**DIGITAL FORENSICS LAB**

|  |  |
| --- | --- |
| Exercise 7 | |
| Name | S Shyam Sundaram |
| Registration Number | 19BCE1560 |
| Slot | L39+L40 |
| Faculty | Dr. Seshu Babu Pulagara |
| Date | 28th September, 2021 |

**AIM**

Comparing file structures with Hex editors.

**PART A**

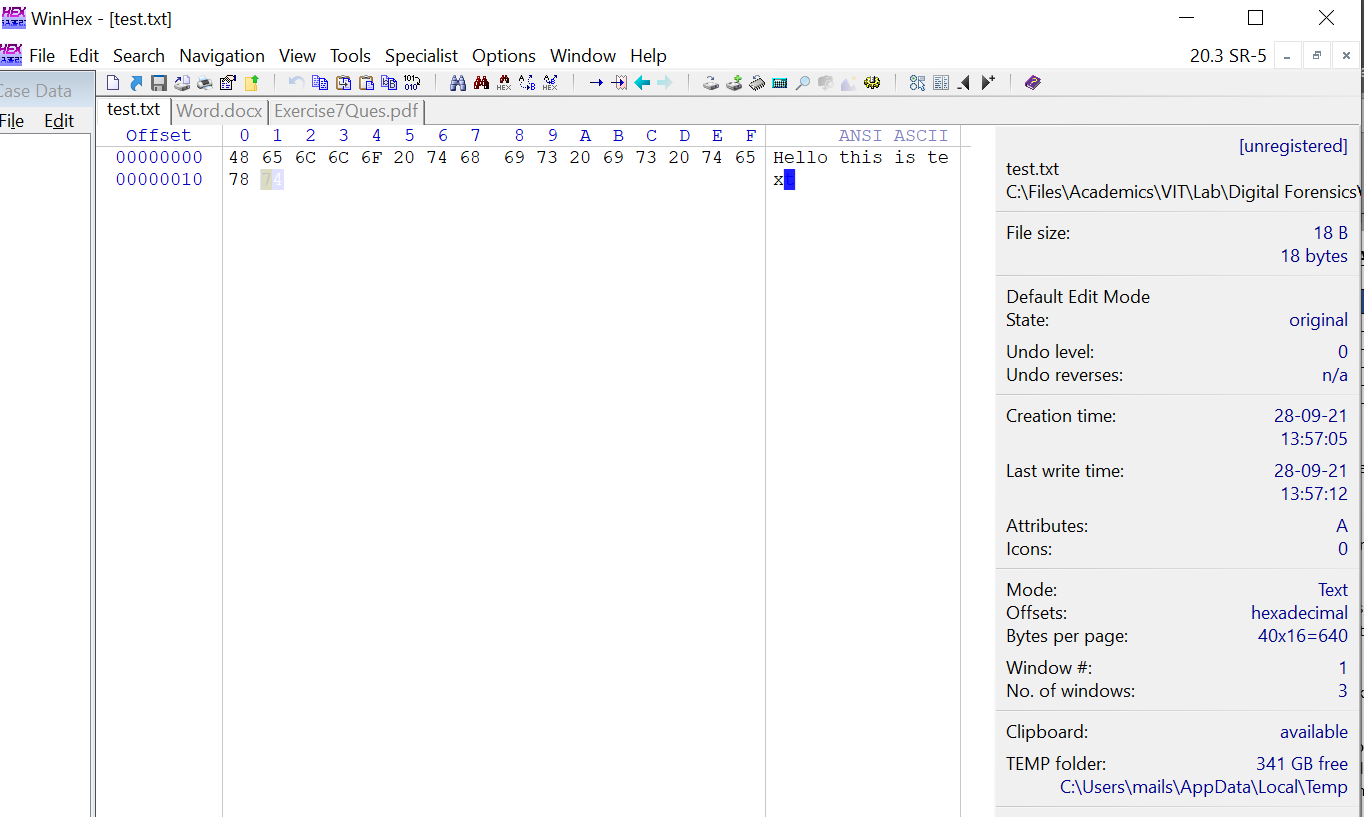
A few commands are executed and their outputs are shown below.

**Q1**

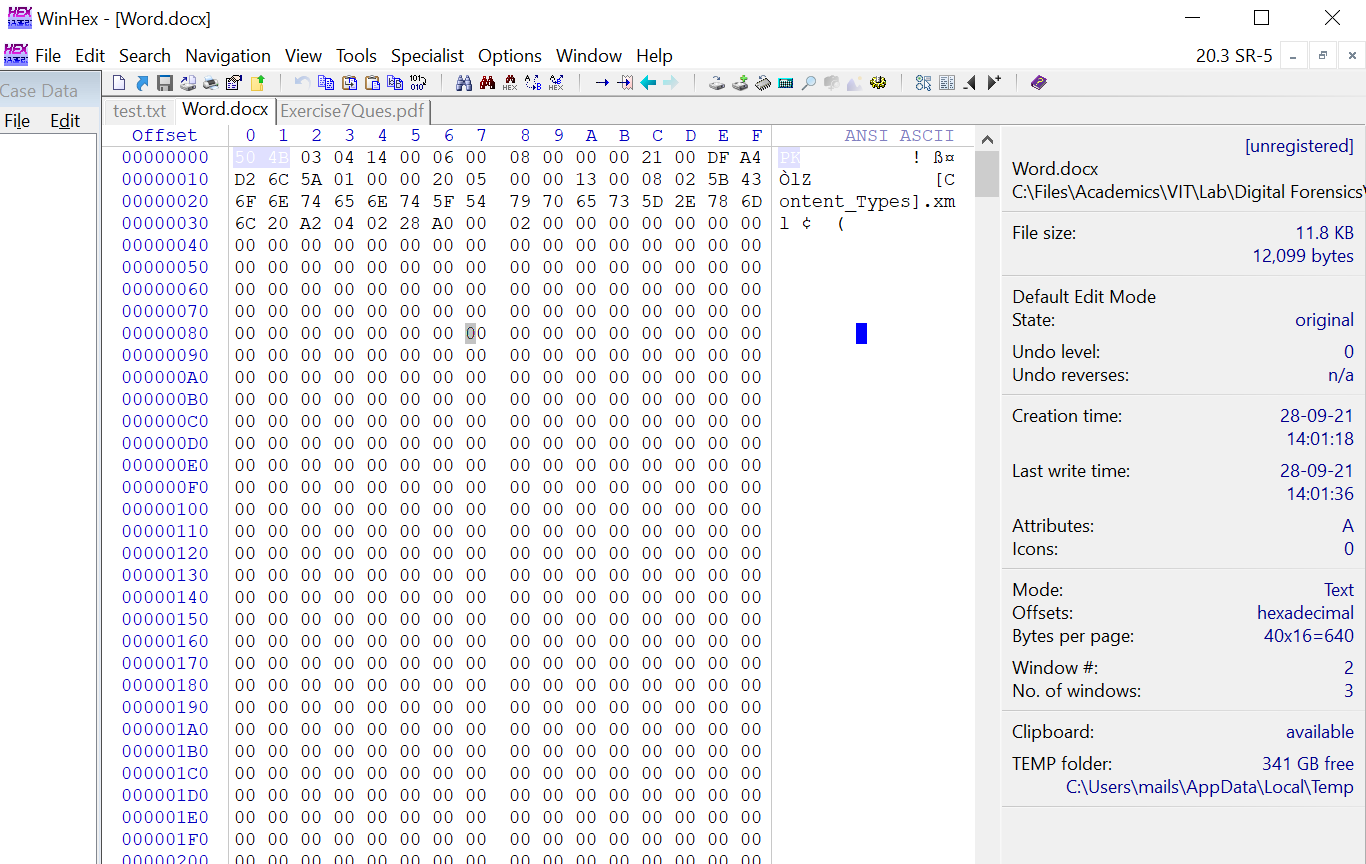
Create text files using these tools. Then use a Hex editor such as vim or WinHex to view these files. What similarities and differences do you notice?

**A**

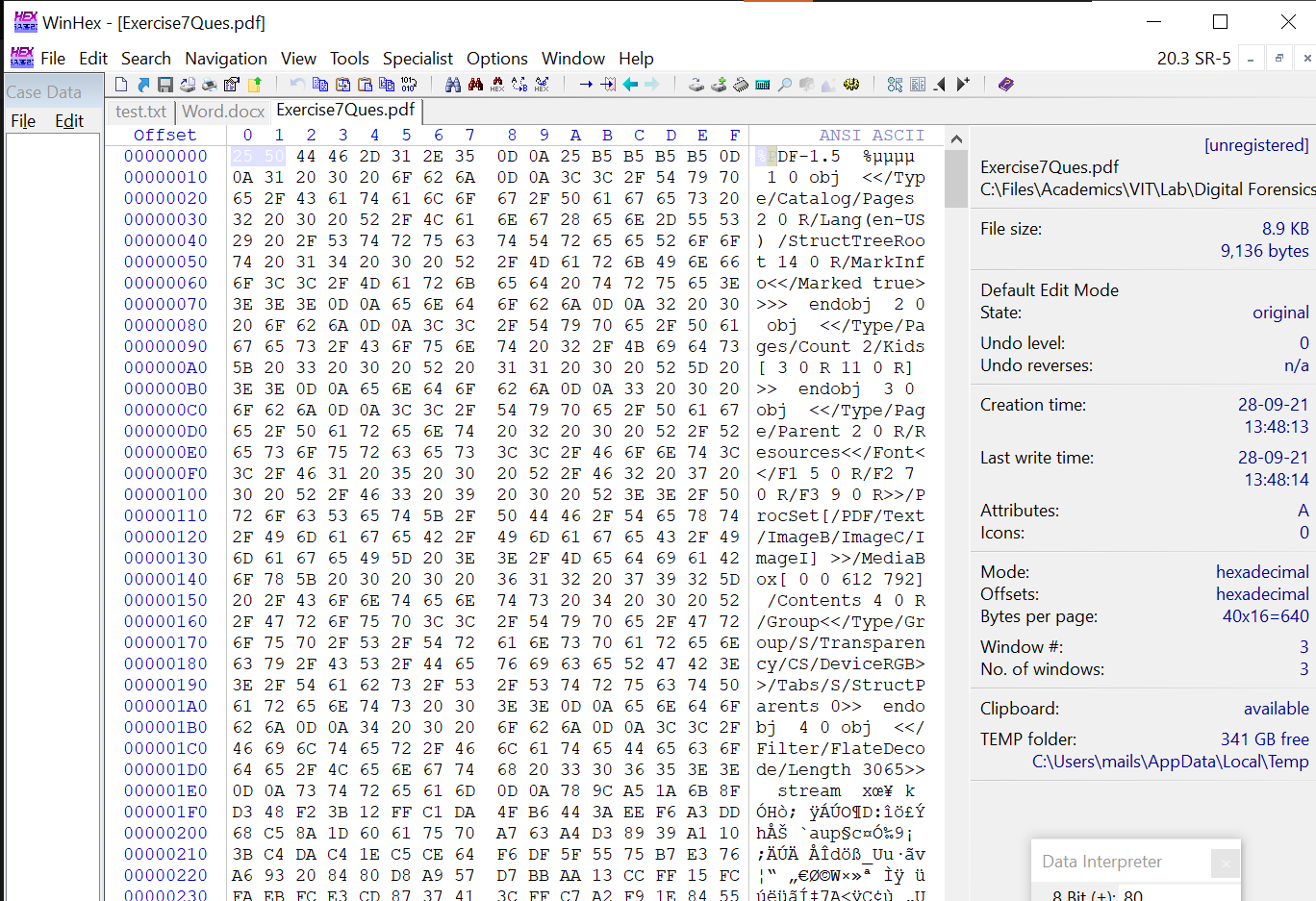
To open a file using WinHex, click ‘File’->’open’. Then browse for the file and open it. For this question, two forms of text files (one using Notepad and the other using MS Word) were created and a PDF file was also used. When opened in WinHex, the following was displayed on the window.



Made using notepad: test.txt



Made using MS Word: Word.docx



A PDF file: Exercise7Ques.pdf

The following observations are made which tell us the differences observed among the files when viewed using WinHex:

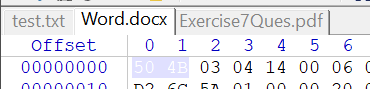
* The windows of each file show us the offset, the content, hexadecimal equivalent of the content and the ANSI ASCII form.
* For text file, the window is simple. We see the contents of test.txt directly on the window and its Hexadecimal equivalent.
* For word files, it is a bit more complicated. We do not see the text content of the docx file, rather, we see the XML files associated with the docx file.
* The same is the case for PDF. Unlike the txt file, we see the formatting used for the content within the PDF.

**Q2**

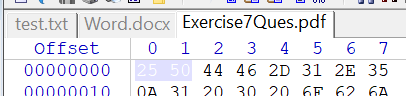
How can you tell what type of file you are looking at by what vim or WinHex shows in the Hex window?

**A**

We can verify the file type by looking at the first few hexadecimal characters as highlighted below.



DOCX File



PDF File

These few characters tell us what file we are working with.

’50 4B’ represents zip file format and formats based on it, such as DOCX, EPUB, JAR, ODF, OOXML etc.

’25 50 44 46 2D’ is the signature for PDF files.

**PART B**

Working with NTFS hidden streams.

**Q1**

Create a folder dirtysecret. (If one already exists, remove all its contents.) In the dirtysecret folder we first create a file and then a stream.

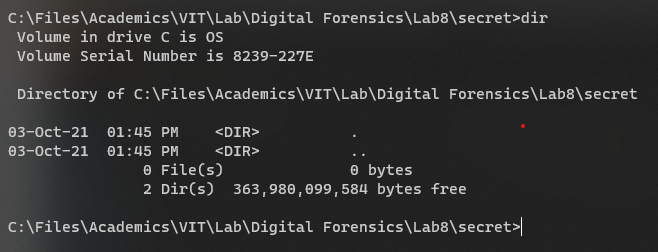
c:\dirtysecret echo "This is a file" > file.txt

c:\dirtysecret echo "This is another file" > file.txt:hiddenstream.txt

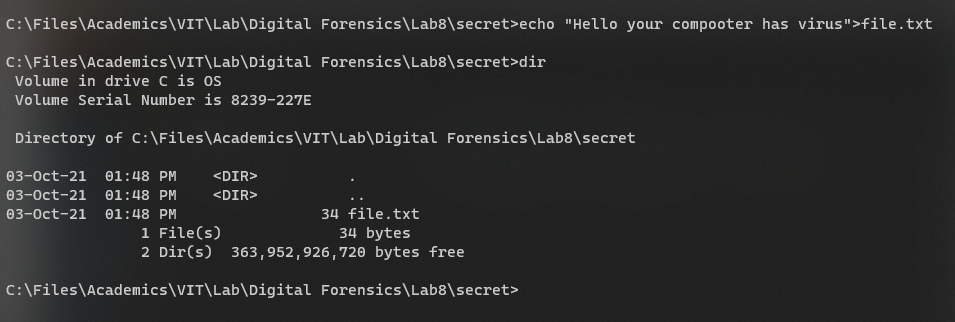
**A**

NTFS file streams or Alternate Data Streams (ADS) can provide attackers with a method of hiding hacker tools on a system and allowthem to execute withour being detected.

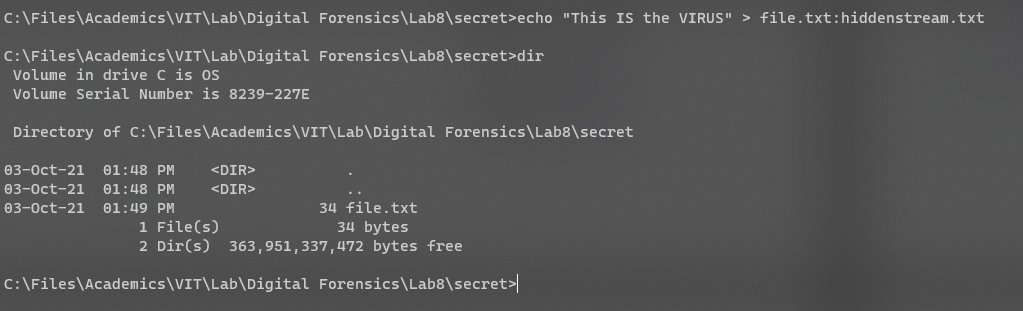
A new folder called ‘secret’ is created and is kept empty as shown.



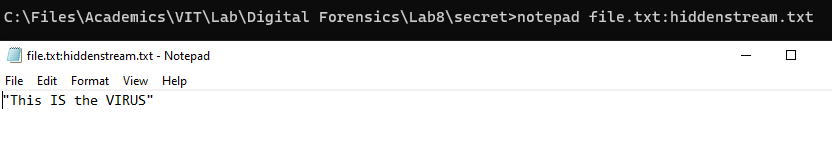
The, we create a text file with some content.



Then, we create a stream with another file.

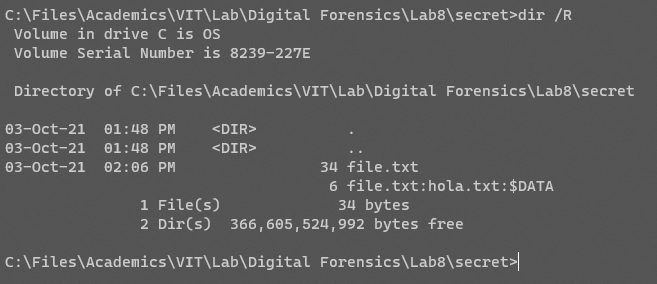


But we see that the second file is not listed by the ‘dir’ command. We can open it with the notepad however.

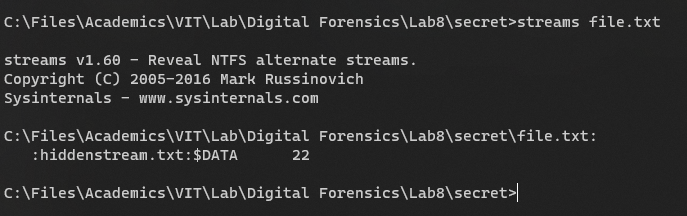


What is happening here is that we have stored data behind a filename (file.txt) with the help of a stream name (hiddenstream.txt). The name after the colon (:) is the hidden stream name. We can discover it only through the terminal.

We can detect hidden streams using the command *‘dir /R’*.

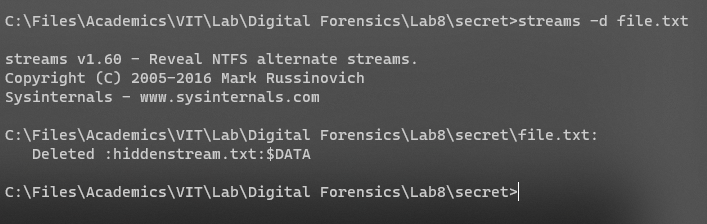


But there are command line applications too like ‘streams’ which we can use to detect files that have streams and their names.



In the screenshot above, we see that ‘*streams <filename>*’ shows the names of streams associated with the filename given (here, file.txt). The hidden stream *‘hiddenstream.txt’* is listed.

We can also delete all streams using the *‘d’­* parameter as follows. We can’t, however, delete a single specific stream.



**CONCLUSION**

We have worked with WinHex, a hex editor, to view contents of files byte by byte and analyse them. We also explored how hidden streams can be made and store files in them.