CSU, College of Engineering and Computer Science
Department of Computer Science
CSC 153 – Computer Forensics Principles and Practice

Catherine Nguyen
CSC 153
10/25/2020

Lab 3: Examining NTFS Disks

Objectives

- Become familiar with the WinHex forensics tool.
- Use WinHex to explore the MFT and be able to analyze both resident and non-resident files.

Part 1: Explore the MFT of a file.

To begin we first need to create a text file names 'forensicsclass.txt' onto our workstation and in that file type "We will have a forensics class on Monday".

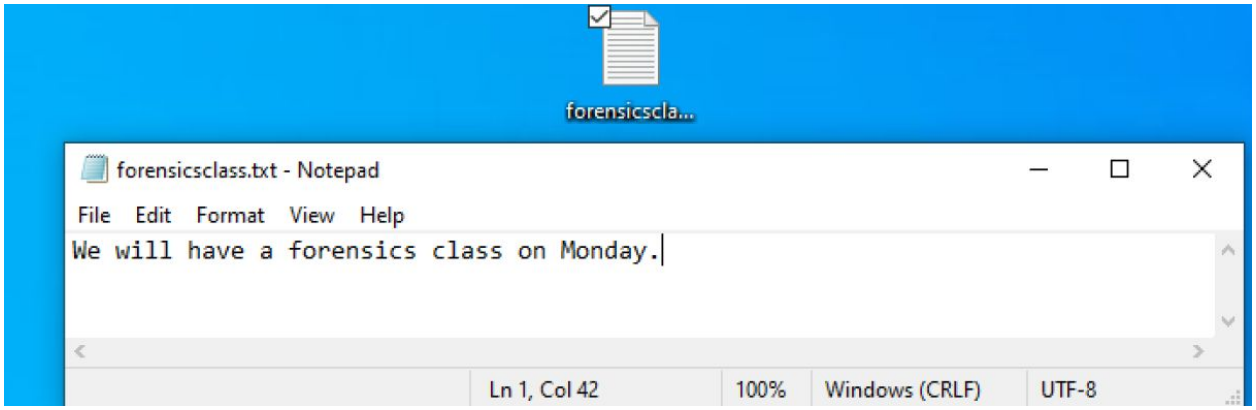

Figure 1: Creating a forensicsclass.txt file.

Next, we append an alternate data stream containing 'If you study hard, then you are likely to succeed' to the file via the command 'below. echo If you study hard, then you are likely to succeed > forensicsclass.txt:secret'. Then We can now display that data stream via the command more < forensicsclass.txt:secret.

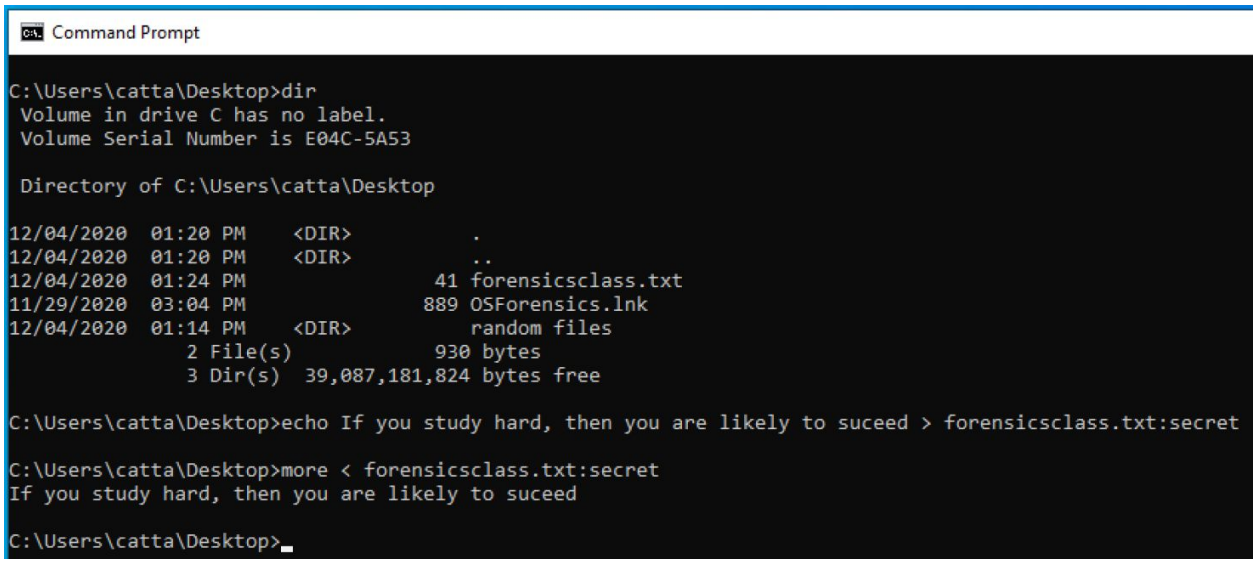

Figure 2: Appending alternative data secret and showing more of forensicsclass.txt file.

Next, we need to examine the metadata of the forensicsclass.txt file stores in the $MFT file by staring up WinHex as administrator. Then click on options, Edit Mode, and select Read-Only Mode in the dialog box, then click OK.

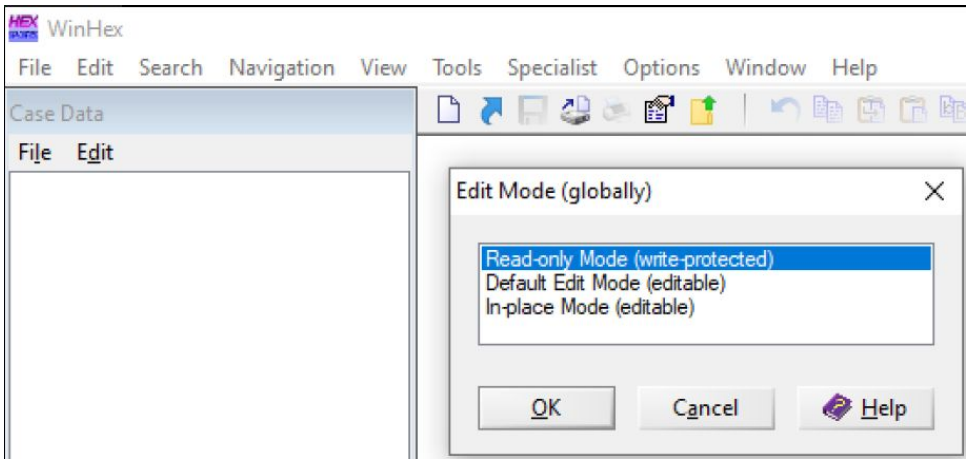

Figure 3: Setting WinHex to Read-Only Mode.

After that change has been made we can click on tools, open disk, and select the drive that has our forensicsclass.txt file then click OK. If you're prompted to take a new snapshot, click Take a new one. This will begin the transversion

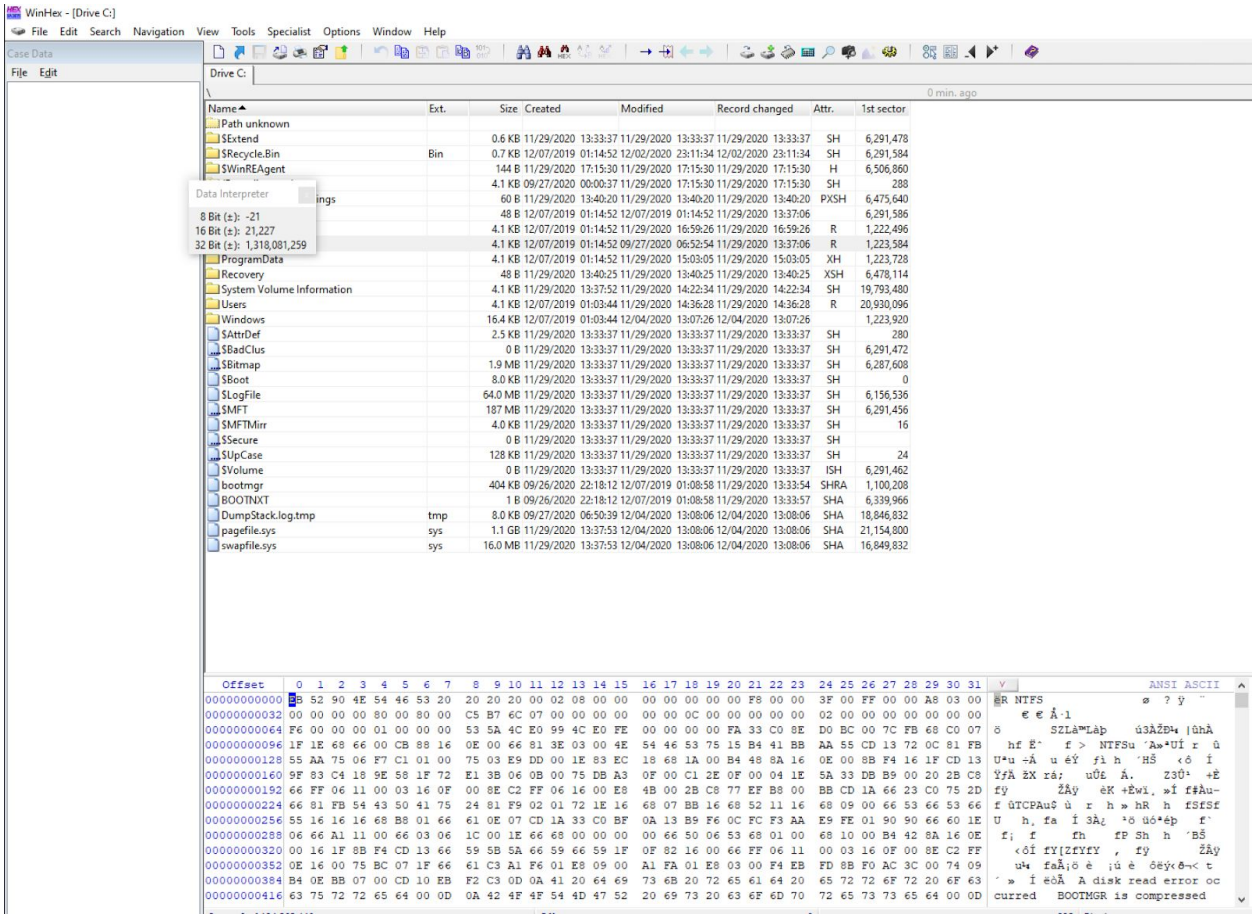

Figure 4: Traversing the C: drive with forensicsclass.txt file.

Once the traversing is complete, click on options, Data Interpreter from the menu. Then in the Data Interpreter Options dialogue box, and click on Win32 FILETIME (64bit) check box and press OK. This will then display the FILETIME.

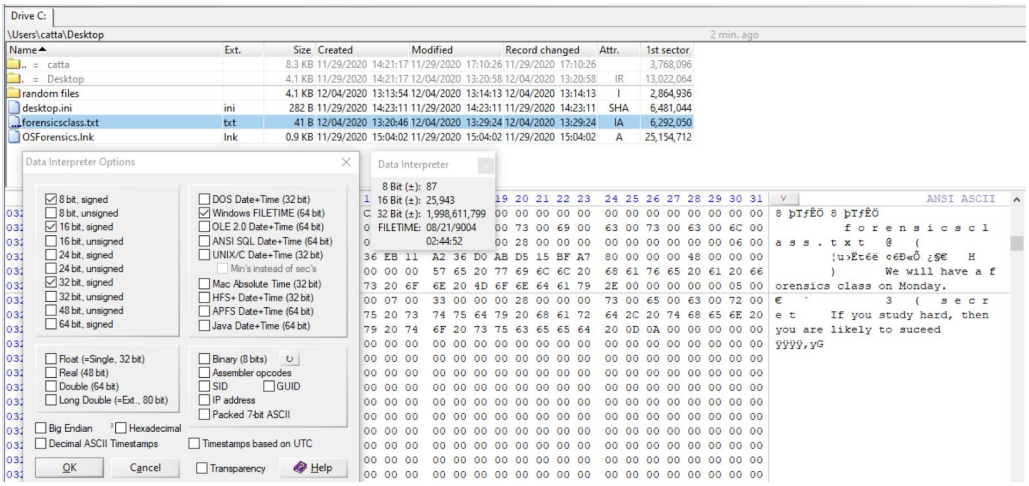

Figure 5: Data Interpreter Option includes Win32 FILETIME (64 bit).

Now we need to find the directory with out forensicsclass.txt file in WinHex. In the upper-right pane of WinHex, scroll down until you see your working directory. Double-click each folder in the path and then click the forensicsclass.txt file. After that, we click at the beginning of the record, on the letter F in FILE, and then drag down and to the right while monitoring the hexadecimal counter in the lower-right corner. For example, the start of attribute 0x10 is at offset 0x38 from the beginning of the MFT record. To find the start of attribute 0x10, drag the cursor until the counter reaches 38. When the counter reaches 38, release the mouse button.

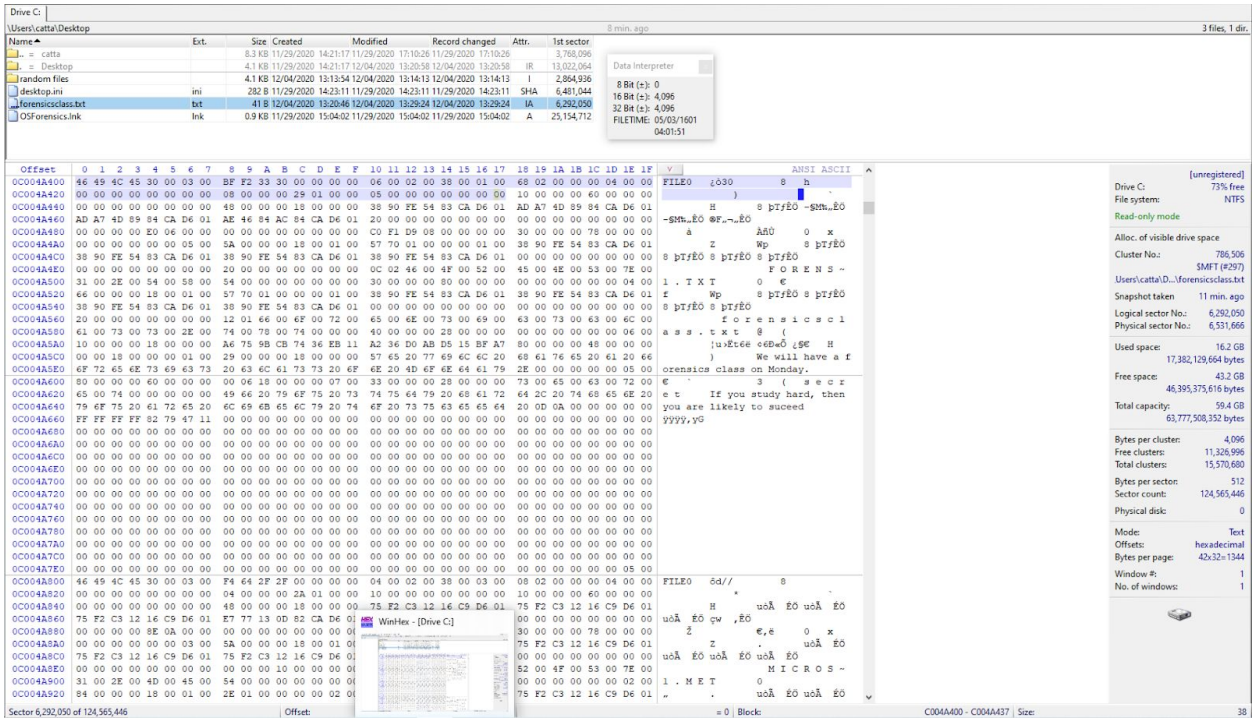

Figure 6: Select forensicsclass.txt, attribute 0x01 at offset 0x38 from the start of the MFT record.

Move the cursor one position to the next byte and then you may start to analyze attribute 0x10. Recall what we learned in class, the file creates date and time can be found at offset 0x18 to 0x1F from the beginning of attribute 0x10. Use a similar method as in step 7 to find the file creation date and time for forensicsclass.txt. Refer to your handout for the attribute details. In the same manner, we used above, we can determine the files created date and time to be 12/04/20 21:20:46.

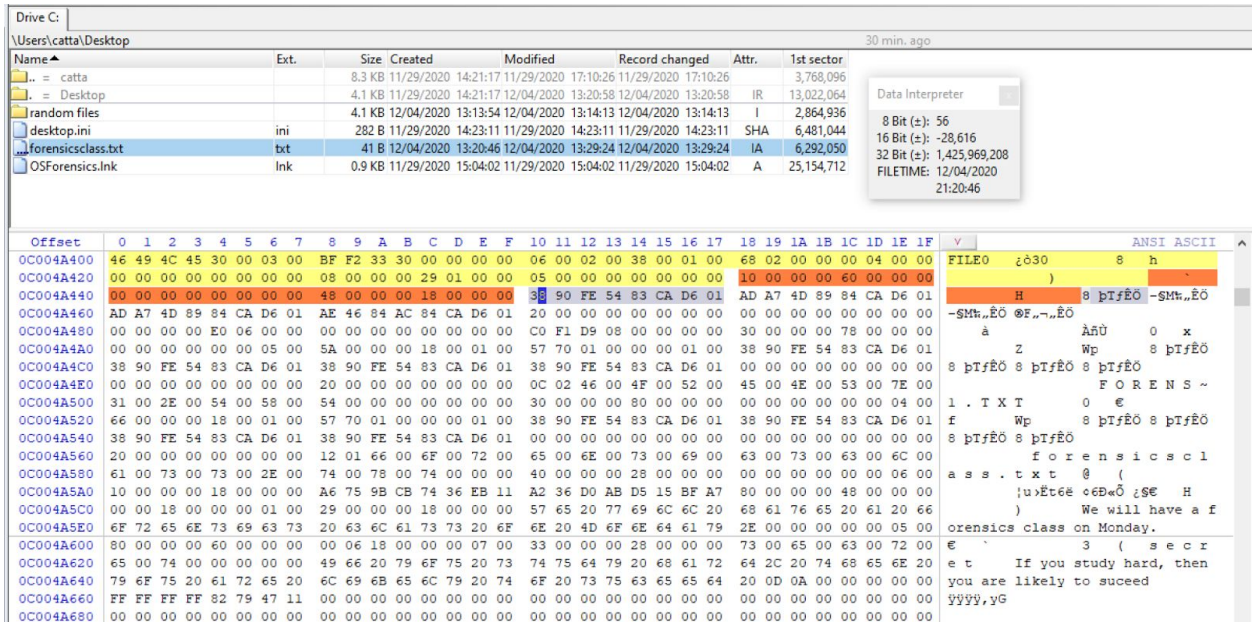

Figure 7: File created date and time for forensicsclass.txt with Tlimestamp UTC unselected.

Questions for Part 1:

1. **According to the data interpreter, what are the file creation date and time for the file forensicsclass.txt?.**
    a. The created date and time for forensicsclass.txt are 12/04/20 21:20:26. It's found from offset 0x18 to 0x1F from the beginning of the attribute 0x10. It should be noted that this is with the box for Timestamps based on UTC unselected.

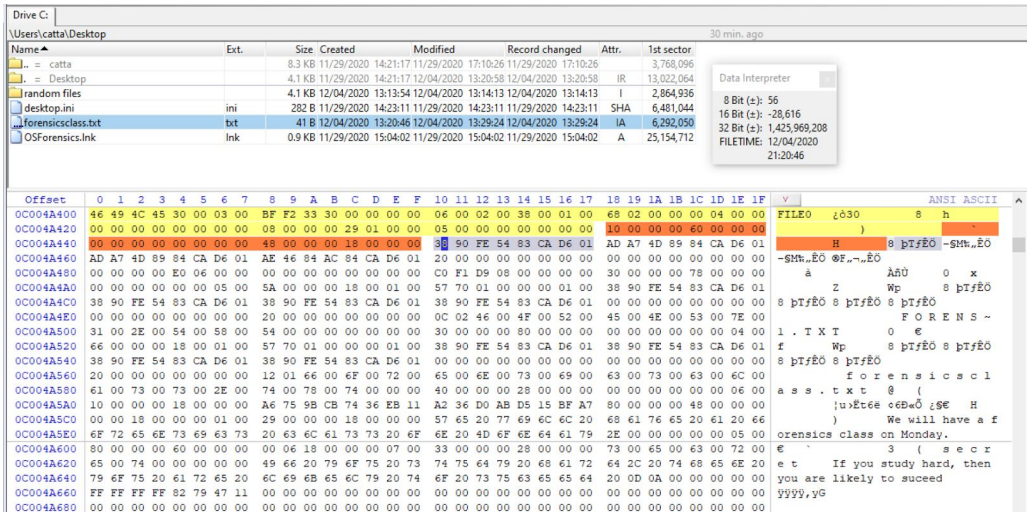

Figure 8: Creation date and time for forensicsclass.txt Tlimestamp UTC unselected.

b. The created date and time for forensicsclass.txt are 12/04/20 13:20:26. It's found from offset 0x18 to 0x1F from the beginning of the attribute 0x10. It should be noted that this is with the box for Timestamps based on UTC selected.

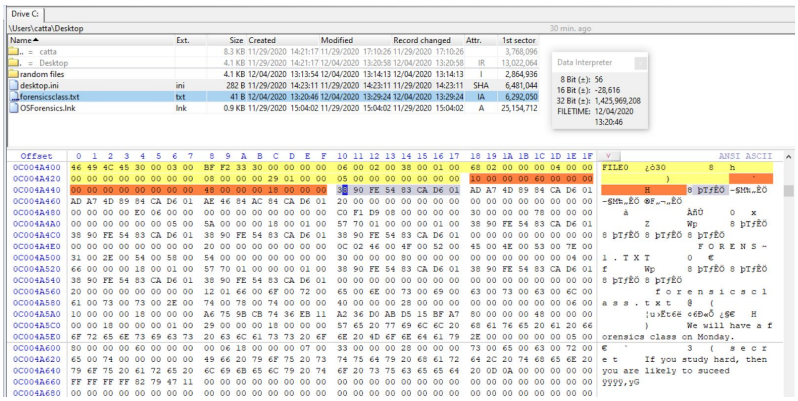

Figure 9: Creation date and time for forensicsclass.txt Tlimestamp UTC unselected.

2. What is the size of the MFT record?

a. The size of the MTF record is, in big-endian, 00 04 00 00. We can find this information from offset 0x1C to 0x1F from attribute 0x00.

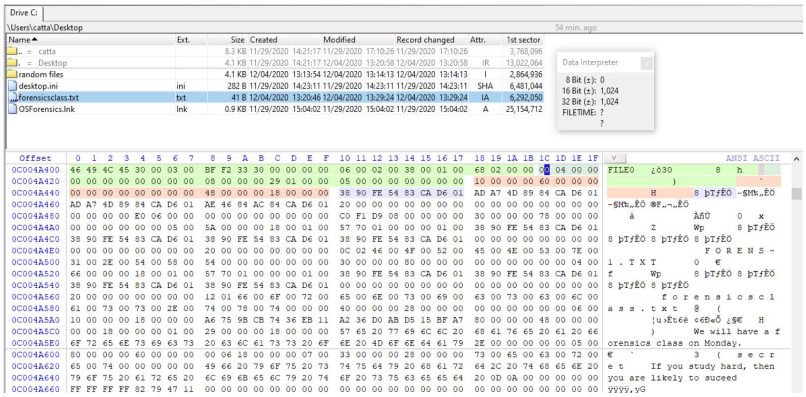

Figure 10: The size of the MTF record for forensicsclass.txt is 0x0400.

3. What is the length of the header?

a. The header length for the MFT record is 0x38. This can be found at offset 0x14 from attribute 0x00.

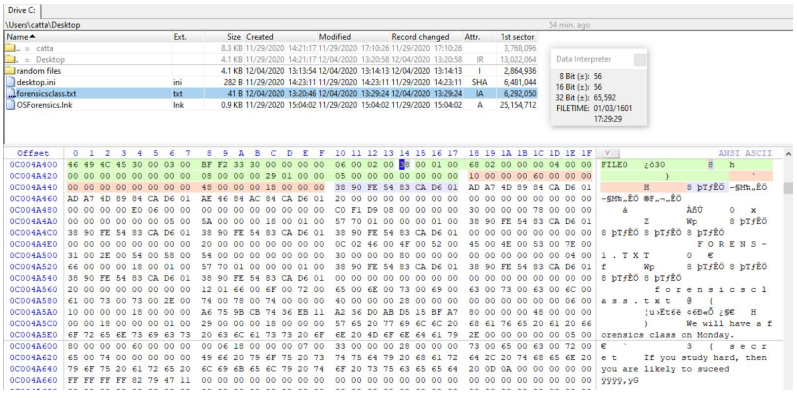

Figure 11: The length of the MTF record header for forensicsclass.txt is 0x38.

4. **What are the file's last modified date and time?**
    a. The file's last modified date and time are 12/04/20 12:20:46. We can find that information from offset 0x20 to 0x27 of attribute 0x10. It should be noted that this is with the box for Timestamps based on UTC selected.

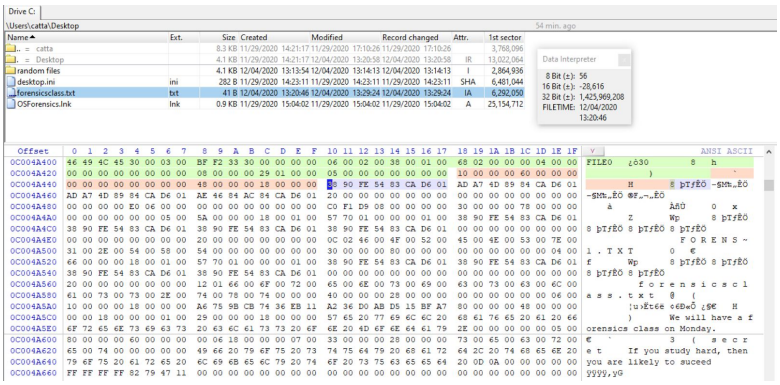

Figure 12: File last modified date and time for forensicsclass.txt.

5. **How many 0x30 attributes does this file have? Why?**
    a. There are two attributes 0x30s'. This is because our file name is longer than 8 characters, so we have a short file name, and a long file name.

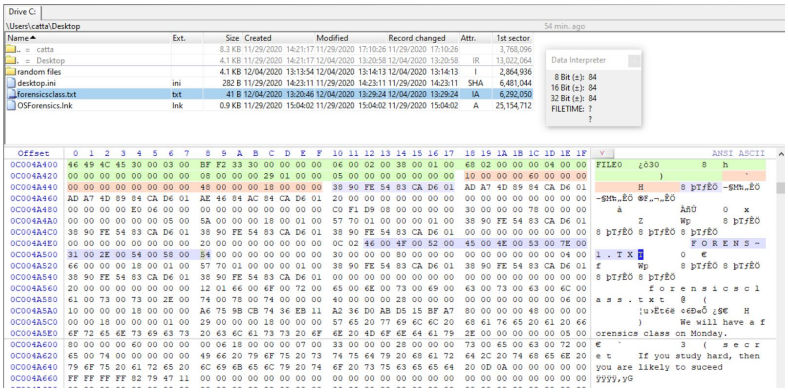

Figure 13: The short file name at 0x5A from the first 0x30 attribute.

    b. Long file names are found at offset 0x5A from the second 0x30 attribute. Our long file name is forensicsclass.txt.

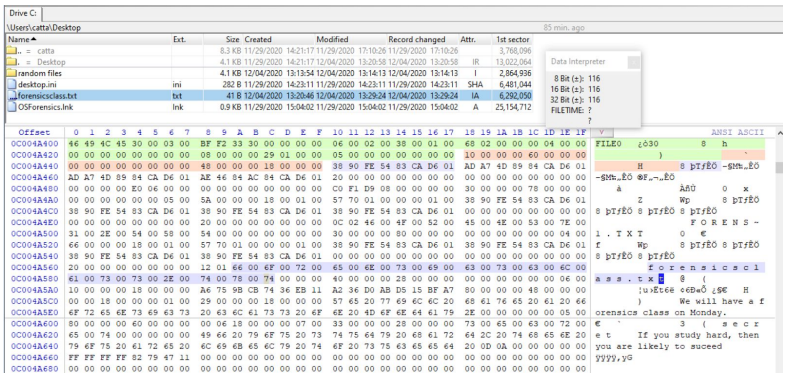

Figure 14: The long file name at 0x5A from the first 0x30 attribute.

6. **What is the name of this file?**
    a. As stated above there are two file names, a short file name, and long file name.
    b. Short file names are found at offset 0x5A from the first 0x30 attribute. Our short file name is FORENS~1.TXT shown in **Figure 13** above.
    c. Long file names are found at offset 0x5A from the second 0x30 attribute. Our long file name is forensicsclass.txt.shown in **Figure 14** above.

7. **Is this file a resident file or a nonresident file? Where can you find the evidence?**
    a. The resident/nonresident flag exists at offset 0x08 from attribute 0x80 . In this case, we can see it is a resident file. This makes sense because it is only 48 bytes in size.

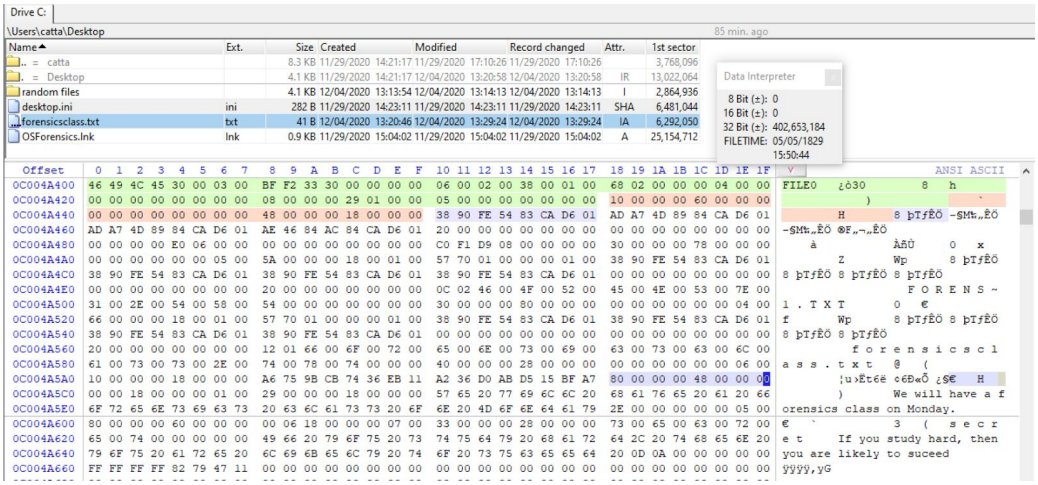

Figure 15: The resident/non-resident flag set to 0x00, meaning resident.

8. **Did you find the hidden message in the file when you check the MFT record?**
    a. Yes, it was not difficult to find the hidden message. It lies inside of a second 0x80 attribute and is easily found by looking at the ASCII screen on WinHex.

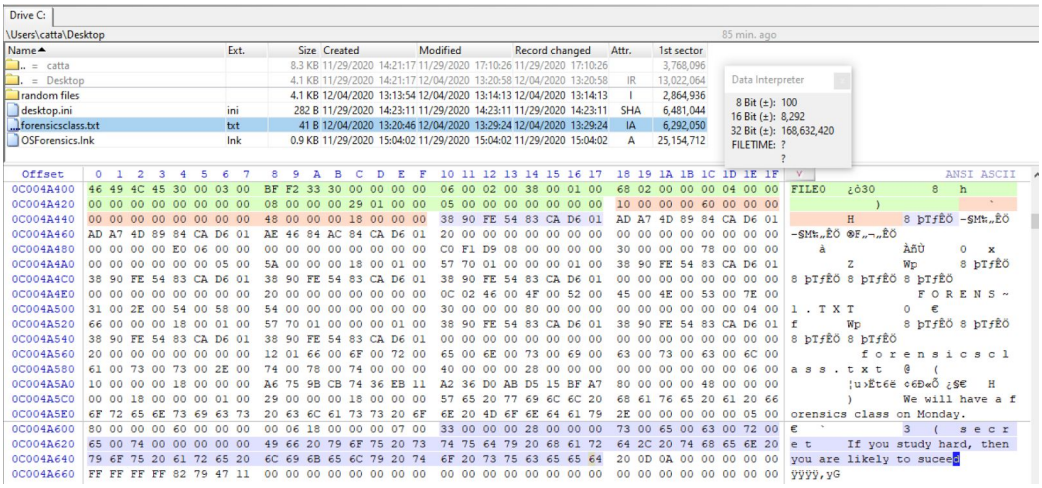

Figure 16: The secret message contained in the second 0x80 attribute.

b. More specifically, it's located in the data run for the second 0x80 attribute at offset 0x18.

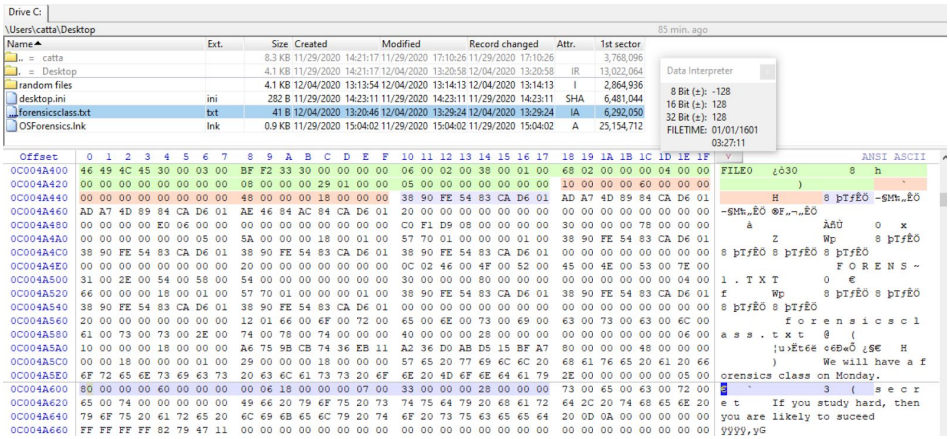

Figure 17: Secret message contained inside of the second 0x80 attribute.

9. How many 0x80 attributes does this file have? What is the possible reason?

a. 2, The reason for this would be the hidden data stream. This creates an additional 0x80 attribute for the stream. We can verify this by going to offset 0x18 for the second 0x80 attribute. This is where the data run is for resident files. This contains the secret message. As shown in **Figure 17** above.

Part 2: Analyze a given MFT record.

Given the MFT record below, please answer the questions from 11-16.

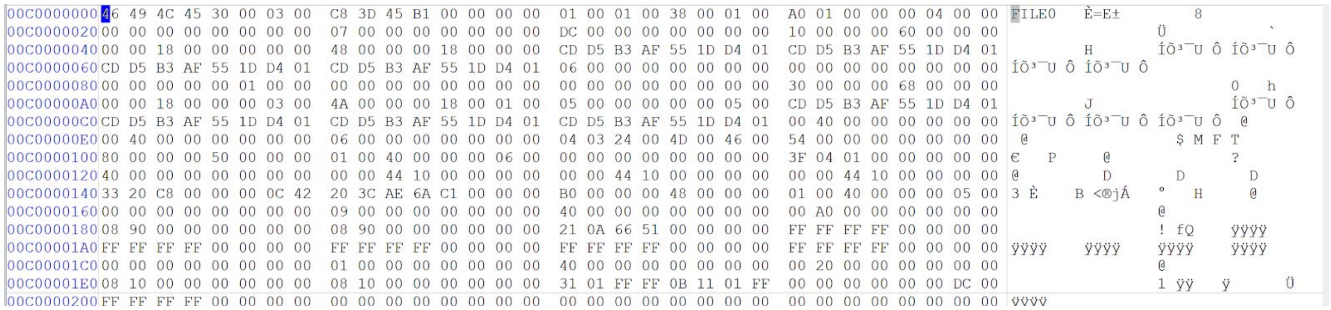

Figure 18: Photo for questions below.

Questions for Part 2:

10. Is this file a resident file or a nonresident file? Where can you find the evidence?

a. The resident/nonresident flag exists at offset 0x08 from attribute 0x80 . In this case, we can see it is a nonresident file, as the flag is 0x01

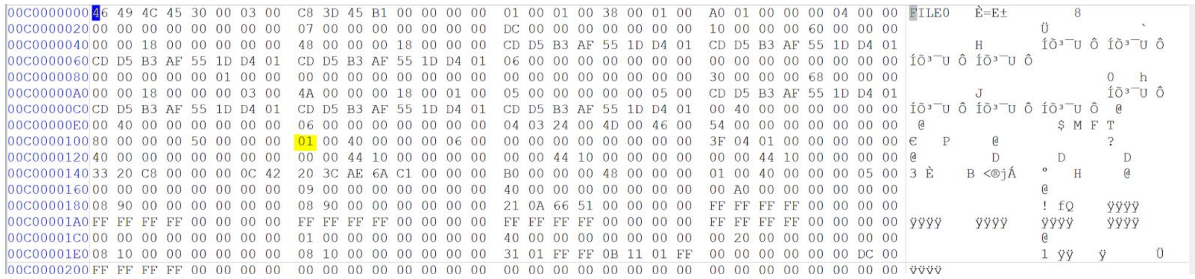

Figure 19: Nonresident file flag.

11. How many data runs does this file have?

a. This file has two data runs.

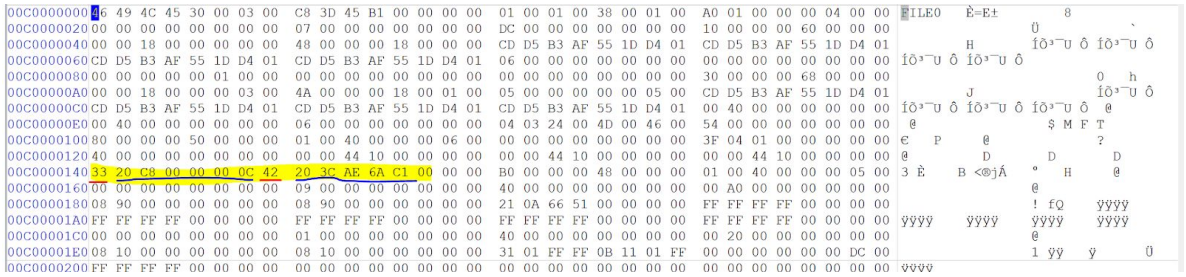

Figure 20: Bytes underlining start of data runs in red, and remainders in black.

12. What is the starting cluster address value for the first data run (LCN)? You don't need to calculate the result if you provide a math expression.

a. The starting cluster address value is 0x0C0000. We multiply this by the cluster size, which is 4096 in decimal or 0x1000 in hexadecimal. So the cluster address value for the first data run is 0x0C0000 * 0x1000 = 0xC00000000.

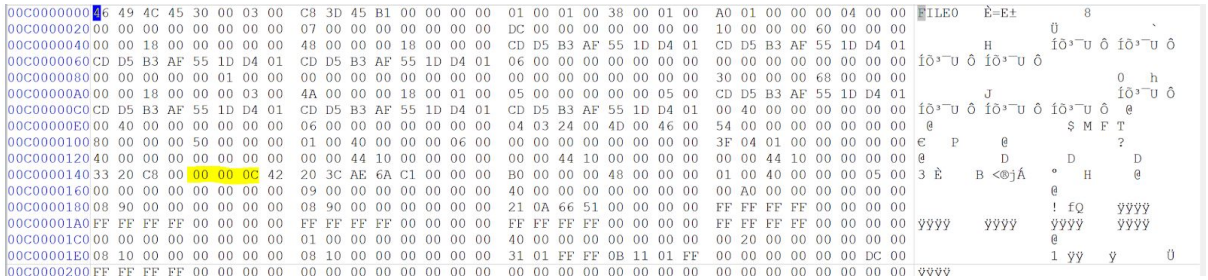

Figure 21: Starting LCN address for first data run.

13. How many clusters are assigned to the first data run?

a. The number of clusters assigned to the first data run is 0x00C820.

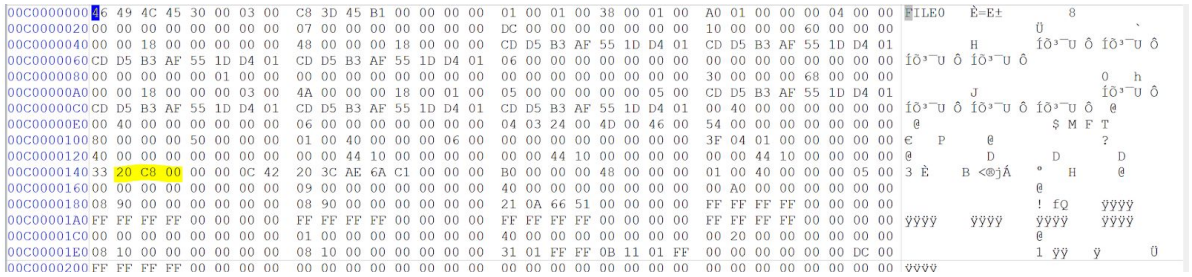

Figure 22: Number of clusters assigned to the first data run.

14. Does this file have other data runs? If yes, what is the starting cluster address value for the second data run (LCN)? You don't need to calculate the result if you provide a math expression.

a. Yes there is a second data run. The starting cluster address value is 0x00C16AAE. We multiply this by the cluster size, which is 4096 in decimal or 0x1000 in hexadecimal. So the cluster address value for the second data run is 0x00C16AAE * 0x1000 = 0xC16AAE0000.

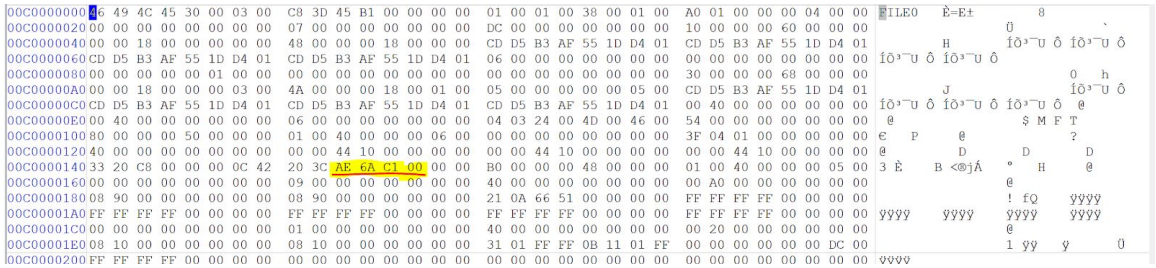

Figure 23: Starting LCN address for second data run.

15. How many clusters are assigned to the second data run?

a. The number of clusters assigned to the second data run is 0x3C20.

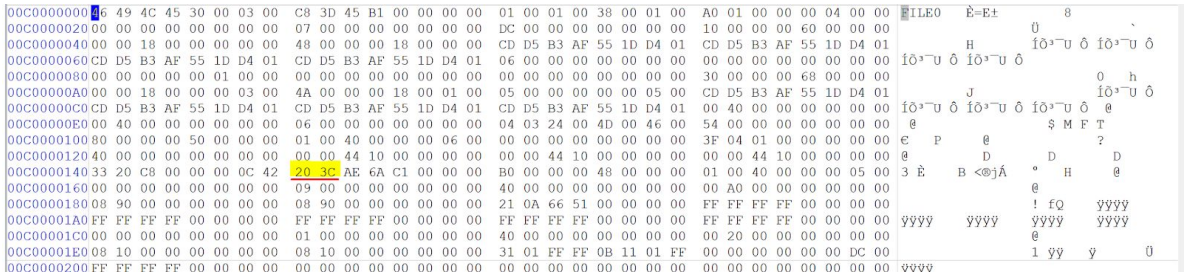

Figure 24: Number of clusters assigned to the second data run.