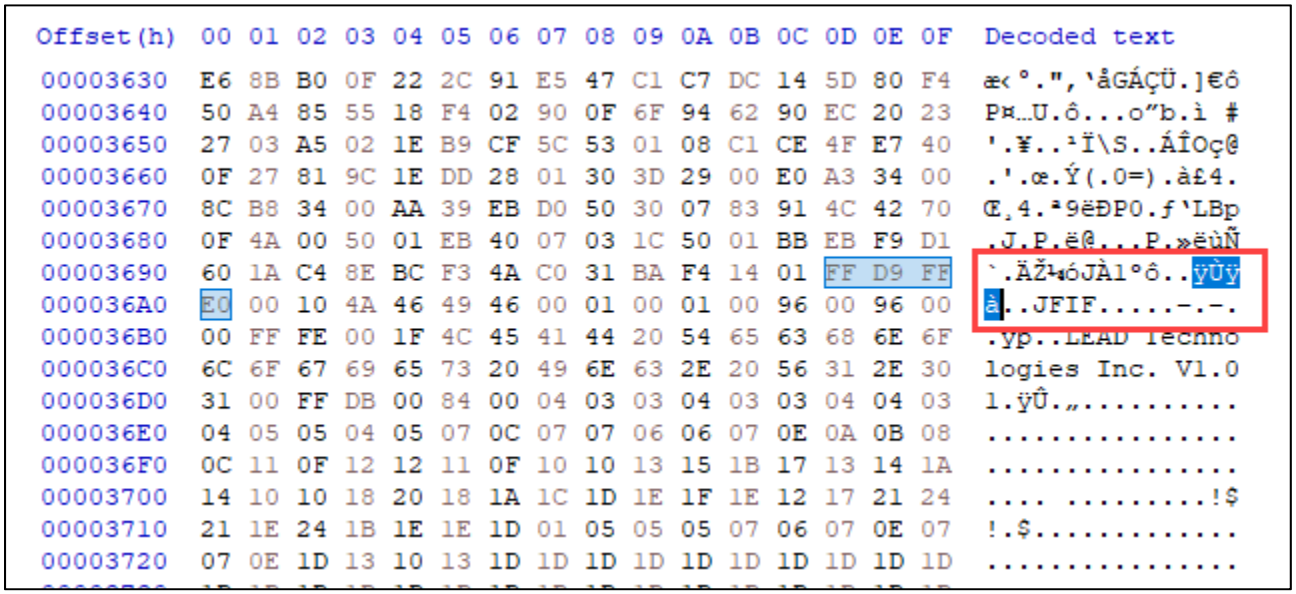
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00002490	01	00	02	00	04	00	00	00	52	39	38	00	02	00	07	00	........R98.....
000024A0	04	00	00	00	30	31	30	30	00	00	00	00	06	00	03	01	....0100........
000024B0	03	00	01	00	00	00	06	00	00	00	1A	01	05	00	01	00	....
000024C0	00	00	86	24	00	00	1B	01	05	00	01	00	00	00	8E	24	..†$..........Ž$
000024D0	00	00	28	01	03	00	01	00	00	00	02	00	00	00	01	02	..(.............
000024E0	04	00	01	00	00	00	96	24	00	00	02	02	04	00	01	00	.......–$........
000024F0	00	00	95	11	00	00	00	00	00	00	48	00	00	00	01	00	..•.......H.....
00002500	00	00	48	00	00	00	01	00	00	00	FF	D8	FF	E0	00	10	..H.......ÿØÿà..
00002510	4A	46	49	46	00	01	00	01	00	96	00	96	00	00	FF	FE	JFIF.....–.–..ÿþ
00002520	00	1F	4C	45	41	44	20	54	65	63	68	6E	6F	6C	6F	67	..LEAD Technolog
00002530	69	65	73	20	49	6E	63	2E	20	56	31	2E	30	31	00	FF	ies Inc. V1.01.ÿ
00002540	DB	00	84	00	10	0B	0C	0E	0C	0A	10	0E	0D	0E	12	11	Û.„..............
00002550	10	13	18	28	1A	18	16	16	18	31	23	25	1D	28	3A	33	...(.....1#%.(:3
00002560	3D	3C	39	33	38	37	40	48	5C	4E	40	44	57	45	37	38	=<9387@H\N@DWE78
00002570	50	6D	51	57	5F	62	67	68	67	3E	4D	71	79	70	64	78	PmQW_bghg>Mqypdx
00002580	5C	65	67	63	01	11	12	12	18	15	18	2F	1A	1A	2F	63	\egc......../../c
00002590	42	38	42	63	63	63	63	63	63	63	63	63	63	63	63	63	B8Bccccccccccccc
000025A0	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	cccccccccccccccc

8. Continue sweeping data until you find a relatively obvious file footer (**ÿÛ**) and stop sweeping.

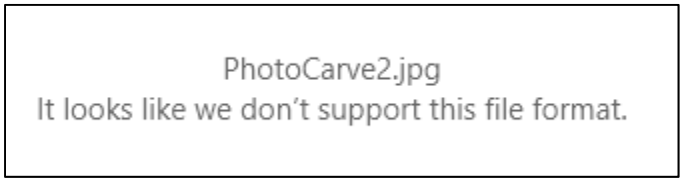
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00003630	E6	8B	B0	0F	22	2C	91	E5	47	C1	C7	DC	14	5D	80	F4	æ‹°."‚'åGÁÇÜ.]€ô
00003640	50	A4	85	55	18	F4	02	90	0F	6F	94	62	90	EC	20	23	P¤…U.ô...o"b.ì #
00003650	27	03	A5	02	1E	B9	CF	5C	53	01	08	C1	CE	4F	E7	40	'.¥..¹Ï\S..ÁÎOç@
00003660	0F	27	81	9C	1E	DD	28	01	30	3D	29	00	E0	A3	34	00	.'.œ.Ý(.0=).à£4.
00003670	8C	B8	34	00	AA	39	EB	D0	50	30	07	83	91	4C	42	70	Œ¸4.ª9ëÐP0.ƒ'LBp
00003680	0F	4A	00	50	01	EB	40	07	03	1C	50	01	BB	EB	F9	D1	.J.P.ë@...P.»ëùÑ
00003690	60	1A	C4	8E	BC	F3	4A	C0	31	BA	F4	14	01	FF	D9	FF	`.ÄŽ¼óJÀ1ºô..ÿÙÿ
000036A0	E0	00	10	4A	46	49	46	00	01	00	01	00	96	00	96	00	à..JFIF.....–.–.
000036B0	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	68	6E	6F	.ÿþ..LEAD Techno
000036C0	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	31	2E	30	logies Inc. V1.0
000036D0	31	00	FF	DB	00	84	00	04	03	03	04	03	03	04	04	03	1.ÿÛ.„..........
000036E0	04	05	05	04	05	07	0C	07	07	06	06	07	0E	0A	0B	08	....
000036F0	0C	11	0F	12	12	11	0F	10	10	13	15	1B	17	13	14	1A	....
00003700	14	10	10	18	20	18	1A	1C	1D	1E	1F	1E	12	17	21	24	....!$

9. Using the technique from exercise **6.3A**, press **Ctrl+C** to copy your highlighted data, and save it out (through the steps inside **HxD**) to your **Desktop**, naming it **PhotoCarve.jpg**. Once done, open the file.

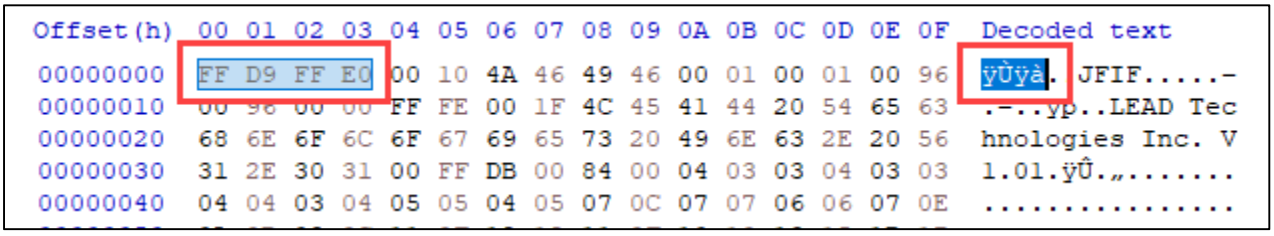
10. Now the picture opens. But it sure is tiny. In fact, this is just the thumbnail that you have recovered. Let's dive back in and see what else we can find, because looking at the amount of data in the hex editor, there certainly seems to be more. Back in your hex editor, return to the .**EPP** file you opened, and look carefully around where the thumbnail code ended. It looks quite like the beginning of another picture!



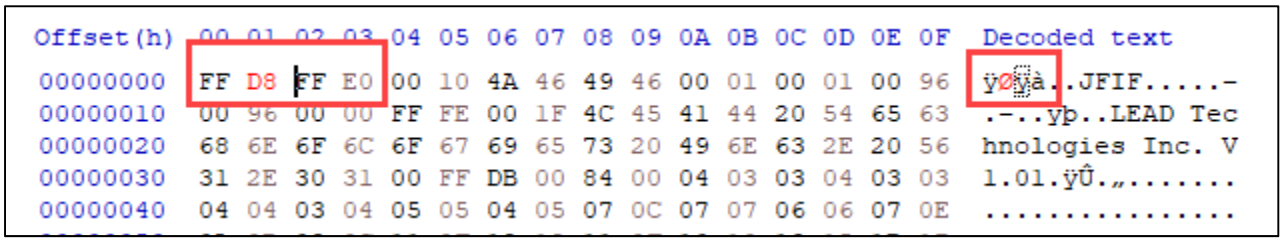
11. Start sweeping from there and sweep until you find the next .**JPG** file footer. This will take some time, as the sweep is long! You need to figure out how to perform a **page down** while holding the **Shift** key and sweeping on your computer, as this will make things go much quicker. Some keyboards have a **page down** key, and on some keyboards, holding the **Fn** key and down arrow will do **page down**. Hint: you will be scrolling to the end of the file. Once you find the end, **Ctrl+C** to copy the content, and then save it into its own data space in **HxD**, and then save it out to your desktop as **PhotoCarve2.jpg**. Navigate to your desktop and open the file. It shouldn't open. WHAT?



12. Go back to **HxD**, and on the file that you just created, scroll to the very top of it, and look at the file header. It looks really close to the file header for a .**JPG**, but close isn't good enough. We see in the hex a signature of **FF D9 FF**, but we know from our training module that the file header for a .**JPG** is **FF D8 FF**. We need to change this.



13. Click your cursor just to the left of **D9** in the hex field, and type ***D8***. You should see the change represented in red.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	00	01	00	96	ÿØÿà..JFIF.....–
00000010	00	96	00	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	.–..ÿþ..LEAD Tec
00000020	68	6E	6F	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	hnologies Inc. V
00000030	31	2E	30	31	00	FF	DB	00	84	00	04	03	03	04	03	03	1.01.ÿÛ.„.....
00000040	04	04	03	04	05	05	04	05	07	0C	07	07	06	06	07	0E	.....

14. Click to save the changes. Then click on **File → Save As** and save this out to your **Desktop** as **PhotoCarve3.jpg**. Navigate to your **Desktop** and try to open it.

15. SUCCESS!!!

Exercise—Key Takeaways

- Data carving and rebuilding can be very time consuming and frustrating.
- What is believed to be an entire file may not be.
- Data is often available, even if it is not in a format that the examiner is familiar with.
- A forensicator could spend hours trying to repair a file and have no success.

This page intentionally left blank.

Exercise 6.3C—Data Carving & Rebuilding – DOC & DOCX

Exercise Objectives

- Recognize various common file signatures
- Understand OLECF and OpenXML Office formats
- Rebuild a partially overwritten file
- Repair corrupted or overwritten files

Exercise Preparation

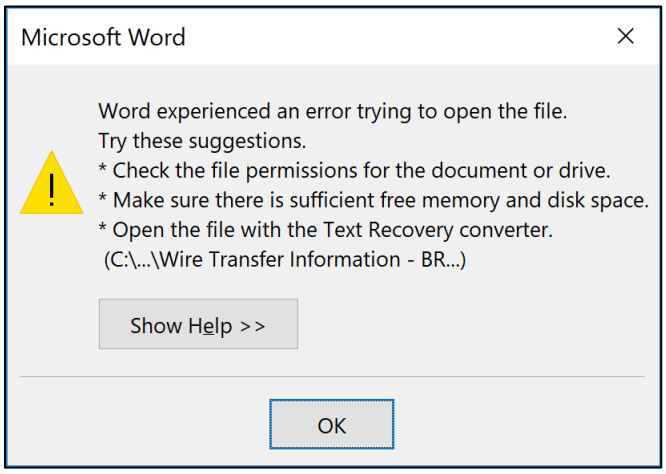
1. Boot your **FOR498 Windows VM**

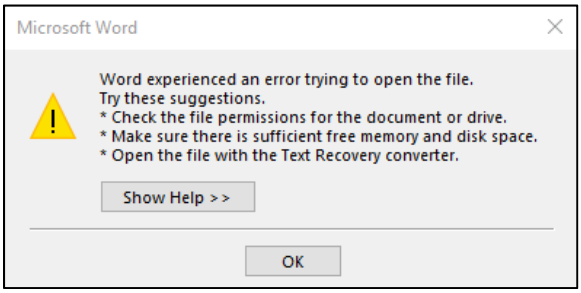
2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. **BEFORE DOING ANTHING WITH ANY INDIVIDUAL FILES, IT WOULD BE BEST TO CREATE A COPY AND WORK ON THE COPY. YOU ARE MAKING CHANGES TO THESE FILES THAT YOU MAY NOT BE ABLE TO REVERSE**. As a last resort, you can always re-extract an original copy from your SANS provided USB drive.

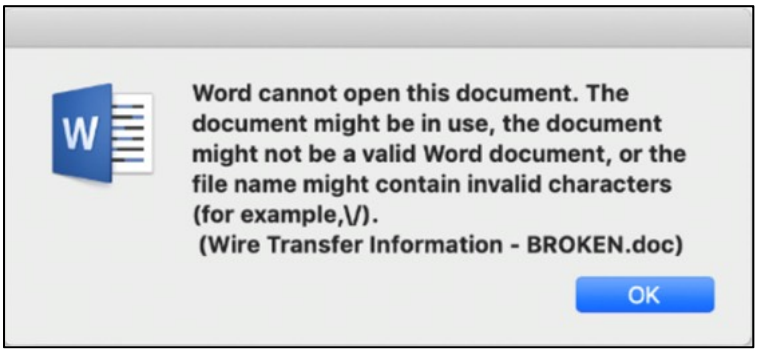
Exercise - Recover/Repair a .DOC File

1. From your **VM**, navigate to **C:\Cases\Exercises\6.3** and locate a file named **Wire Transfer Information – BROKEN.doc**, and copy it to your **host Desktop**. Once copied over, double click on it to open it. You should get the same or similar error messages as you see here. Click **OK** to close out of it.



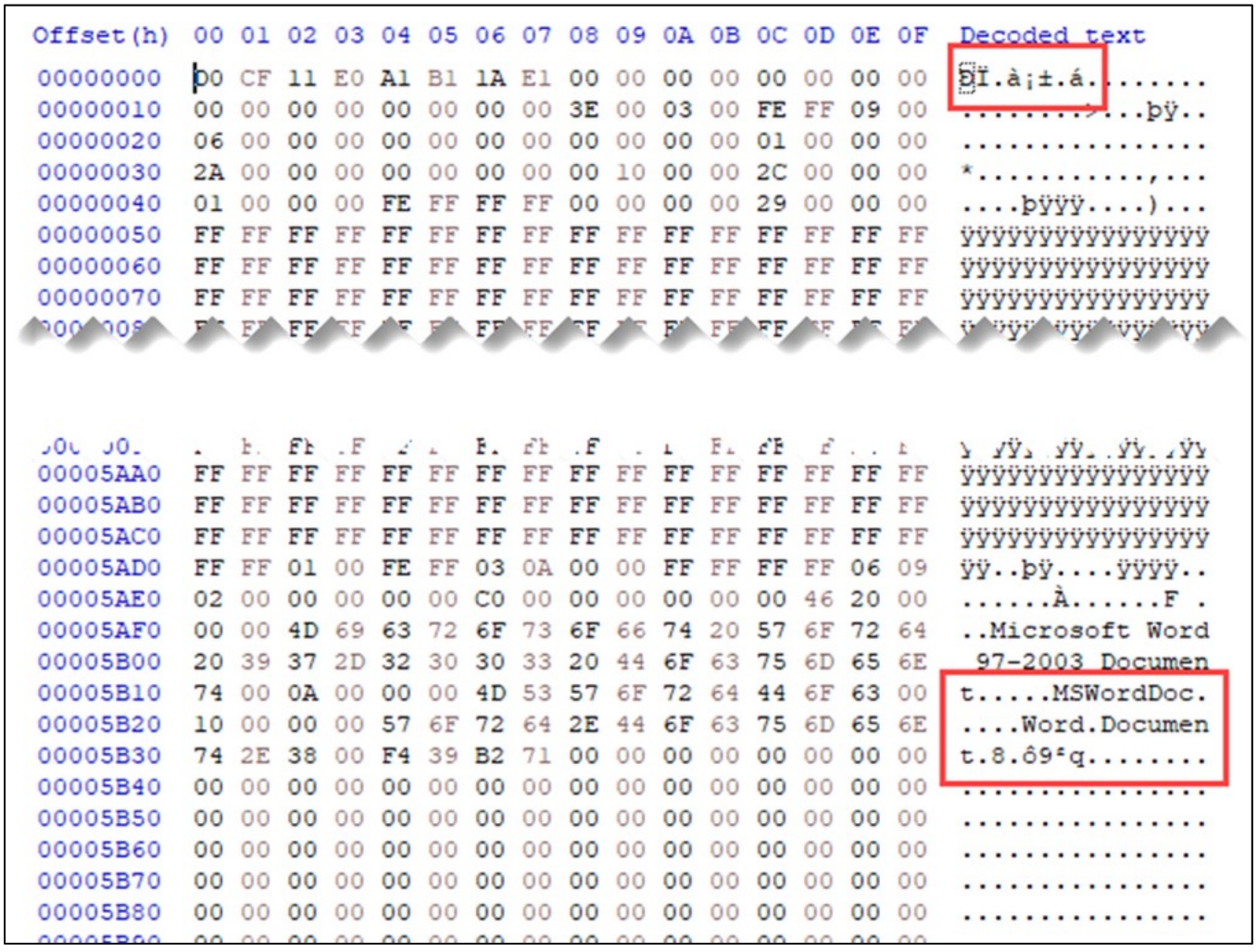


2. If you are using Office for Mac, your error message may look like this.

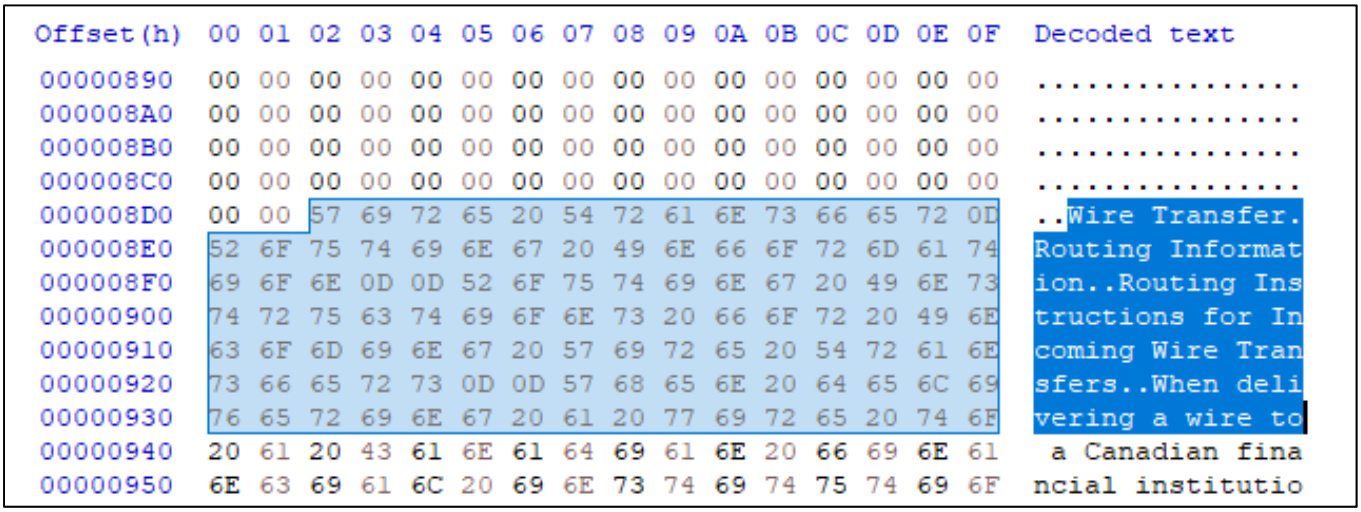


3. Back in your **VM**, open **HxD**.

4. Click on **File → Open** and navigate to **C:\Cases\ Exercises \6.3\Wire Transfer Information – BROKEN.doc** and select it to open. Note that the header looks correct for a .**DOC**, as does the footer, if you scroll to the bottom and look.

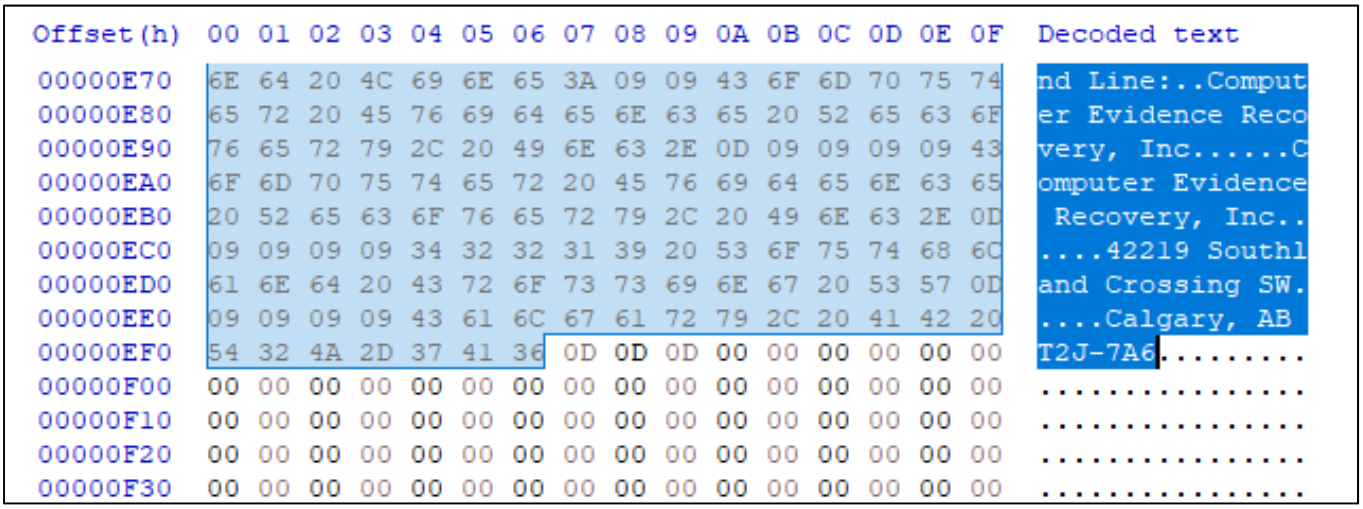


5. While you were scrolling down to check the file footer, you would have seen some plain text. Since, this file has a proper header and a proper footer, but still won't open, we have to surmise that somewhere in the middle of this deleted file, in one of the sectors, data from somewhere else has corrupted it. Our only choice is to carve the useful data out of it. Find the beginning of the useful plain text.



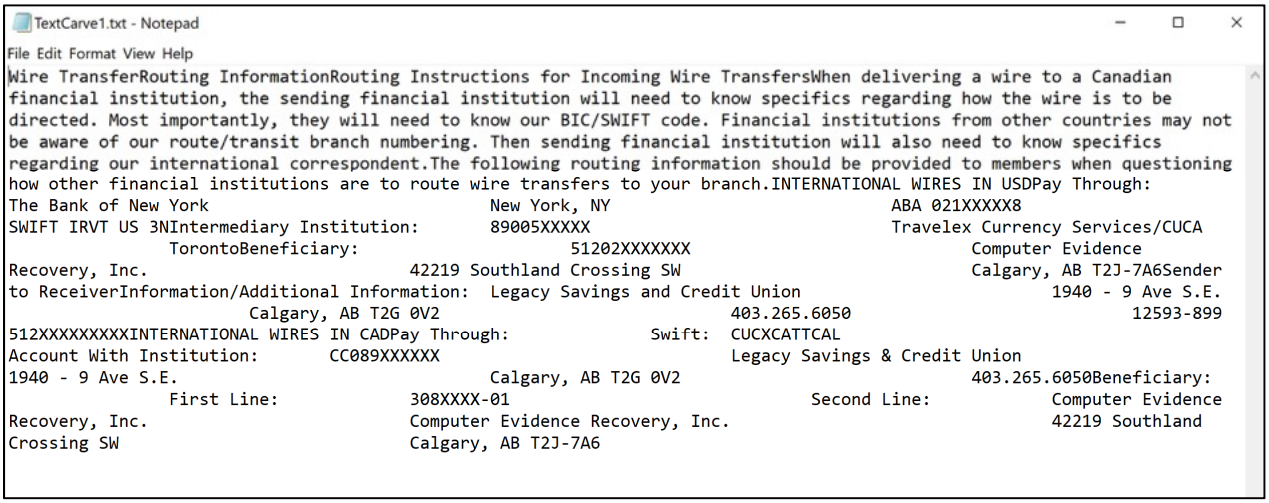
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000890	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008D0	00	00	57	69	72	65	20	54	72	61	6E	73	66	65	72	0D	..Wire Transfer.
000008E0	52	6F	75	74	69	6E	67	20	49	6E	66	6F	72	6D	61	74	Routing Informat
000008F0	69	6F	6E	0D	0D	52	6F	75	74	69	6E	67	20	49	6E	73	ion..Routing Ins
00000900	74	72	75	63	74	69	6F	6E	73	20	66	6F	72	20	49	6E	tructions for In
00000910	63	6F	6D	69	6E	67	20	57	69	72	65	20	54	72	61	6E	coming Wire Tran
00000920	73	66	65	72	73	0D	0D	57	68	65	6E	20	64	65	6C	69	sfers..When deli
00000930	76	65	72	69	6E	67	20	61	20	77	69	72	65	20	74	6F	vering a wire to
00000940	20	61	20	43	61	6E	61	64	69	61	6E	20	66	69	6E	61	a Canadian fina
00000950	6E	63	69	61	6C	20	69	6E	73	74	69	74	75	74	69	6F	ncial institutio

6. Sweep data until you find the end of the useful plain text.

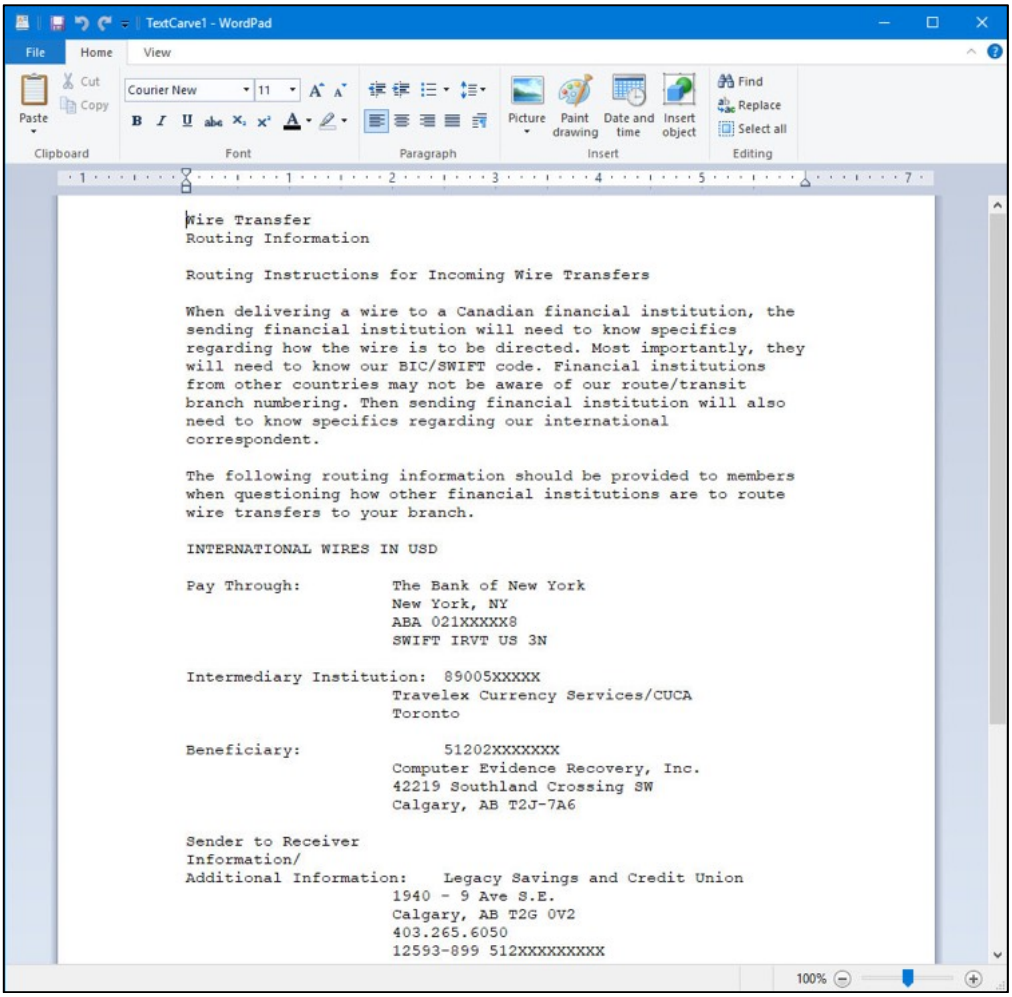


Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000E70	6E	64	20	4C	69	6E	65	3A	09	09	43	6F	6D	70	75	74	nd Line:..Comput
00000E80	65	72	20	45	76	69	64	65	6E	63	65	20	52	65	63	6F	er Evidence Reco
00000E90	76	65	72	79	2C	20	49	6E	63	2E	0D	09	09	09	09	43	very, Inc.....C
00000EA0	6F	6D	70	75	74	65	72	20	45	76	69	64	65	6E	63	65	omputer Evidence
00000EB0	20	52	65	63	6F	76	65	72	79	2C	20	49	6E	63	2E	0D	Recovery, Inc..
00000EC0	09	09	09	09	34	32	32	31	39	20	53	6F	75	74	68	6C	....42219 Southl
00000ED0	61	6E	64	20	43	72	6F	73	73	69	6E	67	20	53	57	0D	and Crossing SW.
00000EE0	09	09	09	09	43	61	6C	67	61	72	79	2C	20	41	42	20	....Calgary, AB
00000EF0	54	32	4A	2D	37	41	36	0D	0D	0D	00	00	00	00	00	00	T2J-7A6.....
00000F00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000F10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000F20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000F30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

7. Use **Ctrl+C** to copy the data, then using previous techniques, paste it to a new data space in **HxD**, and then save it out to your **Desktop** as **TextCarve1.txt**. Navigate to your **Desktop** and open the file with **Notepad**. Again, unless you are using the newest Windows 10 (as your **VM** is), you will see the output below.



8. Better than nothing, but we are professionals, and we can do better. Close the text file, and then right-click on the file and open the file using WordPad.

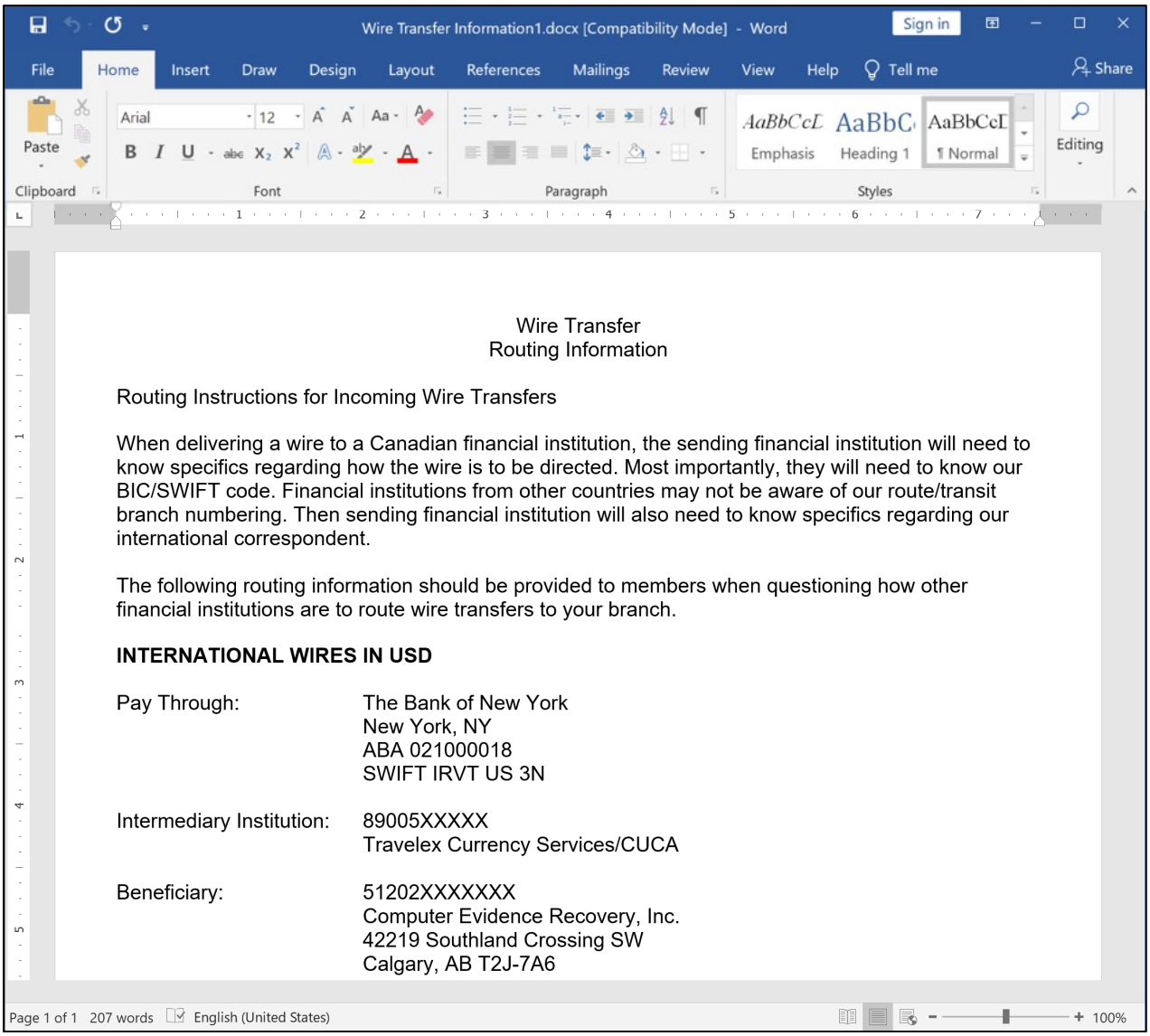


9. Once you see that the file opens nicely, and has formatting, you can close it, go back to the file, and rename it with either a .**RTF** extension, or a .**DOC** extension. This way, when you give it to the client, it will open in the nicer looking way!

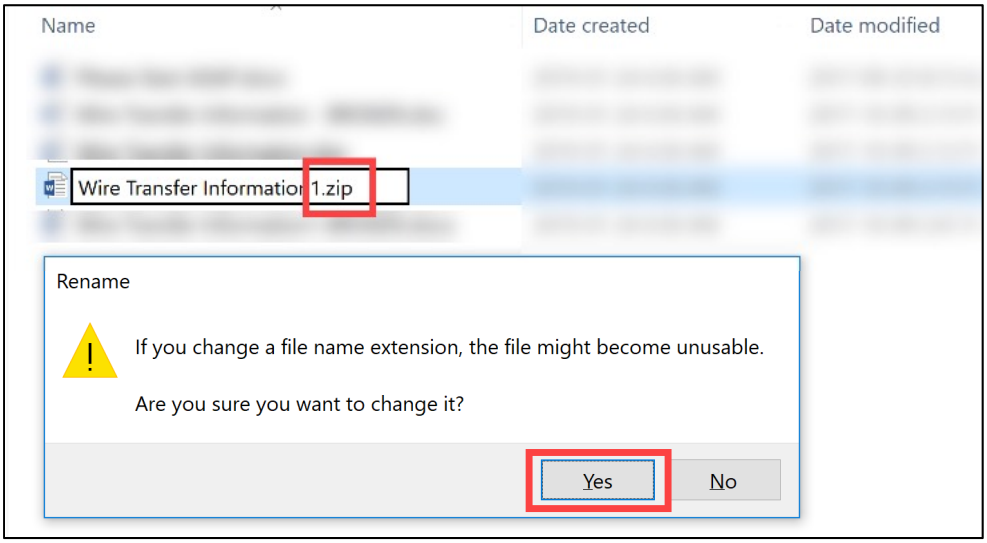
Exercise – Recover/Repair a .DOCX File – Method 1

1. Close out of **HxD** (if it is still open).

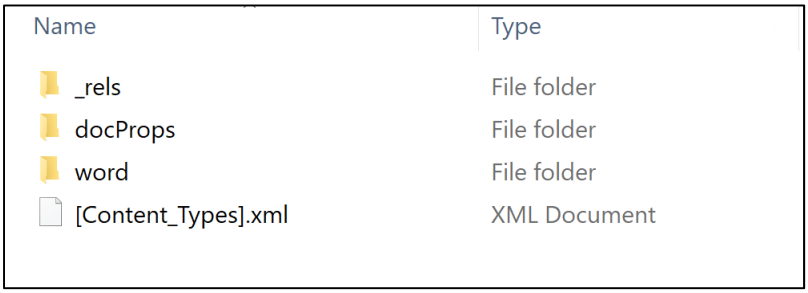
2. From your **VM**, navigate to **C:\Cases\Exercises\6.3** and locate a file named **Wire Transfer Information1.docx**, and copy it out to your host desktop. Go to your host desktop and double click to open the file you just copied there. It opens as it should.



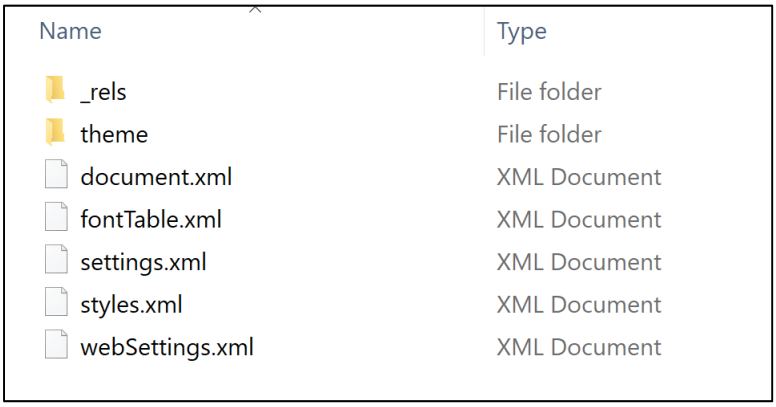
3. Now close the file, go back to your **VM**, and navigate to **C:\Cases\Exercises\6.3** again. Right click on the file **Wire Transfer Information1.docx**, select **Rename**, and change the file extension from **.docx** to .**zip**.



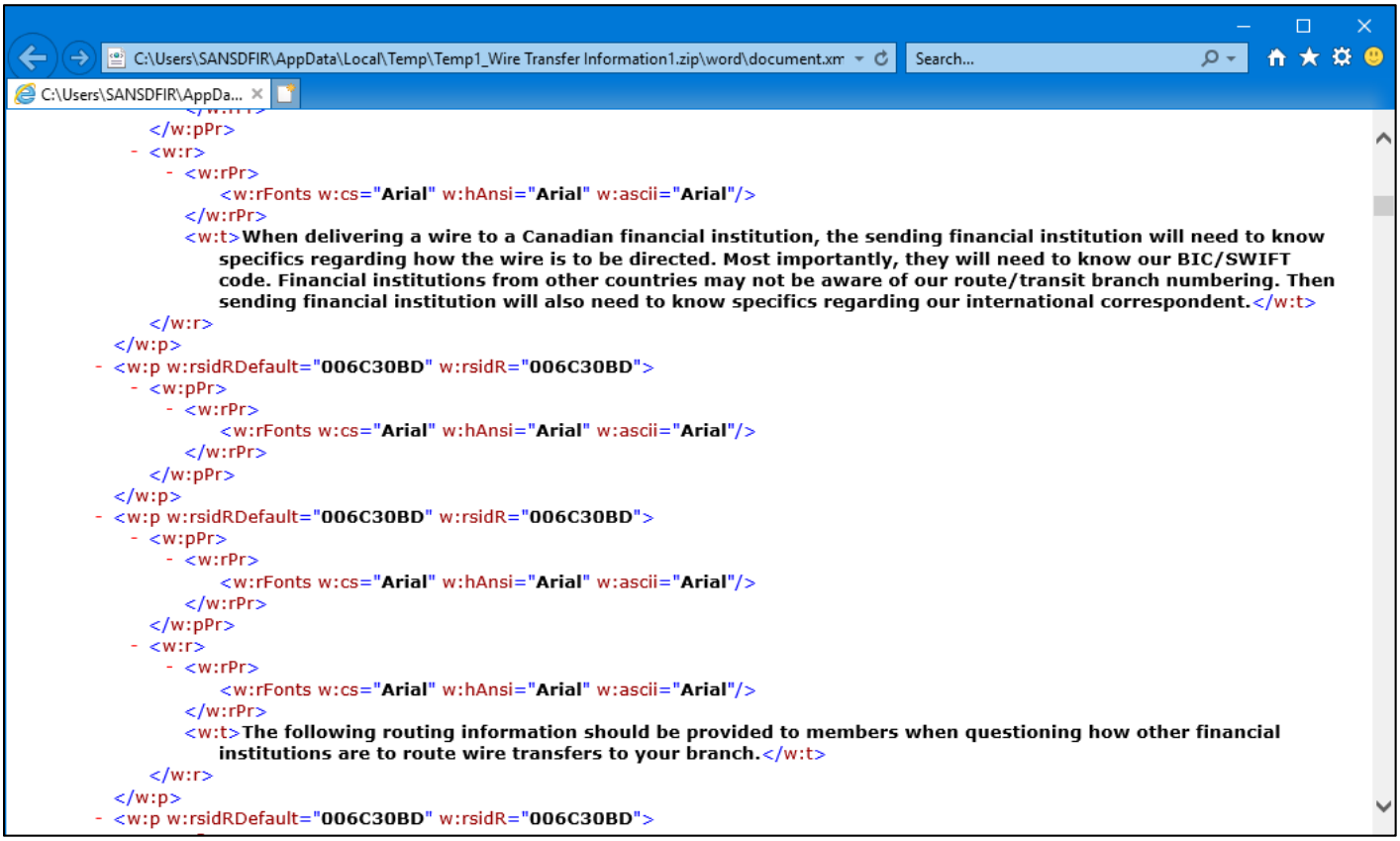
4. The icon has now changed to a **zip** file icon. Double click to open it and click into the folders to see what is there.



5. Open the **word** folder.

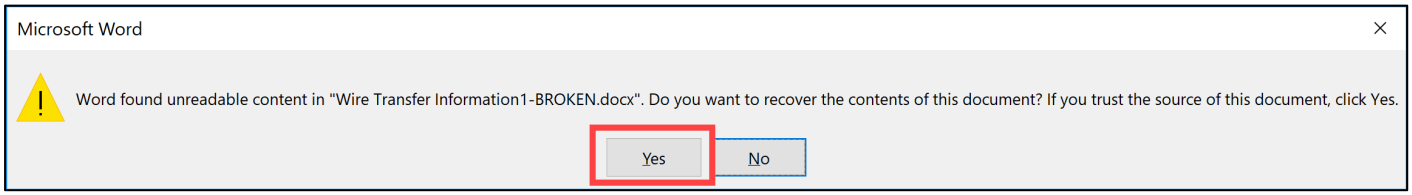


6. Open the **document.xml** file. If you are prompted for a program to open it with, simply use any browser, such as **Internet Explorer**. Scroll down a bit and you can clearly see the content of the document.

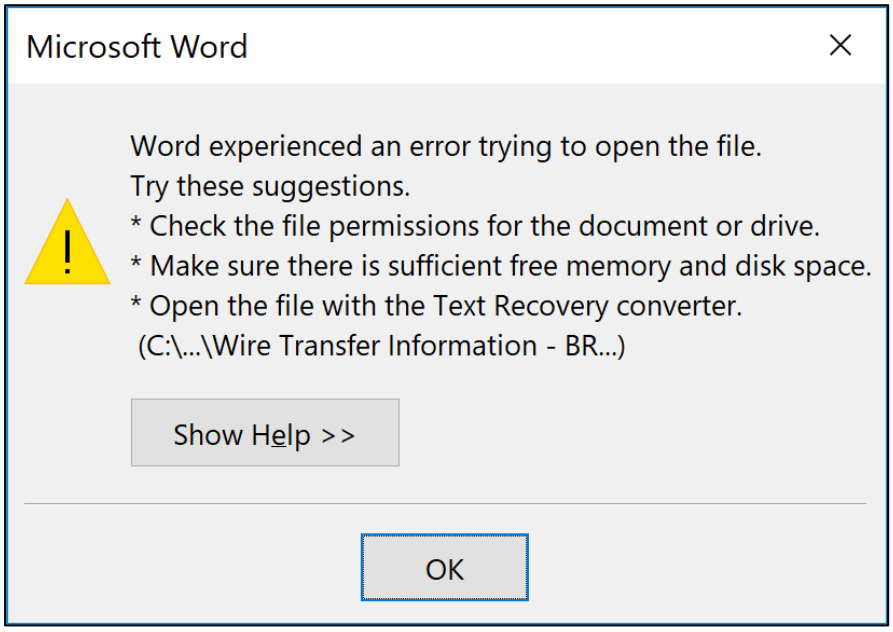


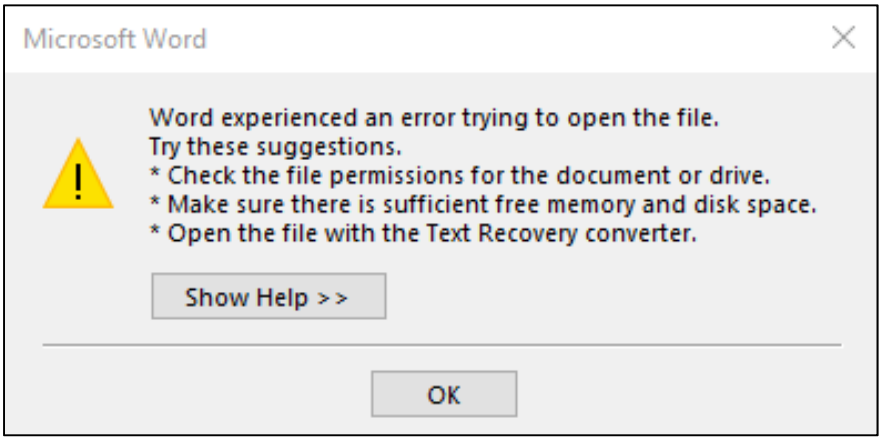
7. Close this file, and any folders, and find your way back to **C:\Cases\Exercises\6.3**. Right click on the **Wire Transfer Information1.zip** file and **Rename** it back to **.docx**.

8. In the same folder, locate a file named **Wire Transfer Information1-BROKEN.docx**, and copy it out to your host desktop. Go to your host desktop and double click to open the file you just copied there.

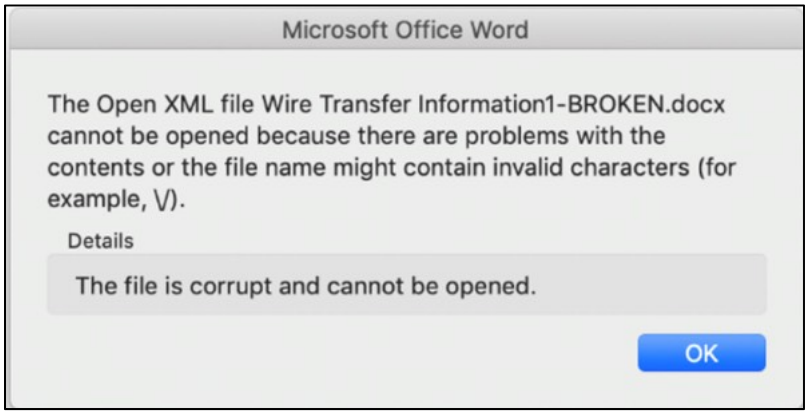


9. Of course we want to recover the contents! Click **Yes**. Then you will be presented with something like error message below. So close and yet so far. Click **OK**. We will have to approach this differently.

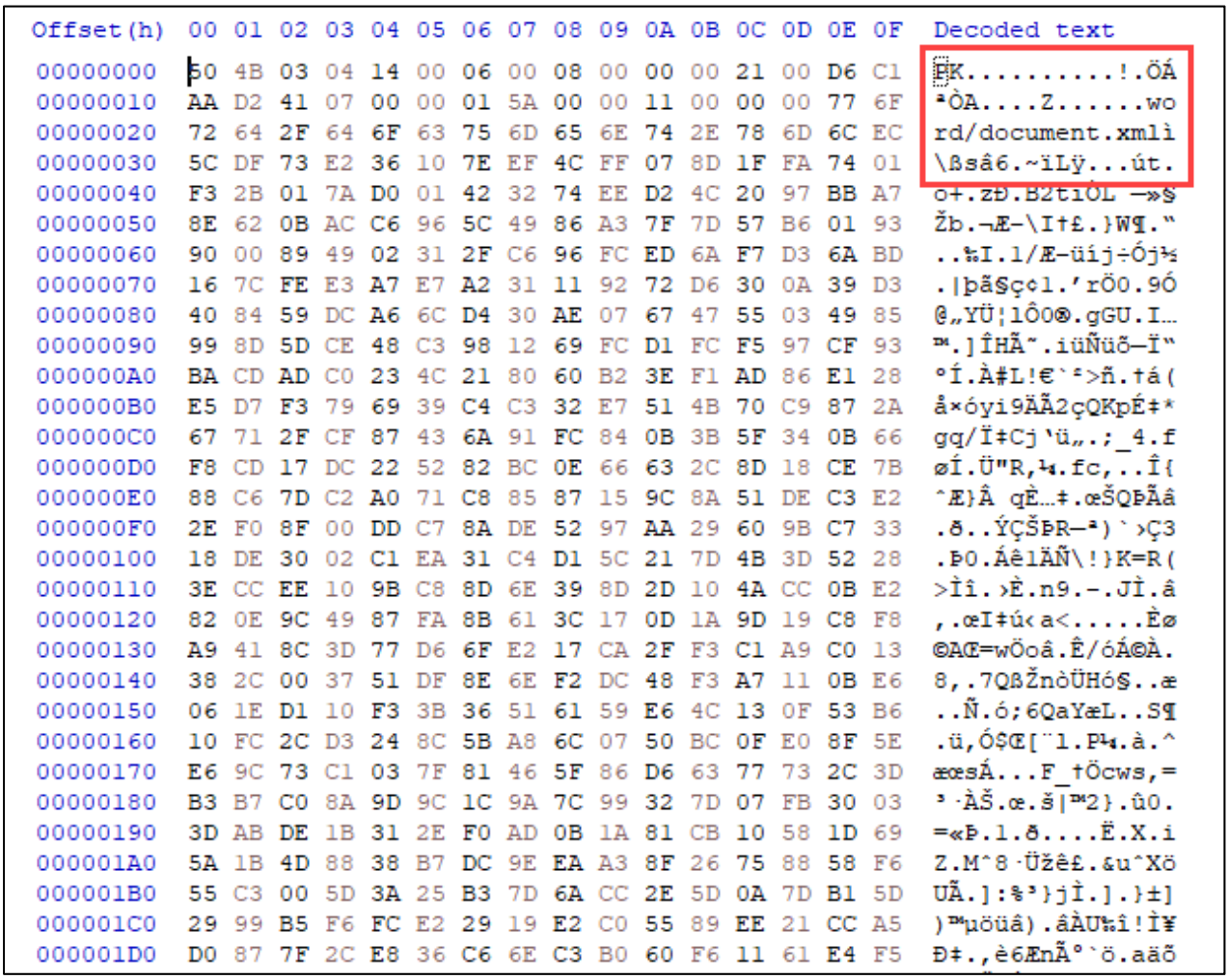




10. If you are using Office for Mac, your error message may look like this.

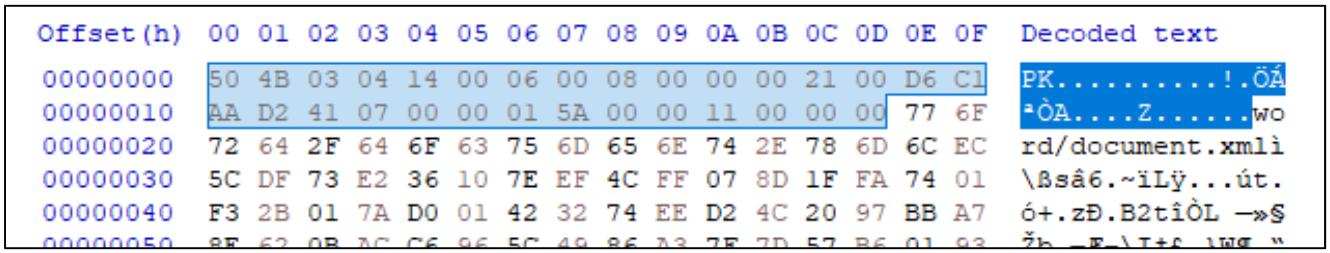


11. We have a couple of options open to us, so let's explore them. Using previous procedures, open **Wire Transfer Information1-BROKEN.docx** in the **HxD** hex editing program. Although it has the proper **.docx** file signature, we can see immediately (through experience) that the structure at the beginning is all messed up. **word/document.xml** should not be up near the file header. If you were to compare this to a known good **.docx** file, you would see that there is a significant amount of data that should be between the header and **word/document.xml**. Worse, it is mostly structure and metadata information that we have lost. This may be a losing battle, but it isn't over until it is over!



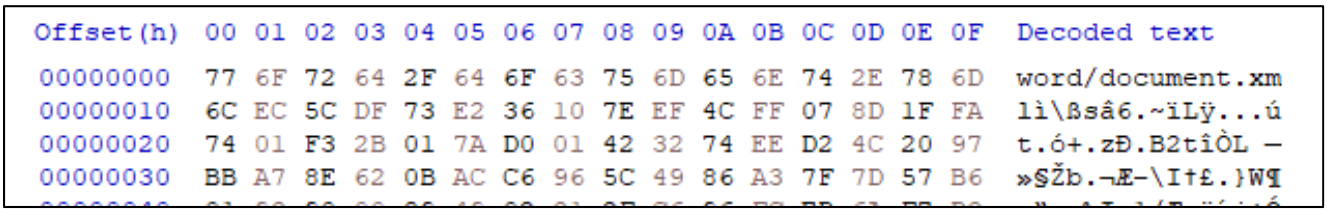
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	D6	C1	PK.....!.ÖÁ
00000010	AA	D2	41	07	00	00	01	5A	00	00	11	00	00	00	77	6F	ªÒA....Z.....wo
00000020	72	64	2F	64	6F	63	75	6D	65	6E	74	2E	78	6D	6C	EC	rd/document.xmlì
00000030	5C	DF	73	E2	36	10	7E	EF	4C	FF	07	8D	1F	FA	74	01	\ßsâ6.~ïLÿ...út.
00000040	F3	2B	01	7A	D0	01	42	32	74	EE	D2	4C	20	97	BB	A7	ó+.zÐ.B2tîÒL —»§
00000050	8E	62	0B	AC	C6	96	5C	49	86	A3	7F	7D	57	B6	01	93	Žb.¬Æ–\I†£.}W¶.“
00000060	90	00	89	49	02	31	2F	C6	96	FC	ED	6A	F7	D3	6A	BD	..‰I.1/Æ–üíj÷Ój½
00000070	16	7C	FE	E3	A7	E7	A2	31	11	92	72	D6	30	0A	39	D3	.|þã§ç¢1.’rÖ0.9Ó
00000080	40	84	59	DC	A6	6C	D4	30	AE	07	67	47	55	03	49	85	@„YÜ¦lÔ0®.gGU.I…
00000090	99	8D	5D	CE	48	C3	98	12	69	FC	D1	FC	F5	97	CF	93	™.]ÎHÃ˜.iüÑüõ—Ï“
000000A0	BA	CD	AD	C0	23	4C	21	80	60	B2	3E	F1	AD	86	E1	28	ºÍ.À#L!€`²>ñ.†á(
000000B0	E5	D7	F3	79	69	39	C4	C3	32	E7	51	4B	70	C9	87	2A	å×óyi9ÄÃ2çQKpÉ‡*
000000C0	67	71	2F	CF	87	43	6A	91	FC	84	0B	3B	5F	34	0B	66	gq/Ï‡Cj‘ü„.;_4.f
000000D0	F8	CD	17	DC	22	52	82	BC	0E	66	63	2C	8D	18	CE	7B	øÍ.Ü"R‚¼.fc,..Î{
000000E0	88	C6	7D	C2	A0	71	C8	85	87	15	9C	8A	51	DE	C3	E2	ˆÆ}Â qÈ…‡.œŠQÞÃâ
000000F0	2E	F0	8F	00	DD	C7	8A	DE	52	97	AA	29	60	9B	C7	33	.ð..ÝÇŠÞR—ª)`›Ç3
00000100	18	DE	30	02	C1	EA	31	C4	D1	5C	21	7D	4B	3D	52	28	.Þ0.ÁêlÄÑ\!}K=R(
00000110	3E	CC	EE	10	9B	C8	8D	6E	39	8D	2D	10	4A	CC	0B	E2	>Ìî.›È.n9.-.JÌ.â
00000120	82	0E	9C	49	87	FA	8B	61	3C	17	0D	1A	9D	19	C8	F8	‚.œI‡ú‹a<.....Èø
00000130	A9	41	8C	3D	77	D6	6F	E2	17	CA	2F	F3	C1	A9	C0	13	©AŒ=wÖoâ.Ê/óÁ©À.
00000140	38	2C	00	37	51	DF	8E	6E	F2	DC	48	F3	A7	11	0B	E6	8,.7QßŽnòÜHó§..æ
00000150	06	1E	D1	10	F3	3B	36	51	61	59	E6	4C	13	0F	53	B6	..Ñ.ó;6QaYæL..S¶
00000160	10	FC	2C	D3	24	8C	5B	A8	6C	07	50	BC	0F	E0	8F	5E	.ü,Ó$Œ[¨l.P¼.à.^
00000170	E6	9C	73	C1	03	7F	81	46	5F	86	D6	63	77	73	2C	3D	æœsÁ...F_†Öcws,=
00000180	B3	B7	C0	8A	9D	9C	1C	9A	7C	99	32	7D	07	FB	30	03	³·ÀŠ.œ.š|™2}.û0.
00000190	3D	AB	DE	1B	31	2E	F0	AD	0B	1A	81	CB	10	58	1D	69	=«Þ.1.ð....Ë.X.i
000001A0	5A	1B	4D	88	38	B7	DC	9E	EA	A3	8F	26	75	88	58	F6	Z.M^8·ÜžêŁ.&u^Xö
000001B0	55	C3	00	5D	3A	25	B3	7D	6A	CC	2E	5D	0A	7D	B1	5D	UÃ.]:%³}jÌ.].}±]
000001C0	29	99	B5	F6	FC	E2	29	19	E2	C0	55	89	EE	21	CC	A5	)™µöüâ).âÀU‰î!Ì¥
000001D0	D0	87	7F	2C	E8	36	C6	6E	C3	B0	60	F6	11	61	E4	F5	Ð‡.,è6ÆnÃ°`ö.aäõ

12. Select all data prior to **word/document.xml**…



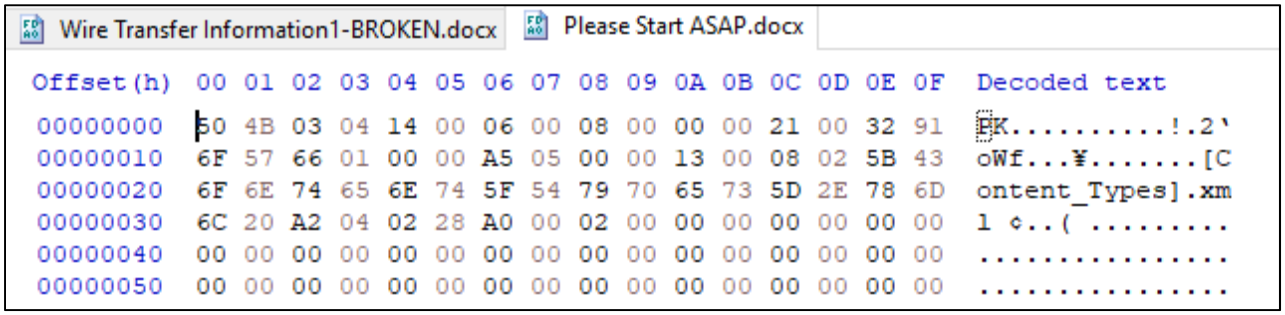
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	D6	C1	PK.....!.ÖÁ
00000010	AA	D2	41	07	00	00	01	5A	00	00	11	00	00	00	77	6F	ªÒA....Z.....wo
00000020	72	64	2F	64	6F	63	75	6D	65	6E	74	2E	78	6D	6C	EC	rd/document.xmlì
00000030	5C	DF	73	E2	36	10	7E	EF	4C	FF	07	8D	1F	FA	74	01	\ßsâ6.~ïLÿ...út.
00000040	F3	2B	01	7A	D0	01	42	32	74	EE	D2	4C	20	97	BB	A7	ó+.zÐ.B2tîÒL —»§

13. …and delete it. Accept the **Confirm** prompt. You should be left with this.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	77	6F	72	64	2F	64	6F	63	75	6D	65	6E	74	2E	78	6D	word/document.xm
00000010	6C	EC	5C	DF	73	E2	36	10	7E	EF	4C	FF	07	8D	1F	FA	lì\ßsâ6.~ïLÿ...ú
00000020	74	01	F3	2B	01	7A	D0	01	42	32	74	EE	D2	4C	20	97	t.ó+.zÐ.B2tîÒL —
00000030	BB	A7	8E	62	0B	AC	C6	96	5C	49	86	A3	7F	7D	57	B6	»§Žb.¬Æ–\I†£.}W¶

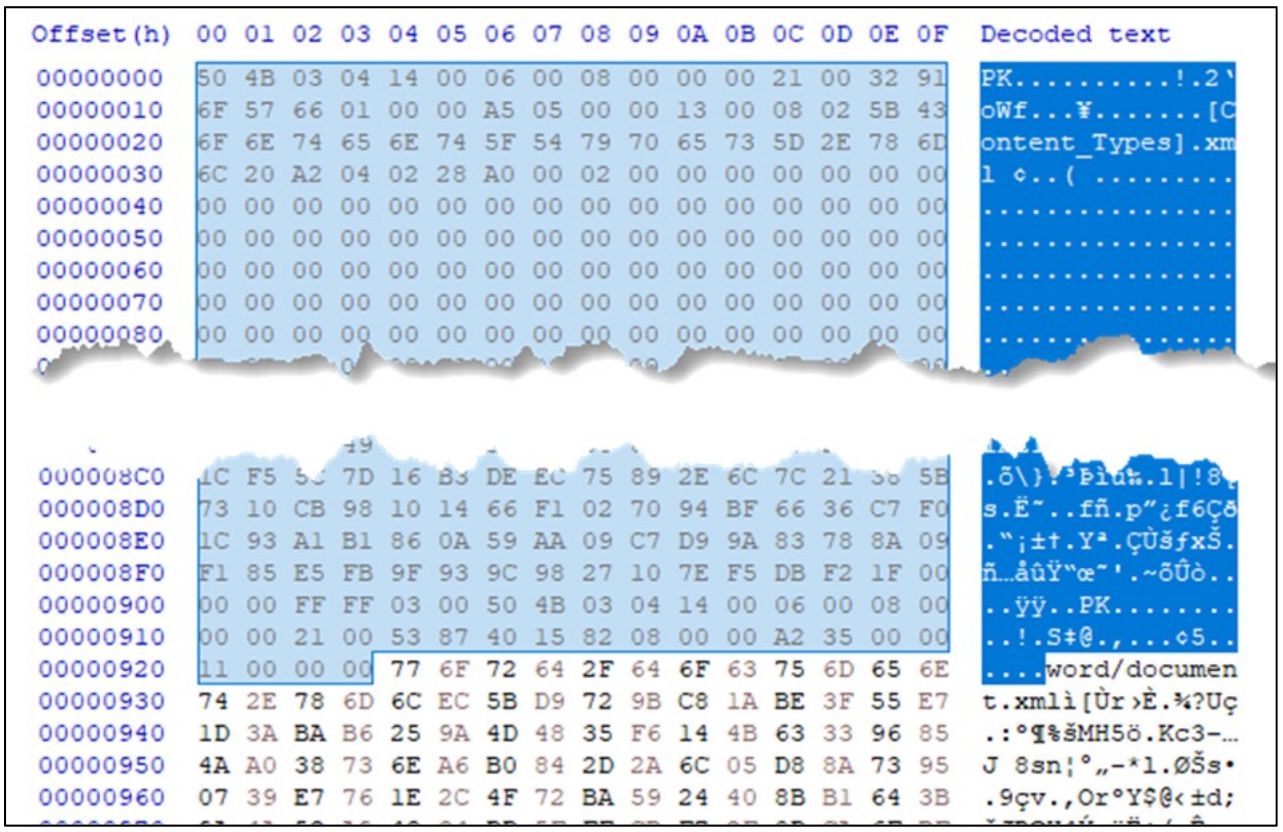
14. In **HxD**, click on **File → Open**, and navigate to **C:\Cases\Exercises\6.3** and locate a file called **Please Start ASAP.docx**, and open it. You will see a second pane of data in **HxD**, and you should be looking at the content of the new file you just opened. This is a known good **.docx** file. Note the difference between the beginning of this file and the beginning of the **BROKEN** file. Take a few moments and scroll down through this file looking for **word/document.xml**. Note just how much data is missing from the **BROKEN** file.



Wire Transfer Information1-BROKEN.docx | Please Start ASAP.docx

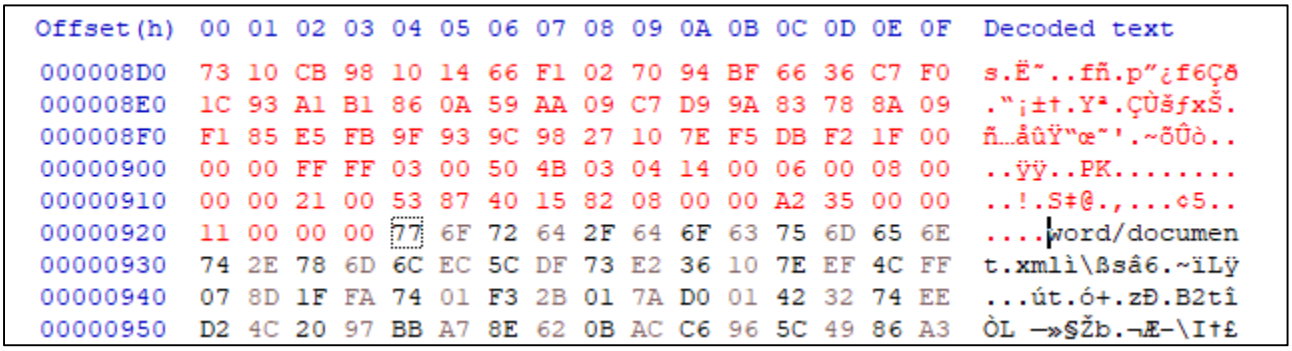
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	32	91	PK.....!.2'
00000010	6F	57	66	01	00	00	A5	05	00	00	13	00	08	02	5B	43	oWf...¥.....[C
00000020	6F	6E	74	65	6E	74	5F	54	79	70	65	73	5D	2E	78	6D	ontent_Types].xm
00000030	6C	20	A2	04	02	28	A0	00	02	00	00	00	00	00	00	00	l ¢..(.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

15. Scroll back to the top of the known good file and start sweeping from the beginning of the file signature, all the way down to just before **word/document.xml**.



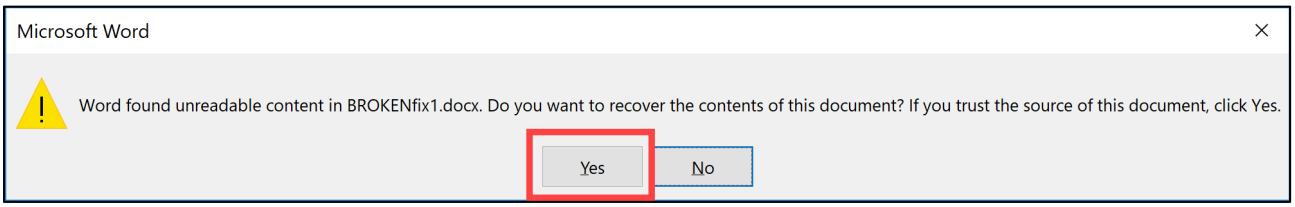
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	50	4B	03	04	14	00	06	00	08	00	00	00	21	00	32	91	PK.....!.2'
00000010	6F	57	66	01	00	00	A5	05	00	00	13	00	08	02	5B	43	oWf...¥.....[C
00000020	6F	6E	74	65	6E	74	5F	54	79	70	65	73	5D	2E	78	6D	ontent_Types].xm
00000030	6C	20	A2	04	02	28	A0	00	02	00	00	00	00	00	00	00	l ¢..(.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000008C0	1C	F5	5C	7D	16	B3	DE	EC	75	89	2E	6C	7C	21	38	5B	.õ\}.³Þìu‰.l|!8[
000008D0	73	10	CB	98	10	14	66	F1	02	70	94	BF	66	36	C7	F0	s.Ë˜..fñ.p”¿f6Çð
000008E0	1C	93	A1	B1	86	0A	59	AA	09	C7	D9	9A	83	78	8A	09	.“¡±†.Yª.ÇÙšƒxŠ.
000008F0	F1	85	E5	FB	9F	93	9C	98	27	10	7E	F5	DB	F2	1F	00	ñ…åûŸ“œ˜'.~õÛò..
00000900	00	00	FF	FF	03	00	50	4B	03	04	14	00	06	00	08	00	..ÿÿ..PK.....
00000910	00	00	21	00	53	87	40	15	82	08	00	00	A2	35	00	00	..!.S‡@.‚...¢5..
00000920	11	00	00	00	77	6F	72	64	2F	64	6F	63	75	6D	65	6E	....word/documen
00000930	74	2E	78	6D	6C	EC	5B	D9	72	9B	C8	1A	BE	3F	55	E7	t.xmlì[Ùr›È.¾?Uç
00000940	1D	3A	BA	B6	25	9A	4D	48	35	F6	14	4B	63	33	96	85	.:º¶%šMH5ö.Kc3–…
00000950	4A	A0	38	73	6E	A6	B0	84	2D	2A	6C	05	D8	8A	73	95	J 8sn¦°„-*l.ØŠs•
00000960	07	39	E7	76	1E	2C	4F	72	BA	59	24	40	8B	B1	64	3B	.9çv.,Orº Y$@‹±d;

16. Click on **Ctrl+C** to copy the data you have just swept. Accept any warnings. Switch back to the **BROKEN** file by clicking on its tab at the top of the **HxD** program. Position your cursor at the very beginning of the file, just in front of **word/document.xml**, and press **Ctrl-V**. This will paste your copied content into the **BROKEN** document. You will see your pasted data in red.

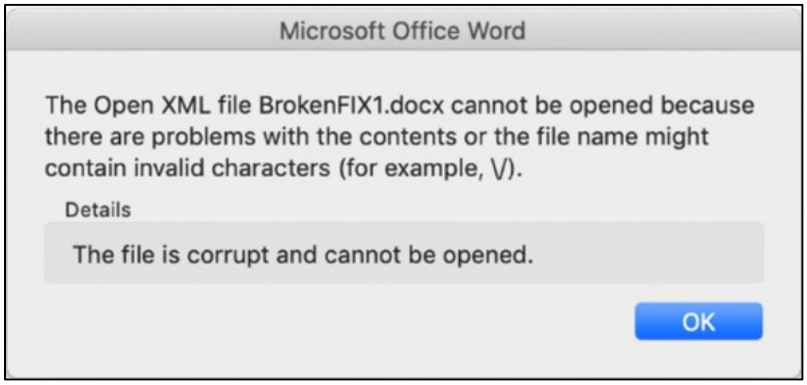


Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
000008D0	73	10	CB	98	10	14	66	F1	02	70	94	BF	66	36	C7	F0	s.Ë˜..fñ.p”¿f6Çð
000008E0	1C	93	A1	B1	86	0A	59	AA	09	C7	D9	9A	83	78	8A	09	.“¡±†.Yª.ÇÙšƒxŠ.
000008F0	F1	85	E5	FB	9F	93	9C	98	27	10	7E	F5	DB	F2	1F	00	ñ…åûŸ“œ˜'.~õÛò..
00000900	00	00	FF	FF	03	00	50	4B	03	04	14	00	06	00	08	00	..ÿÿ..PK.....
00000910	00	00	21	00	53	87	40	15	82	08	00	00	A2	35	00	00	..!.S‡@.,...¢5..
00000920	11	00	00	00	77	6F	72	64	2F	64	6F	63	75	6D	65	6E	....word/documen
00000930	74	2E	78	6D	6C	EC	5C	DF	73	E2	36	10	7E	EF	4C	FF	t.xmlì\ßsâ6.~ïLÿ
00000940	07	8D	1F	FA	74	01	F3	2B	01	7A	D0	01	42	32	74	EE	...út.ó+.zÐ.B2tî
00000950	D2	4C	20	97	BB	A7	8E	62	0B	AC	C6	96	5C	49	86	A3	ÒL –»§Žb.¬Æ–\I†£

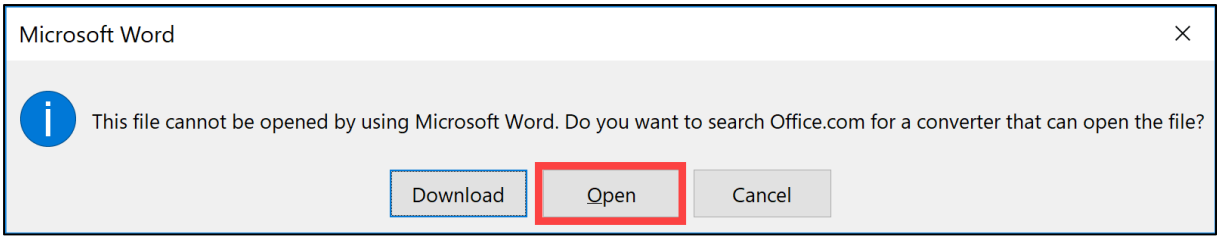
17. Click on **File -> Save As**, and save this new data set to your **Desktop**, calling it **BROKENfix1.docx**. Navigate to the **Desktop** and copy this file to your **host desktop**. Try to open it. No luck. We get the error message as before. Click **Yes**, to try to recover contents.



18. If you are using Office for Mac, your error message may look like this.

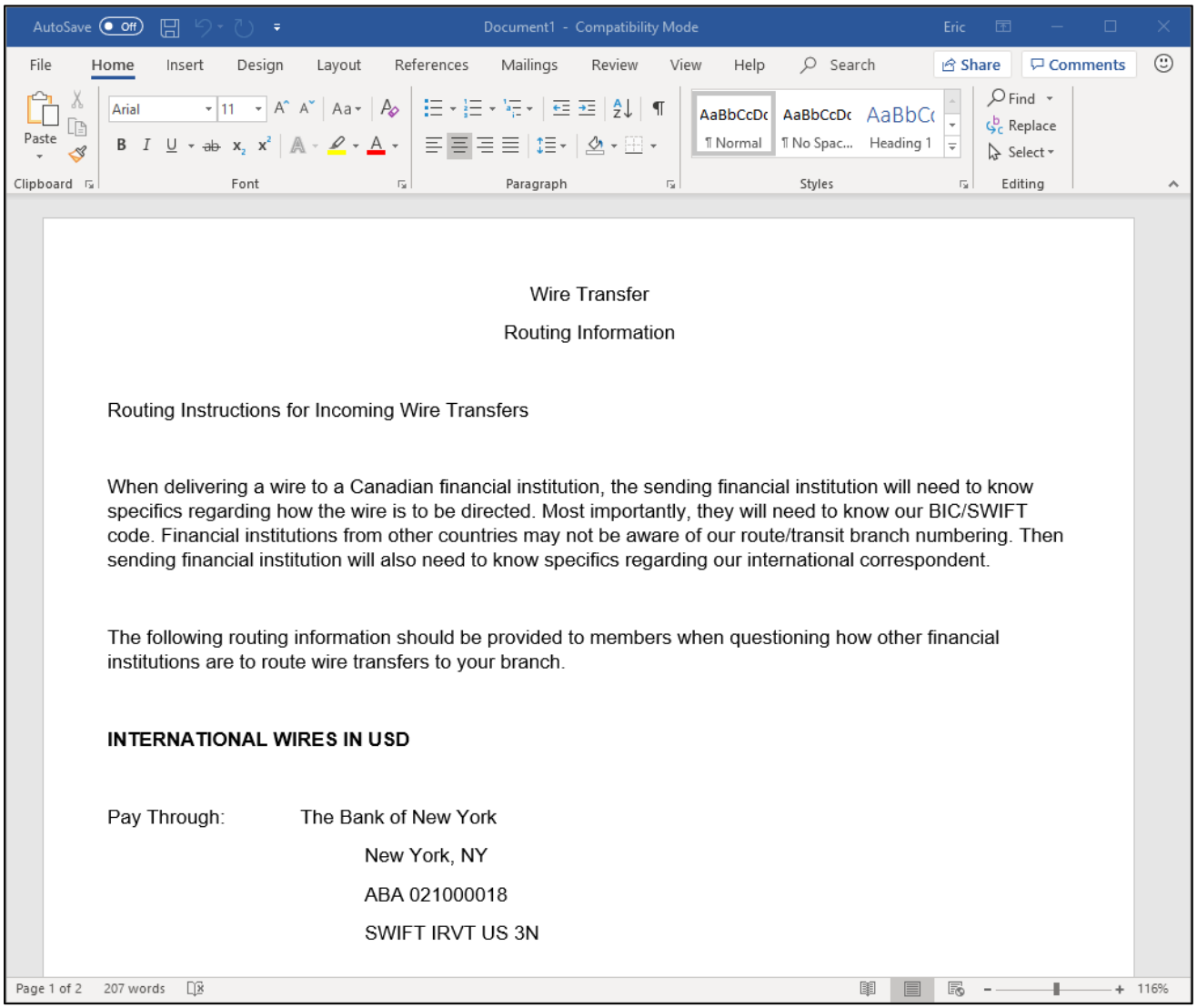


19. Now we get a different error message. We don’t want to download a converter, but let’s click **Open**.



20. If you are using Office for Mac, your error message may look like this.

21. SUCCESS!

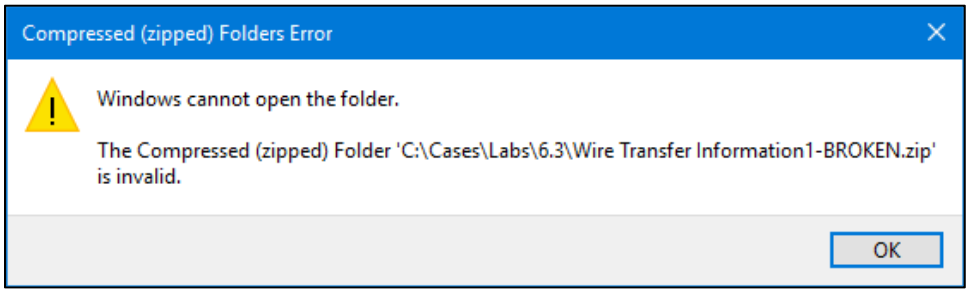


Exercise – Recover/Repair a .DOCX File – Method 2

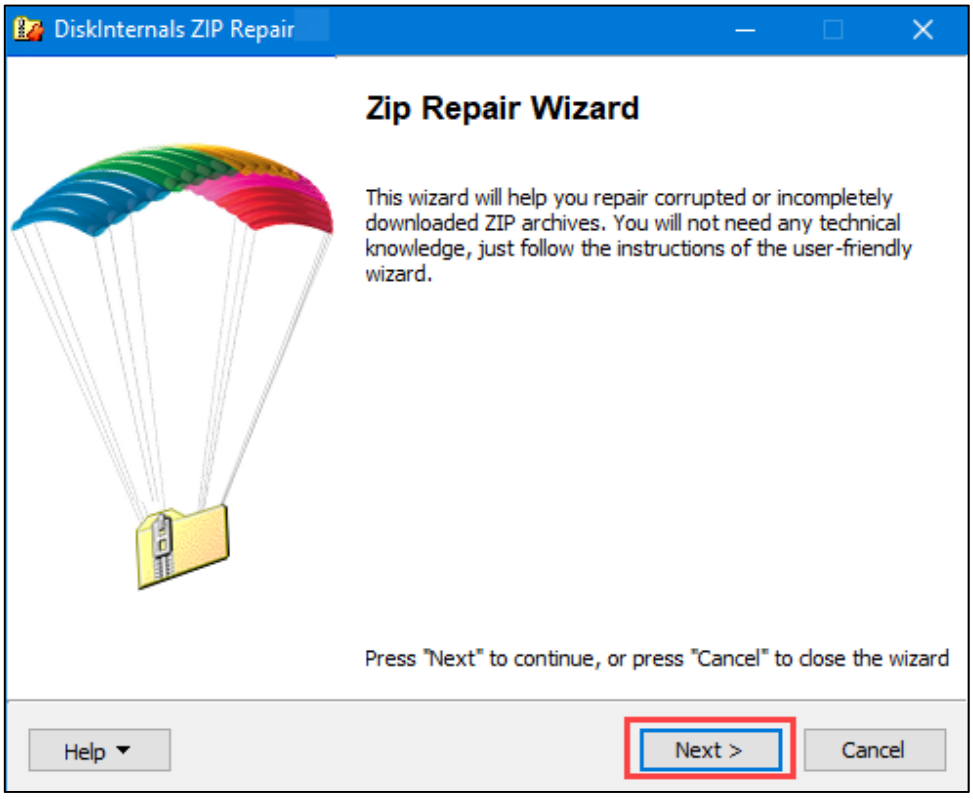
As mentioned in the previous section, there may be more than one way to gain success in repairing a **.docx** file and extracting the data from it. In our previous section, we saw that if we edited the hex code of the file, we achieved our goal. But what if we hadn't? In this section, we will explore a second possible method in hopes of accomplishing the same goal.

1. Ensure all files and folders are closed from the last section.

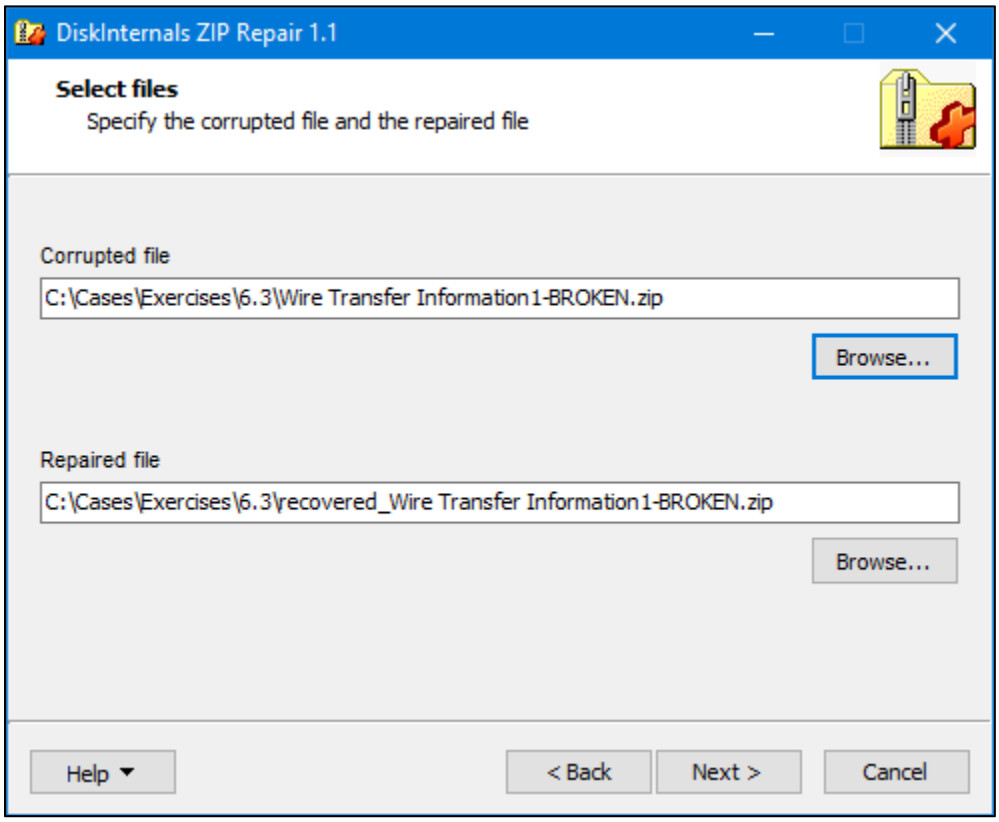
2. Navigate to **C:\Cases\Exercises\6.3** and locate **Wire Transfer Information1-BROKEN.docx.** Once found, try to rename it from **.docx** to **.zip**, and open it. Remember that even if we couldn't repair it in the hex editor, maybe we can open up the **word/document.xml** file, and pull raw data from there as a last resort. Although not optimal or pretty, it is better than nothing. But when you try to open the **zip** file, you should receive an error message. Click **OK** to close the error message.



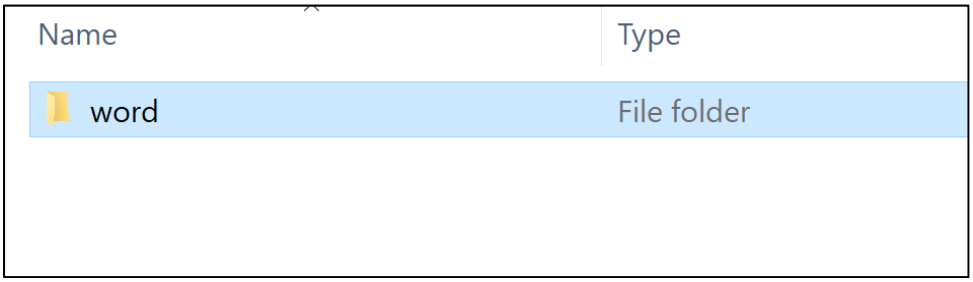
3. Navigate to your **VM Desktop** inside the **Utilities** fence, and locate a program called **DiskInternals Zip Repair**. Double click to open it, and then click **Next**.



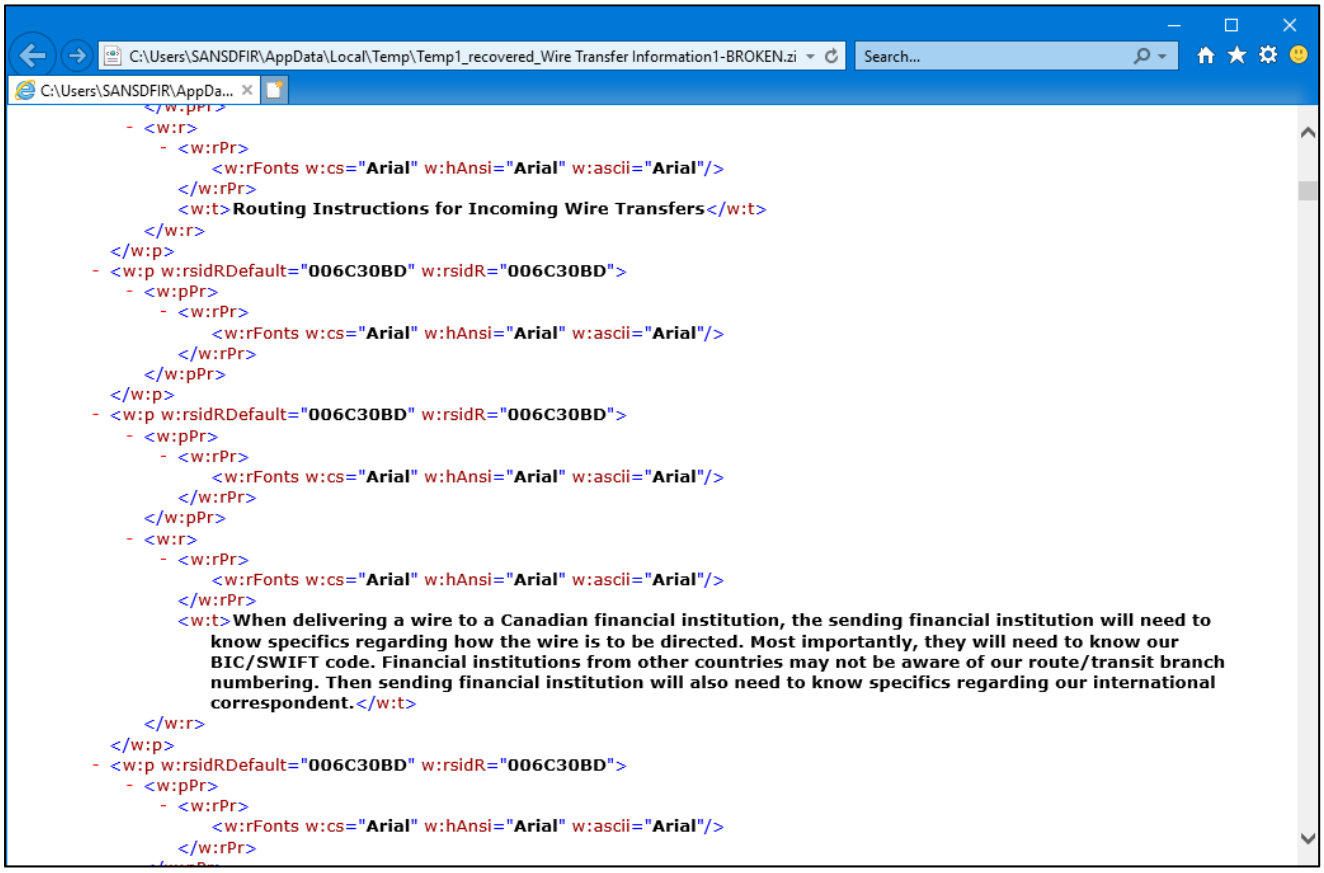
4. Select the file you wish to repair, which is now titled **Wire Transfer Information1-BROKEN.zip** by browsing to its location. A **Repaired file** location will be chosen for you, and you can change if you want, but we will accept the default. Make sure you take a note of it. Then click **Next**.



5. You will be shown a list of what was repaired, which in this case isn't much because of how much damage there was. Click **Next**, then **Finish**. Navigate to the repaired **zip** file and open it.



6. There certainly is not much to see, but if you open the **word** folder, and then the **document.xml** file inside, you will be able to see the text content of the original file. If this is all you are able to get, it is still a win, because before today, you would have tried to open the .**docx**, gotten an error message, deleted it, and moved on to the next automatically carved file.



7. By the way, as we have often said, never trust just one tool. Surprisingly enough (or maybe not), the .**zip** file that wouldn't open using the Windows unzip tool (causing this repair) seems to open just fine using **7-zip**, without any repair!

Exercise—Key Takeaways

- Data is often available, even if it is not in a format that the examiner is familiar with.
- If you receive an error warning that a document cannot be opened, and you don't investigate further, this can have significant consequences.
- It takes practice to know what is and is not possible.

This page intentionally left blank.

Optional - Out of Class
Exercise 6.3D—Data Carving & Rebuilding Bonus 1

Exercise Objectives

- Recognize various common file signatures
- Carve manually for a deleted file
- Identify hidden data to gain context

OPTIONAL ADVANCED EXERCISE 1

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

3. Using the techniques you have learned, mount the **CARVEY.E01** file with **Arsenal Image Mounter**, and then open the mounted physical drive in **HxD**.

OPTIONAL ADVANCED EXERCISE 1 - Questions

1. Using the search term **`Tutush`**, see if you can locate responsive data.

2. If you do, try to extract something usable.

   a. Did you find a file?

   _____

   b. If so, what type of file is it?

   _____

3. Open the file and read through it.

   a. What information is in there about the term **Tutush**?

   _____

b. According to internal metadata, what was the file's **Title**?

_____

c. What is **Tutush** in reference to?

_____

d. What is the **Applicant** name in the file?

_____

4. Close the recovered file.

5. Refer back to **HxD** and try to locate data using the search term of **`engineering`**.

a. Were you able to extract a full file?

_____

b. If yes, what type of file is it?

_____

c. What content, if any, was recovered?

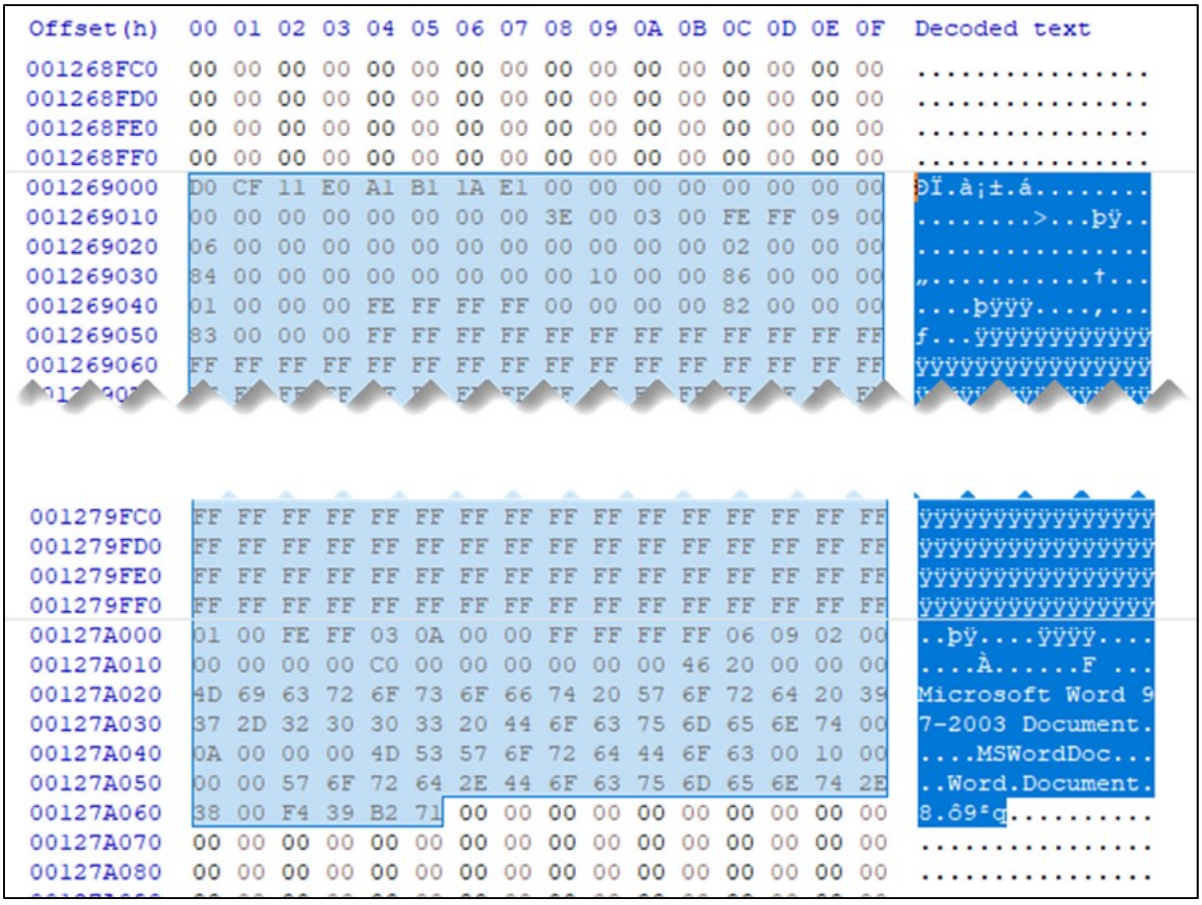
_____

OPTIONAL ADVANCED EXERCISE 1 - Step-by-Step

1. Using the search term **`Tutush`**, see if you can locate responsive data.

2. If you do, try to extract something usable.

   After finding the search term, look downwards from the search hit and keep scrolling until you see the Word file footer ô9²q. Once found, place the cursor immediately following the q and start sweeping upwards, looking for the file header. Look at the Offset line number in the screenshot for a hint of where the beginning and end are.



   a. Did you find a file?

      YES

   b. If so, what type of file is it?

      This is a .DOC Word document. You can tell by the file header and footer.

3. Open the file and read through it.

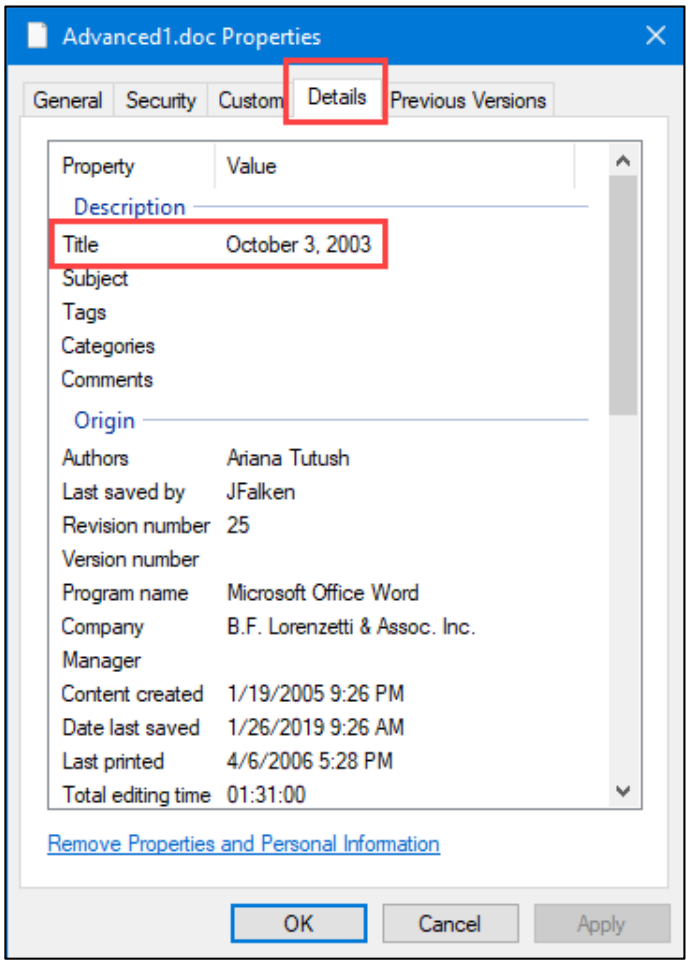
   a. What information is in there about the term **Tutush**?

   No reference to the term Tutush is in the document.

   b. According to internal metadata, what was the file's **Title**?

   October 3, 2003

   Once the document has been saved out to the desktop, right click on it and select Properties. The Title of the document is in the information under the Details tab.



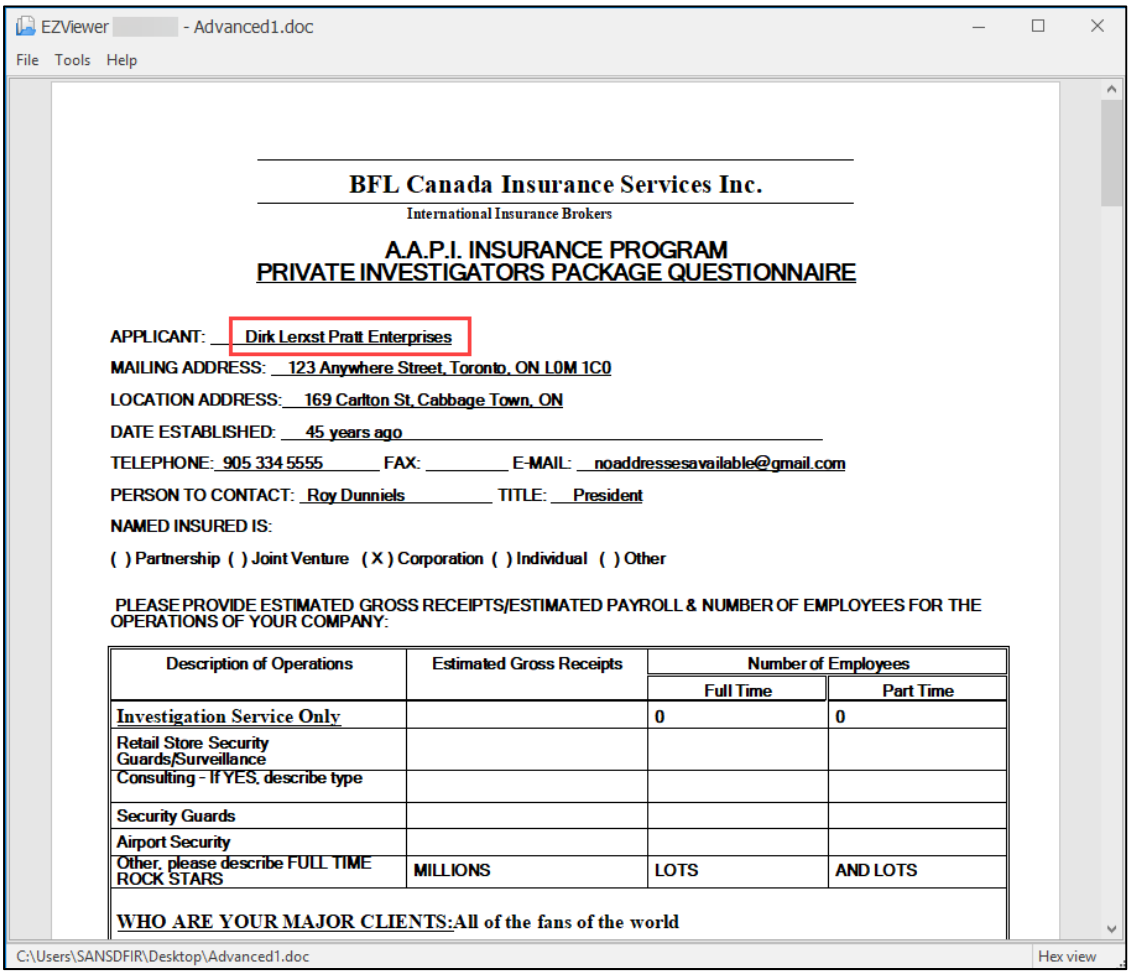
   c. What is **Tutush** in reference to?

   This is in reference to Ariana Tutush. She is the Author of the document, and this can be seen in the Properties. Many examiners miss the relevance of a random hit like Tutush in a data search, simply because it has no context around it, and so looks simply like a remnant of something that is no longer available.

d. What is the **Applicant** name in the file?

Dirk Lerxst Pratt Enterprises

Right click on the file and open with EZViewer to find this answer.



4. Close the recovered file.

5. Refer back to **HxD**, start back in **0** sector, and try to locate data using the search term of **`engineering`**.

a. Were you able to extract a full file?

YES

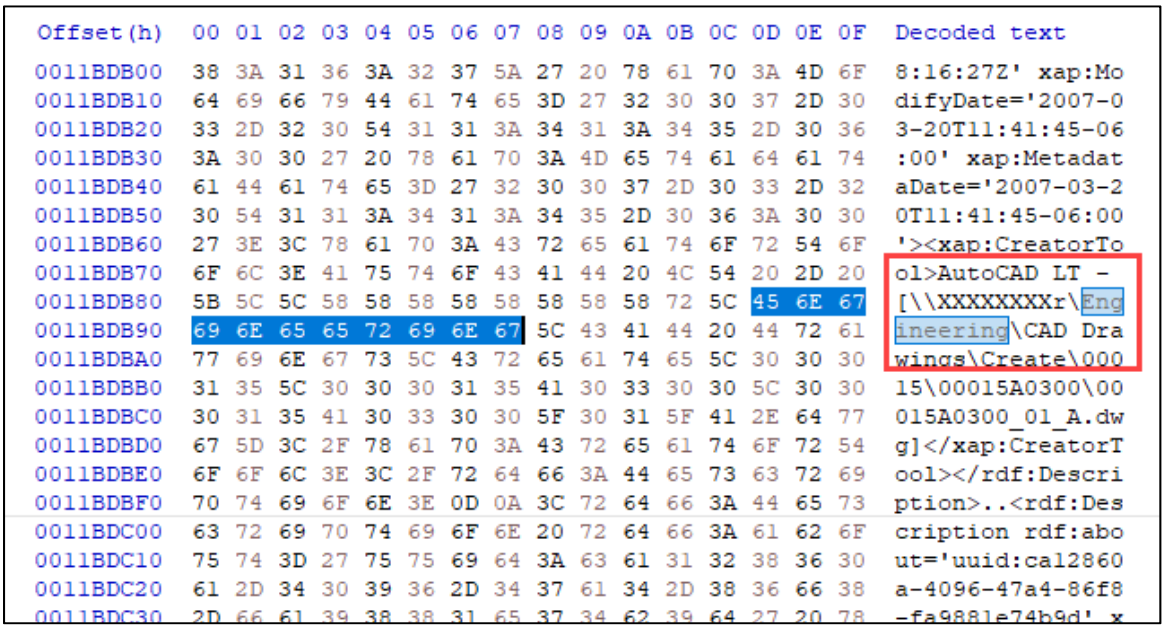
b. If yes, what type of file is it?

PDF

c. What content, if any, was recovered?

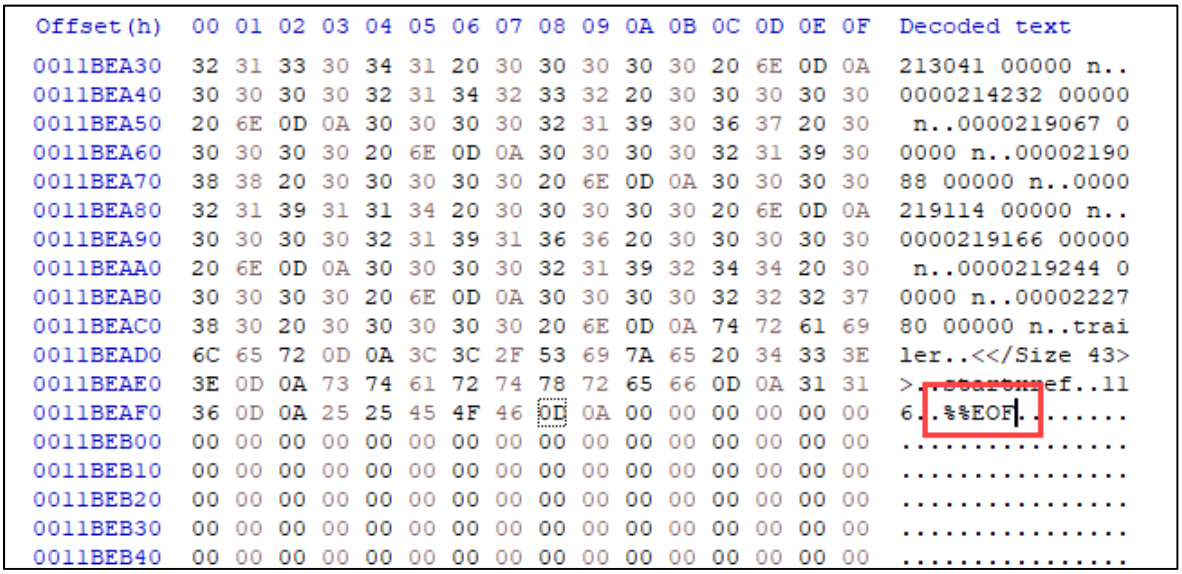
Blueprint drawings

We see the search hit in a bunch of hard to read data that suggests an AutoCAD file.



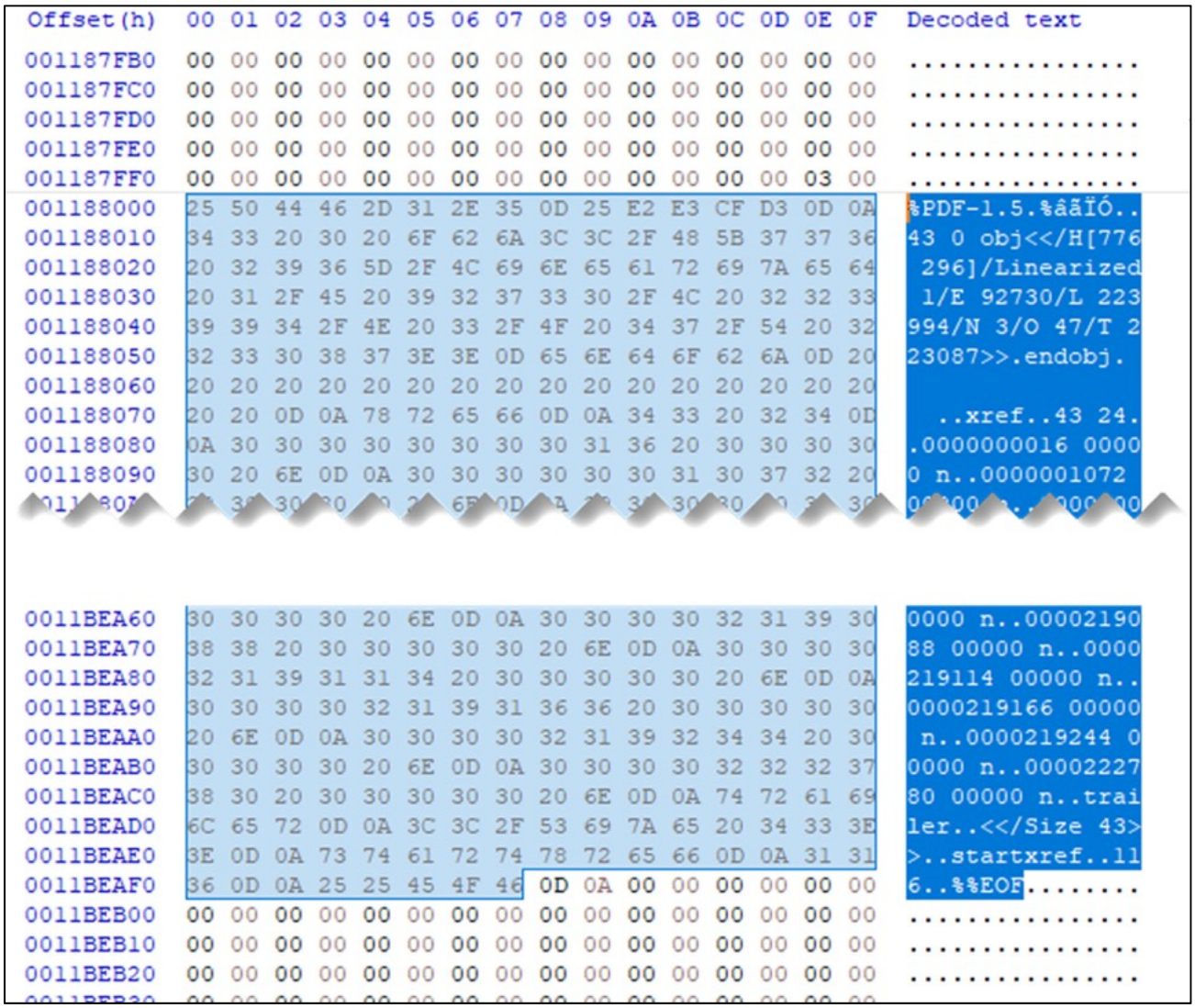
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0011BDB00	38	3A	31	36	3A	32	37	5A	27	20	78	61	70	3A	4D	6F	8:16:27Z' xap:Mo
0011BDB10	64	69	66	79	44	61	74	65	3D	27	32	30	30	37	2D	30	difyDate='2007-0
0011BDB20	33	2D	32	30	54	31	31	3A	34	31	3A	34	35	2D	30	36	3-20T11:41:45-06
0011BDB30	3A	30	30	27	20	78	61	70	3A	4D	65	74	61	64	61	74	:00' xap:Metadat
0011BDB40	61	44	61	74	65	3D	27	32	30	30	37	2D	30	33	2D	32	aDate='2007-03-2
0011BDB50	30	54	31	31	3A	34	31	3A	34	35	2D	30	36	3A	30	30	0T11:41:45-06:00
0011BDB60	27	3E	3C	78	61	70	3A	43	72	65	61	74	6F	72	54	6F	'><xap:CreatorTo
0011BDB70	6F	6C	3E	41	75	74	6F	43	41	44	20	4C	54	20	2D	20	ol>AutoCAD LT -
0011BDB80	5B	5C	5C	58	58	58	58	58	58	58	58	72	5C	45	6E	67	[\\XXXXXXXXr\Eng
0011BDB90	69	6E	65	65	72	69	6E	67	5C	43	41	44	20	44	72	61	ineering\CAD Dra
0011BDBA0	77	69	6E	67	73	5C	43	72	65	61	74	65	5C	30	30	30	wings\Create\000
0011BDBB0	31	35	5C	30	30	30	31	35	41	30	33	30	30	5C	30	30	15\00015A0300\00
0011BDBC0	30	31	35	41	30	33	30	30	5F	30	31	5F	41	2E	64	77	015A0300_01_A.dw
0011BDBD0	67	5D	3C	2F	78	61	70	3A	43	72	65	61	74	6F	72	54	g]</xap:CreatorT
0011BDBE0	6F	6F	6C	3E	3C	2F	72	64	66	3A	44	65	73	63	72	69	ool></rdf:Descri
0011BDBF0	70	74	69	6F	6E	3E	0D	0A	3C	72	64	66	3A	44	65	73	ption>..<rdf:Des
0011BDC00	63	72	69	70	74	69	6F	6E	20	72	64	66	3A	61	62	6F	cription rdf:abo
0011BDC10	75	74	3D	27	75	75	69	64	3A	63	61	31	32	38	36	30	ut='uuid:ca12860
0011BDC20	61	2D	34	30	39	36	2D	34	37	61	34	2D	38	36	66	38	a-4096-47a4-86f8
0011BDC30	2D	66	61	39	38	38	31	65	37	34	62	39	64	27	20	78	-fa9881e74b9d' x

Start looking further down to see if you find anything to suggest the end of a file.



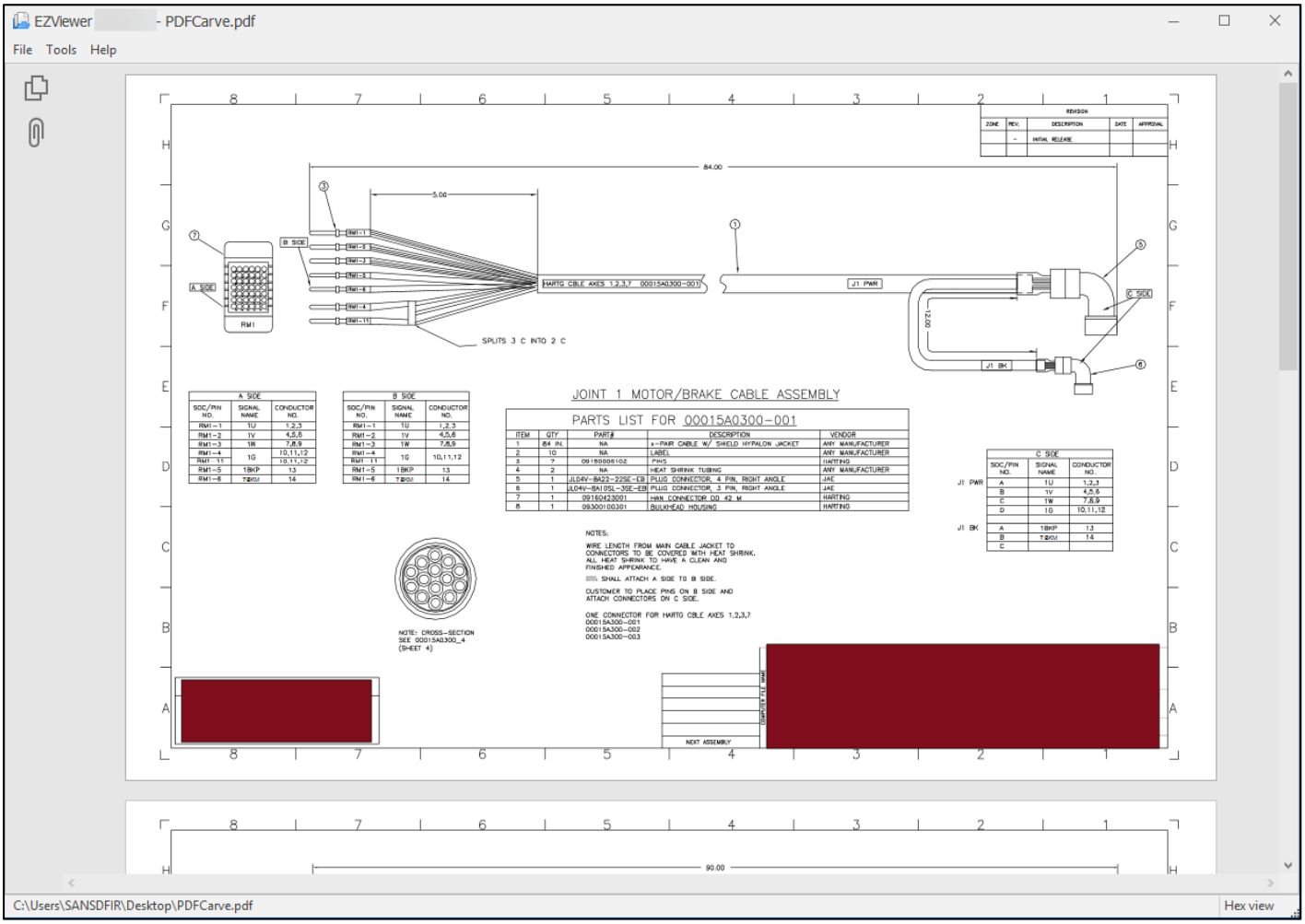
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0011BEA30	32	31	33	30	34	31	20	30	30	30	30	30	20	6E	0D	0A	213041 00000 n..
0011BEA40	30	30	30	30	32	31	34	32	33	32	20	30	30	30	30	30	0000214232 00000
0011BEA50	20	6E	0D	0A	30	30	30	30	32	31	39	30	36	37	20	30	n..0000219067 0
0011BEA60	30	30	30	30	20	6E	0D	0A	30	30	30	30	32	31	39	30	0000 n..00002190
0011BEA70	38	38	20	30	30	30	30	30	20	6E	0D	0A	30	30	30	30	88 00000 n..0000
0011BEA80	32	31	39	31	31	34	20	30	30	30	30	30	20	6E	0D	0A	219114 00000 n..
0011BEA90	30	30	30	30	32	31	39	31	36	36	20	30	30	30	30	30	0000219166 00000
0011BEAA0	20	6E	0D	0A	30	30	30	30	32	31	39	32	34	34	20	30	n..0000219244 0
0011BEAB0	30	30	30	30	20	6E	0D	0A	30	30	30	30	32	32	32	37	0000 n..00002227
0011BEAC0	38	30	20	30	30	30	30	30	20	6E	0D	0A	74	72	61	69	80 00000 n..trai
0011BEAD0	6C	65	72	0D	0A	3C	3C	2F	53	69	7A	65	20	34	33	3E	ler..<</Size 43>
0011BEAE0	3E	0D	0A	73	74	61	72	74	78	72	65	66	0D	0A	31	31	>..startxref..11
0011BEAF0	36	0D	0A	25	25	45	4F	46	0D	0A	00	00	00	00	00	00	6..%%EOF.....
0011BEB00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0011BEB10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0011BEB20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0011BEB30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0011BEB40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

You have found what looks to be a PDF file footer. Start sweeping up from here until you find what you believe to be the file header. What does the file header look like in a PDF? Refer back to the slides on the subject.



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
001187FB0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001187FC0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001187FD0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001187FE0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001187FF0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00  ................
001188000  25 50 44 46 2D 31 2E 35 0D 25 E2 E3 CF D3 0D 0A  %PDF-1.5.%âãÏÓ..
001188010  34 33 20 30 20 6F 62 6A 3C 3C 2F 48 5B 37 37 36  43 0 obj<</H[776
001188020  20 32 39 36 5D 2F 4C 69 6E 65 61 72 69 7A 65 64   296]/Linearized
001188030  20 31 2F 45 20 39 32 37 33 30 2F 4C 20 32 32 33   1/E 92730/L 223
001188040  39 39 34 2F 4E 20 33 2F 4F 20 34 37 2F 54 20 32  994/N 3/O 47/T 2
001188050  32 33 30 38 37 3E 3E 0D 65 6E 64 6F 62 6A 0D 20  23087>>.endobj.
001188060  20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
001188070  20 20 0D 0A 78 72 65 66 0D 0A 34 33 20 32 34 0D    ..xref..43 24.
001188080  0A 30 30 30 30 30 30 30 30 31 36 20 30 30 30 30  .0000000016 0000
001188090  30 20 6E 0D 0A 30 30 30 30 30 30 31 30 37 32 20  0 n..0000001072
0011880A0  30 30 30 30 20 6E 0D 0A 30 30 30 30 30 30 31 30  0000 n..0000010
...
0011BEA60  30 30 30 30 20 6E 0D 0A 30 30 30 30 32 31 39 30  0000 n..00002190
0011BEA70  38 38 20 30 30 30 30 30 20 6E 0D 0A 30 30 30 30  88 00000 n..0000
0011BEA80  32 31 39 31 31 34 20 30 30 30 30 30 20 6E 0D 0A  219114 00000 n..
0011BEA90  30 30 30 30 32 31 39 31 36 36 20 30 30 30 30 30  0000219166 00000
0011BEAA0  20 6E 0D 0A 30 30 30 30 32 31 39 32 34 34 20 30   n..0000219244 0
0011BEAB0  30 30 30 30 20 6E 0D 0A 30 30 30 30 32 32 32 37  0000 n..00002227
0011BEAC0  38 30 20 30 30 30 30 30 20 6E 0D 0A 74 72 61 69  80 00000 n..trai
0011BEAD0  6C 65 72 0D 0A 3C 3C 2F 53 69 7A 65 20 34 33 3E  ler..<</Size 43>
0011BEAE0  3E 0D 0A 73 74 61 72 74 78 72 65 66 0D 0A 31 31  >..startxref..11
0011BEAF0  36 0D 0A 25 25 45 4F 46 0D 0A 00 00 00 00 00 00  6..%%EOF........
0011BEB00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0011BEB10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0011BEB20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Use the techniques you have learned to create a new data space in HxD. Copy this sweep, save it to that space, then save it out as a .PDF onto the VM desktop. Open it up using EZViewer.



Exercise - Key Takeaways

- Data carving and rebuilding can be very time consuming and frustrating.
- A forensicator could spend hours trying to repair a file and have no success.
- If it was easy, everyone would be doing it.

Optional - Out of Class
Exercise 6.3E—Data Carving & Rebuilding Bonus 2

Exercise Objectives

- Recognize various common file signatures
- Carve manually for a deleted file
- Rebuild a partially overwritten file
- Repair corrupted or overwritten files

Exercise Preparation

1. Boot your **FOR498 Windows VM**

2. Login to the **FOR498 Windows VM** using the following credentials:
   a. Username: **SANSDFIR**
   b. Password: **forensics**

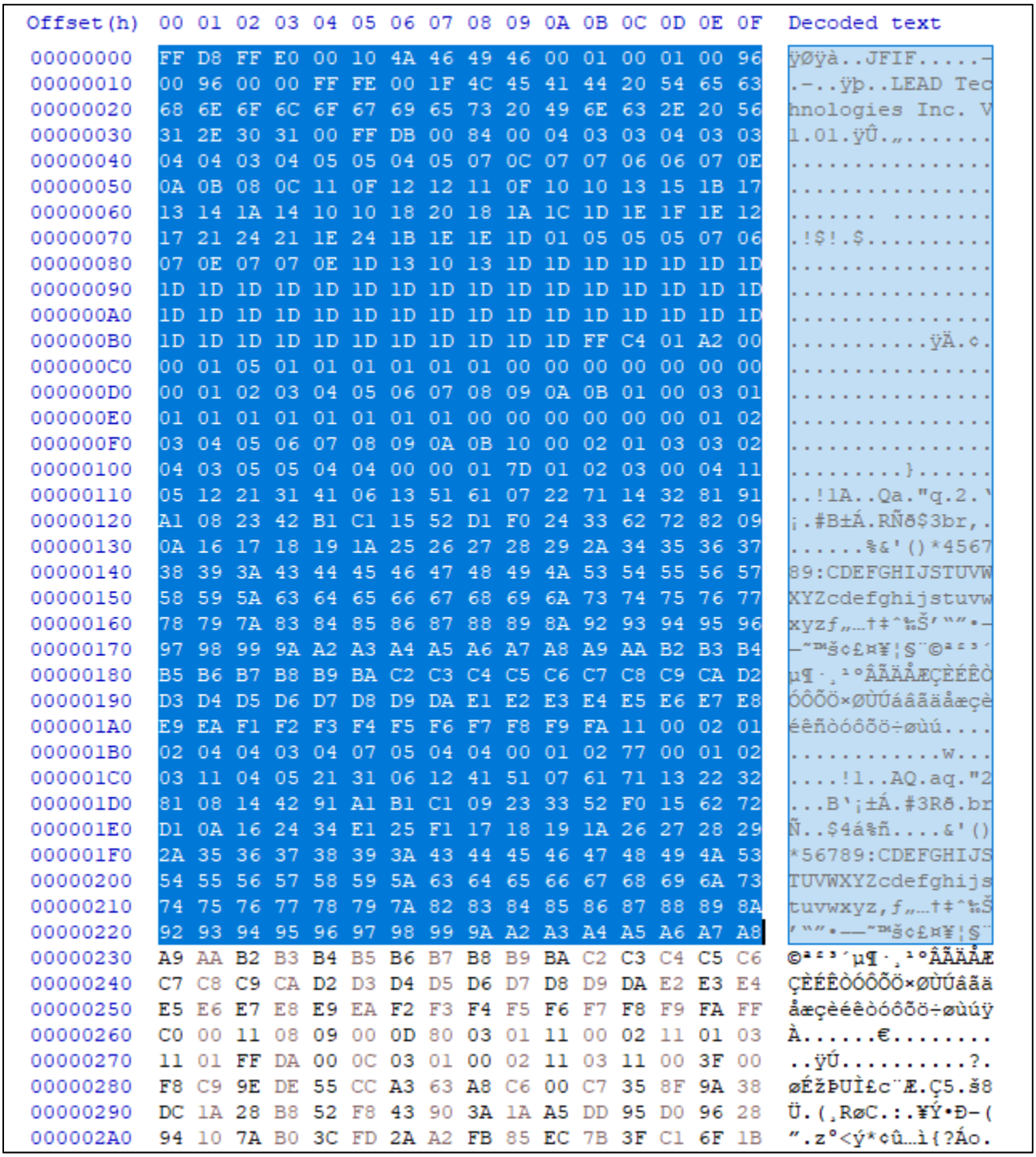
3. **BEFORE DOING ANTHING WITH ANY INDIVIDUAL FILES, IT WOULD BE BEST TO CREATE A COPY AND WORK ON THE COPY. YOU ARE MAKING CHANGES TO THESE FILES THAT YOU MAY NOT BE ABLE TO REVERSE**. As a last resort, you can always re-extract an original copy from your SANS provided USB drive.

OPTIONAL ADVANCED EXERCISE 2

1. Close any open files, folders, and program windows, and locate the **PhotoCarve3.jpg** file that you created in a previous part of this exercise. Click once on it to highlight it, then press **Ctrl+C** to copy, and then **Ctrl+V** to paste. You should now see a copy of that file. We will be working with this copy.

2. Open the **HxD** program. Using **File → Open**, point the program to the copy you just created, and open it in **HxD**.

3. Sweep the first **35** rows of hex characters. This equals **560** bytes of data.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	00	01	00	96	ÿØÿà..JFIF.....–
00000010	00	96	00	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	.–..ÿþ..LEAD Tec
00000020	68	6E	6F	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	hnologies Inc. V
00000030	31	2E	30	31	00	FF	DB	00	84	00	04	03	03	04	03	03	1.01.ÿÛ.„.....
00000040	04	04	03	04	05	05	04	05	07	0C	07	07	06	06	07	0E	.....
00000050	0A	0B	08	0C	11	0F	12	12	11	0F	10	10	13	15	1B	17	.....
00000060	13	14	1A	14	10	10	18	20	18	1A	1C	1D	1E	1F	1E	12	....
00000070	17	21	24	21	1E	24	1B	1E	1E	1D	01	05	05	05	07	06	.!$!.$.....
00000080	07	0E	07	07	0E	1D	13	10	13	1D	1D	1D	1D	1D	1D	1D	.....
00000090	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	.....
000000A0	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	.....
000000B0	1D	1D	1D	1D	1D	1D	1D	1D	1D	1D	FF	C4	01	A2	00		.....ÿÄ.¢.
000000C0	00	01	05	01	01	01	01	01	01	00	00	00	00	00	00	00	.....
000000D0	00	01	02	03	04	05	06	07	08	09	0A	0B	01	00	03	01	.....
000000E0	01	01	01	01	01	01	01	01	00	00	00	00	00	00	01	02	.....
000000F0	03	04	05	06	07	08	09	0A	0B	10	00	02	01	03	03	02	.....
00000100	04	03	05	05	04	04	00	00	01	7D	01	02	03	00	04	11	.....}.....
00000110	05	12	21	31	41	06	13	51	61	07	22	71	14	32	81	91	..!1A..Qa."q.2.'
00000120	A1	08	23	42	B1	C1	15	52	D1	F0	24	33	62	72	82	09	¡.#B±Á.RÑð$3br,.
00000130	0A	16	17	18	19	1A	25	26	27	28	29	2A	34	35	36	37	.....%&'()*4567
00000140	38	39	3A	43	44	45	46	47	48	49	4A	53	54	55	56	57	89:CDEFGHIJSTUVW
00000150	58	59	5A	63	64	65	66	67	68	69	6A	73	74	75	76	77	XYZcdefghijstuvw
00000160	78	79	7A	83	84	85	86	87	88	89	8A	92	93	94	95	96	xyzƒ„…†‡ˆ‰Š'""•–
00000170	97	98	99	9A	A2	A3	A4	A5	A6	A7	A8	A9	AA	B2	B3	B4	—˜™š¢£¤¥¦§¨©ª²³´
00000180	B5	B6	B7	B8	B9	BA	C2	C3	C4	C5	C6	C7	C8	C9	CA	D2	µ¶·¸¹ºÂÃÄÅÆÇÈÉÊÒ
00000190	D3	D4	D5	D6	D7	D8	D9	DA	E1	E2	E3	E4	E5	E6	E7	E8	ÓÔÕÖ×ØÙÚáâãäåæçè
000001A0	E9	EA	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	11	00	02	01	éêñòóôõö÷øùú....
000001B0	02	04	04	03	04	07	05	04	04	00	01	02	77	00	01	02	..........w...
000001C0	03	11	04	05	21	31	06	12	41	51	07	61	71	13	22	32	....!1..AQ.aq."2
000001D0	81	08	14	42	91	A1	B1	C1	09	23	33	52	F0	15	62	72	...B'¡±Á.#3Rð.br
000001E0	D1	0A	16	24	34	E1	25	F1	17	18	19	1A	26	27	28	29	Ñ..$4á%ñ....&'()
000001F0	2A	35	36	37	38	39	3A	43	44	45	46	47	48	49	4A	53	*56789:CDEFGHIJS
00000200	54	55	56	57	58	59	5A	63	64	65	66	67	68	69	6A	73	TUVWXYZcdefghijs
00000210	74	75	76	77	78	79	7A	82	83	84	85	86	87	88	89	8A	tuvwxyz‚ƒ„…†‡ˆ‰Š
00000220	92	93	94	95	96	97	98	99	9A	A2	A3	A4	A5	A6	A7	A8	‘’“”•–—˜™š¢£¤¥¦§¨
00000230	A9	AA	B2	B3	B4	B5	B6	B7	B8	B9	BA	C2	C3	C4	C5	C6	©ª²³´µ¶·¸¹ºÂÃÄÅÆ
00000240	C7	C8	C9	CA	D2	D3	D4	D5	D6	D7	D8	D9	DA	E2	E3	E4	ÇÈÉÊÒÓÔÕÖ×ØÙÚâãä
00000250	E5	E6	E7	E8	E9	EA	F2	F3	F4	F5	F6	F7	F8	F9	FA	FF	åæçèéêòóôõö÷øùúÿ
00000260	C0	00	11	08	09	00	0D	80	03	01	11	00	02	11	01	03	À.....€.....
00000270	11	01	FF	DA	00	0C	03	01	00	02	11	03	11	00	3F	00	..ÿÚ.....?.
00000280	F8	C9	9E	DE	55	CC	A3	63	A8	C6	00	C7	35	8F	9A	38	øÉžÞUÌ£c¨Æ.Ç5.š8
00000290	DC	1A	28	B8	52	F8	43	90	3A	1A	A5	DD	95	D0	96	28	Ü.(¸RøC.:.¥Ý•Ð–(
000002A0	94	10	7A	B0	3C	FD	2A	A2	FB	85	EC	7B	3F	C1	6F	1B	”.z°<ý*¢û…ì{?Áo.

4. Now delete it. What??? Then click on the save button in **HxD**. Go to your **VM Desktop** and try to open the file. Of course, it will not open.

5. Go back to **HxD** and now **File → Open** and point to **C:\Cases\Exercises\6.3**. Inside this directory is a file entitled **AnyPhoto.jpg**.

6. Open **AnyPhoto.jpg** and sweep **35** rows of data from it. Copy this and return to the **PhotoCarve3-Copy.jpg** workspace in **HxD** and paste it at the beginning of the data.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	FF	D8	FF	E0	00	10	4A	46	49	46	00	01	00	01	00	F0	ÿØÿà..JFIF.....ð
00000010	00	F0	00	00	FF	FE	00	1F	4C	45	41	44	20	54	65	63	.ð..ÿþ..LEAD Tec
00000020	68	6E	6F	6C	6F	67	69	65	73	20	49	6E	63	2E	20	56	hnologies Inc. V
00000030	31	2E	30	31	00	FF	DB	00	84	00	05	05	05	08	05	08	1.01.ÿÛ.„.....
00000040	0C	07	07	0C	0C	09	09	09	0C	0D	0C	0C	0C	0C	0D	0D	.....
00000050	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	.....
00000060	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	.....
00000070	0D	0D	0D	0D	0D	0D	0D	0D	0D	01	05	08	08	0A	07		.....
00000080	0A	0C	07	07	0C	0D	0C	0A	0C	0D	0D	0D	0D	0D	0D	0D	.....
00000090	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	.....
000000A0	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	.....
000000B0	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	0D	FF	C4	01	A2	00	.....ÿÄ.¢.
000000C0	00	01	05	01	01	01	01	01	01	00	00	00	00	00	00	00	.....
000000D0	00	01	02	03	04	05	06	07	08	09	0A	0B	01	00	03	01	.....
000000E0	01	01	01	01	01	01	01	01	00	00	00	00	00	00	01	02	.....
000000F0	03	04	05	06	07	08	09	0A	0B	10	00	02	01	03	03	02	.....
00000100	04	03	05	05	04	04	00	00	01	7D	01	02	03	00	04	11	.....}.....
00000110	05	12	21	31	41	06	13	51	61	07	22	71	14	32	81	91	..!1A..Qa."q.2.'
00000120	A1	08	23	42	B1	C1	15	52	D1	F0	24	33	62	72	82	09	¡.#B±Á.RÑð$3br,.
00000130	0A	16	17	18	19	1A	25	26	27	28	29	2A	34	35	36	37	......%&'()*4567
00000140	38	39	3A	43	44	45	46	47	48	49	4A	53	54	55	56	57	89:CDEFGHIJSTUVW
00000150	58	59	5A	63	64	65	66	67	68	69	6A	73	74	75	76	77	XYZcdefghijstuvw
00000160	78	79	7A	83	84	85	86	87	88	89	8A	92	93	94	95	96	xyzƒ„…†‡ˆ‰Š'"•–
00000170	97	98	99	9A	A2	A3	A4	A5	A6	A7	A8	A9	AA	B2	B3	B4	—˜™š¢£¤¥¦§¨©ª²³´
00000180	B5	B6	B7	B8	B9	BA	C2	C3	C4	C5	C6	C7	C8	C9	CA	D2	µ¶·¸¹ºÂÃÄÅÆÇÈÉÊÒ
00000190	D3	D4	D5	D6	D7	D8	D9	DA	E1	E2	E3	E4	E5	E6	E7	E8	ÓÔÕÖ×ØÙÚáâãäåæçè
000001A0	E9	EA	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	11	00	02	01	éêñòóôõö÷øùú....
000001B0	02	04	04	03	04	07	05	04	04	00	01	02	77	00	01	02	..........w...
000001C0	03	11	04	05	21	31	06	12	41	51	07	61	71	13	22	32	....!1..AQ.aq."2
000001D0	81	08	14	42	91	A1	B1	C1	09	23	33	52	F0	15	62	72	...B'¡±Á.#3Rð.br
000001E0	D1	0A	16	24	34	E1	25	F1	17	18	19	1A	26	27	28	29	Ñ..$4á%ñ....&'()
000001F0	2A	35	36	37	38	39	3A	43	44	45	46	47	48	49	4A	53	*56789:CDEFGHIJS
00000200	54	55	56	57	58	59	5A	63	64	65	66	67	68	69	6A	73	TUVWXYZcdefghijs
00000210	74	75	76	77	78	79	7A	82	83	84	85	86	87	88	89	8A	tuvwxyz‚ƒ„…†‡ˆ‰Š
00000220	92	93	94	95	96	97	98	99	9A	A2	A3	A4	A5	A6	A7	A8	'"•–—˜™š¢£¤¥¦§¨
00000230	A9	AA	B2	B3	B4	B5	B6	B7	B8	B9	BA	C2	C3	C4	C5	C6	©ª²³´µ¶·¸¹ºÂÃÄÅÆ
00000240	C7	C8	C9	CA	D2	D3	D4	D5	D6	D7	D8	D9	DA	E2	E3	E4	ÇÈÉÊÒÓÔÕÖ×ØÙÚâãä
00000250	E5	E6	E7	E8	E9	EA	F2	F3	F4	F5	F6	F7	F8	F9	FA	FF	åæçèéêòóôõö÷øùúÿ
00000260	C0	00	11	08	09	00	0D	80	03	01	11	00	02	11	01	03	À.....€.....
00000270	11	01	FF	DA	00	0C	03	01	00	02	11	03	11	00	3F	00	..ÿÚ.....?.
00000280	F8	C9	9E	DE	55	CC	A3	63	A8	C6	00	C7	35	8F	9A	38	øÉžÞUÌ£c¨Æ.Ç5.š8
00000290	DC	1A	28	B8	52	F8	43	90	3A	1A	A5	DD	95	D0	96	28	Ü.(¸RøC.:.¥Ý•Ð–(
000002A0	94	10	7A	B0	3C	FD	2A	A2	FB	85	EC	7B	3F	C1	6F	1B	”.z°<ý*¢û…ì{?Áo.
000002B0	68	BF	0E	FC	59	16	AB	76	B2	5C	83	6A	D1	F9	42	2C	h¿.üY.«v²\ƒjÑùB,
000002C0	90	E7	A6	33	C6	3A	73	5F	47	81	C5	27	0F	63	17	FD	.ç¦3Æ:s_G.Å'.c.ý
000002D0	6E	71	E2	39	9E	A8	F7	CD	37	C1	7A	77	8F	23	D4	7C	nqâ9ž¨÷Í7Ázw.#Ô|
000002E0	5B	E3	DB	88	61	D5	2E	06	EB	4B	16	00	66	3C	7C	A7	[ãÛˆaÕ..ëK..f<|§
000002F0	1D	F8	AE	D9	52	54	DE	8A	EC	E3	8C	F9	75	3C	2B	E2	.ø®ÙRTÞŠìãŒùu<+â
00000300	DC	5A	8C	1A	62	58	C3	1C	90	E8	E9	26	F4	B7	78	F6	ÜZŒ.bXÃ..èé&ô·xö

7. Click the **Save** icon in **HxD**, and then navigate to the file on the **Desktop** and see if it now opens. It should, if you did everything correctly.

8. Go back into **HxD** to the **PhotoCarve3 - Copy.jpg** file that you have been working with and grab the scroll bar on the right of the **Decoded text** field with your mouse. Pull it down so it is about ¾ of the way down the window. Now anywhere in the hex, start sweeping, and sweep all the way to the end of the file.

9. Delete everything you have just swept.

10. Click on the **Save** icon in **HxD** to save what you have just done. Navigate to the **VM Desktop** and try to open the file. A .**JPG** is a type of file that will open and show whatever is still available, as long as the header works. You can see that the photo cuts off at the point that you deleted the data.

Exercise—Key Takeaways

- Data carving and rebuilding can be very time consuming and frustrating.
- Sometimes half the battle is knowing where to cut and paste from one file to another.
- Even a partial file can contain data relevant to your case.