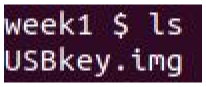
Claire Fei (z5161105)

Report

Aim
The objective of this task is to perform a computer forensic process in order to create a forensic copy of the evidence provided in USB form as per the instructions of the lawyers.

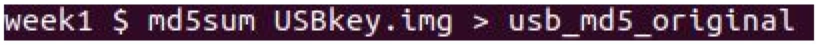
STEP 1 - MAKING A HASH OF THE ORIGINAL
The first step in the forensic process is to generate an MD5 hash of the provided USB image. (A USB is a storage device and an image is a bootable/runnable version of the product which in this case is a USB). The MD5 hash functions as a digital fingerprint of a file, allowing us to verify the integrity of the forensic copy that will be made at a later point in time. It achieves this by converting the contents of a file into a shorter value by using the contents. Hence if the contents are changed, the hash will be changed, thus allowing the relevant parties to check whether the original and the final copy we produce is the same.

Initially it is obvious that the only file in our current directory (the file currently open for viewing) is the provided evidence, the USB image. This is achieved by executing the "ls" command which lists computer files in the current directory.



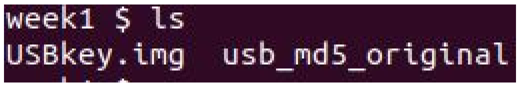
```
week1 $ ls
USBkey.img
```

An MD5 hash of the original file can now be generated and saved by executing the command as shown below. This is done using the program "md5sum" which is used with the argument as the original provided image and placed into a new file called "usb_md5_original". This action is done by using the pipe symbol ">" followed by the target file name which tells the program to place the results of the md5sum program (the hash) into this file.



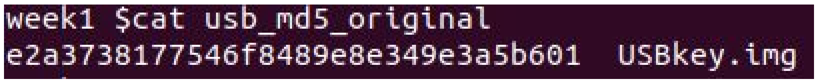
```
week1 $ md5sum USBkey.img > usb_md5_original
```

Doing the "ls" command to view the files allows us to verify that the new file has indeed been created.



```
week1 $ ls
USBkey.img  usb_md5_original
```

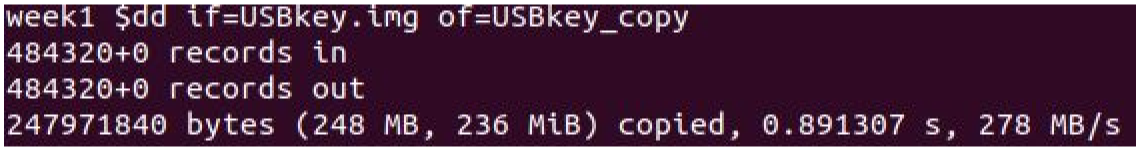
Using another computer tool "cat" which allows us to view the contents of a file, the hash of the USBkey.img (the provided USB image) can be verified to indeed have been generated into the "usb_md5_original" file.



```
week1 $cat usb_md5_original
e2a3738177546f8489e8e349e3a5b601  USBkey.img
```

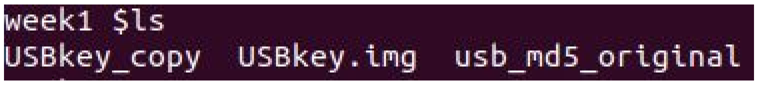
STEP 2 - MAKING A FORENSIC COPY
The next step in the process is to create the copy of the original image by utilising the widely accepted tool "dd". The goal of creating a copy of the original file allows demonstration that the

copy maintains its integrity and has not been tampered with, whether it be intentionally or accidentally during the investigation process. The “dd” tool is used by declaring the argument “if” with the original image and “of” as the name of the copy that would like to be created. The program will then return a success message upon copying the image successfully.



```
week1 $dd if=USBkey.img of=USBkey_copy
484320+0 records in
484320+0 records out
247971840 bytes (248 MB, 236 MiB) copied, 0.891307 s, 278 MB/s
```

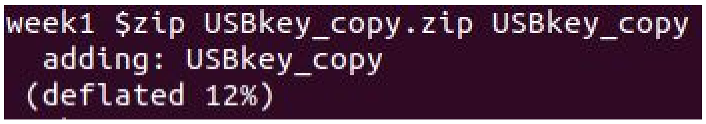
The creation of this new file can be verified by doing another “ls”.



```
week1 $ls
USBkey_copy  USBkey.img  usb_md5_original
```

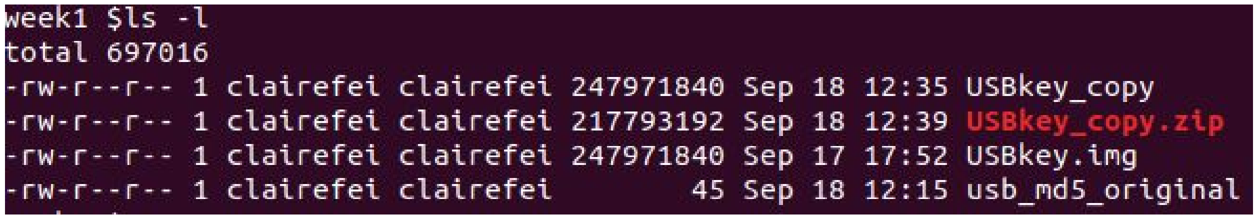
STEP 3 - COMPRESSING THE FORENSIC COPY

Now the file can be compressed (reduce the size of the image copy whilst maintaining original data - this is further demonstrated in STEP 6). This allows the transfer of the files to relevant parties to be faster and more efficient size since the file will be a smaller size. This is done by using the “zip” tool with the first argument as the target file name of the compressed file and the second argument, the original file to be compressed.



```
week1 $zip USBkey_copy.zip USBkey_copy
  adding: USBkey_copy
 (deflated 12%)
```

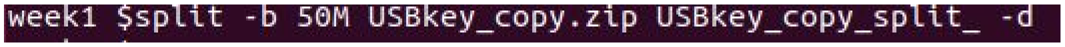
An “ls” can demonstrate that the new compressed file has been created. Specifying “-l” will allow the display additional information (including access rights to the file, size, etc). The names of the files are displayed on the very right hand side column, and the size of the files (in bytes) can be seen in the fourth column. As clearly seen, the USBkey_copy has a larger size than the newly compressed image USBkey_copy.zip hence indicating that the file compression has succeeded.



```
week1 $ls -l
total 697016
-rw-r--r-- 1 clairefei clairefei 247971840 Sep 18 12:35 USBkey_copy
-rw-r--r-- 1 clairefei clairefei 217793192 Sep 18 12:39 USBkey_copy.zip
-rw-r--r-- 1 clairefei clairefei 247971840 Sep 17 17:52 USBkey.img
-rw-r--r-- 1 clairefei clairefei        45 Sep 18 12:15 usb_md5_original
```

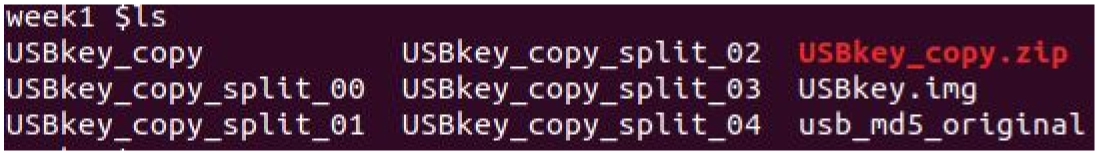
STEP 3 - SPLITTING THE COMPRESSED COPY

The next step in the process is to split the compressed copy. This will allow the distribution of the larger medium over to smaller medium (for example we can put 1 large usb onto multiple smaller CDs) before further distribution out to the relevant parties (or back to the lawyer who gave us the task) is achieved. This is done by using the program “split” with arguments “-b 50M” which split the copy into 50Mb files, the file to split and the target file base name with “-d” to append numerical values to the base filename.



```
week1 $split -b 50M USBkey_copy.zip USBkey_copy_split_ -d
```

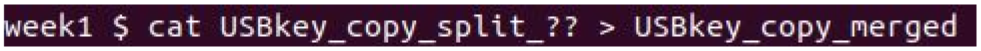
Doing an “ls” we can see that 5 smaller copies have been created “USBkey_copy_split_00”, “USBkey_copy_split_01”, “USBkey_copy_split_02”, “USBkey_copy_split_03”, “USBkey_copy_split_04”. They are now ready to be burnt to CDs or any other medium ready for distribution.



```
week1 $ls
USBkey_copy           USBkey_copy_split_02  USBkey_copy.zip
USBkey_copy_split_00  USBkey_copy_split_03  USBkey.img
USBkey_copy_split_01  USBkey_copy_split_04  usb_md5_original
```

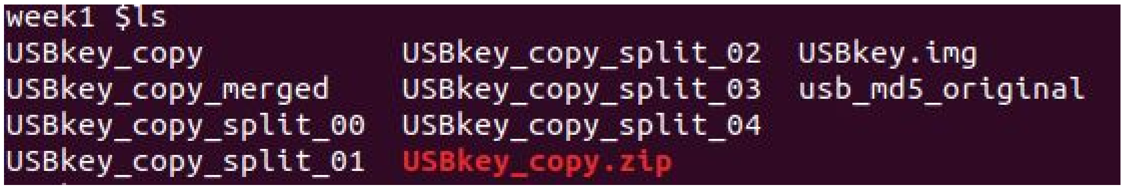
STEP 4 - PUTTING IT ALL TOGETHER

Now that all the smaller copies have ben obtained, the original copy can be rebuilt by using the “cat” program to join all the smaller files into one bigger file. The first argument refers to all the smaller split files and we use the pipe (explained earlier) to direct the resultant bigger file into a file called “USBkey_copy_merged”.



```
week1 $ cat USBkey_copy_split_?? > USBkey_copy_merged
```

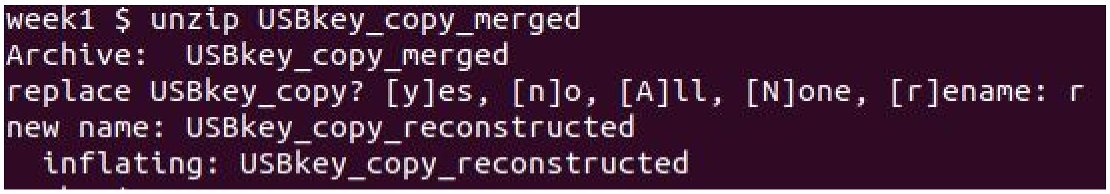
The “ls” shows that the “USBkey_copy_merged” file has been created.



```
week1 $ls
USBkey_copy           USBkey_copy_split_02  USBkey.img
USBkey_copy_merged    USBkey_copy_split_03  usb_md5_original
USBkey_copy_split_00  USBkey_copy_split_04
USBkey_copy_split_01  USBkey_copy.zip
```

STEP 5 - DECOMPRESSING

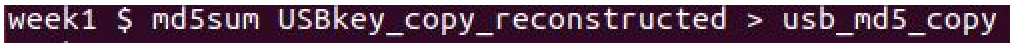
In the fifth step the goal is to decompress the file. Earlier we compressed it using the zip command to compress the file. Now we would like to uncompress it by using the unzip command on the target merged file as achieved in the image below. The goal of decompressing a file is so that the relevant parties can view the original whole, as well as verify that this copy is an exact replica of the original, as explained in the next step.



```
week1 $ unzip USBkey_copy_merged
Archive:  USBkey_copy_merged
replace USBkey_copy? [y]es, [n]o, [A]ll, [N]one, [r]ename: r
new name: USBkey_copy_reconstructed
  inflating: USBkey_copy_reconstructed
```

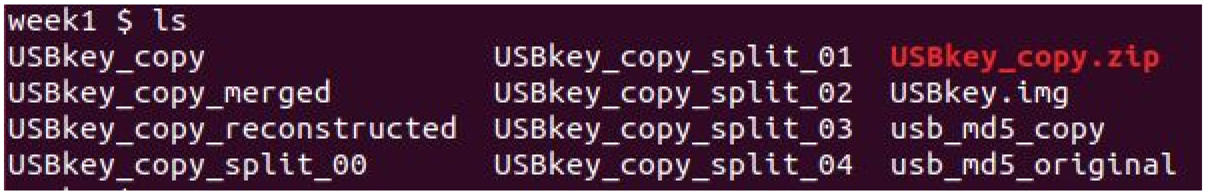
STEP 6 - VALIDATING THE COPY

The last step is to verify that the final rebuilt copy is an exact replica of the original given image. This is done by repeating the first step but on the new copy file we have reconstructed and computing the md5 hash of the reconstructed copy file.



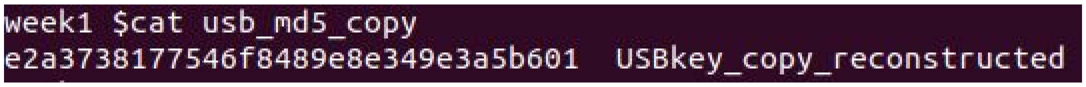
```
week1 $ md5sum USBkey_copy_reconstructed > usb_md5_copy
```

Doing an “ls” confirms that this new file has been created.

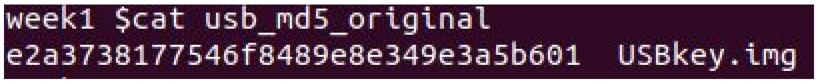


```
week1 $ ls
USBkey_copy                   USBkey_copy_split_01  USBkey_copy.zip
USBkey_copy_merged            USBkey_copy_split_02  USBkey.img
USBkey_copy_reconstructed     USBkey_copy_split_03  usb_md5_copy
USBkey_copy_split_00          USBkey_copy_split_04  usb_md5_original
```

Using the program “cat” to read from the “usb_md5_copy” file it is clear that the md5 hash of the copy and the original file are the same by comparing the two outputs.



```
week1 $cat usb_md5_copy
e2a3738177546f8489e8e349e3a5b601  USBkey_copy_reconstructed
```



```
week1 $cat usb_md5_original
e2a3738177546f8489e8e349e3a5b601  USBkey.img
```

Reflection

After asking for a non-technical person to read the report outlined above, I received numerous comments and feedback on how I may improve the report to further reflect the validity of the evidence and understanding of the process. This will be discussed below.

One of the main pieces of feedback I received was that there was a lot of information to process which made it difficult to remember and reflect on what happened after reading it or at a later date. One of the main suggestions I received was that I should include a brief summary at the end of the report outlining what had occurred during the process. For example, using the instructions given from the activity instructions:

1. Make the copy and a hash of the copy i.e make a forensic copy
2. Compress the copy
3. Split the compressed copy into 50Mb pieces
4. Put it back together
5. Demonstrate the re-constructed image is a reliable copy of the forensic copy

Another way I could address the issue that was discussed above is by including a simple flow diagram. This could include the main steps as outlined above with arrows interconnecting these as appropriate. Then, I could label the arrows with the specific tool used to achieve this with a short, few words explaining what happened. This visual effect may aid the reader in their understanding as opposed to just providing a block of text which may be hard for a non-technical person to understand at a glance. This diagram could be provided at the end of the report with the summary suggestion that was mentioned in the paragraph above.

Furthermore, to help in understanding the mass amount of information in the report, another suggestion I received was that I should highlight or bold key terms and tools used to make them stand out to the reader as often these would get blended into the text as the reader has difficulty processing all the information about the process, tool, goal, validity and etc. By highlighting key words this may also help the reader in understanding the important points and distinguishing these terms explicitly from their explanation.

Another thing that I could have included to facilitate the non-technical reader's understanding is that I could have annotated and captioned the images. As sometimes the images can be confusing as to what the reader should look at, especially at sections with a lot of content such as the later "ls" screenshots when there are a lot of files, it would be helpful to the reader to perhaps circle the target file we should be looking at as to help the reader instantly understand what they should be looking at instead of letting them go to the trouble of trying to figure out what they should be looking at which could then lead to them forgetting critical information that they have read beforehand.

Lastly, I found that the non-technical reader had trouble remembering what a lot of the terminal commands/ tools achieved after reading the report. Since a lot of the time, I only explain what the commands/ tools do the first time they are encountered, but omit them later when they are re-encountered, this creates trouble for the reader as they have to scroll up through a wall of text to figure out where the explanation was. To combat this issue, I suggest that I should have created a separate page containing a glossary of all the important terms, commands and tools (e.g md5, ls, cat, dd, etc.). This will allow the reader to freely reference the glossary any time they do not understand a term and will save me from repeating myself with explanations over and over which may break the coherence. This idea, coupled with the suggesting of bolding the important terms (or marking them numerically) can be a powerful tool in helping and guiding the reader through the report more flawlessly.

On the other hand, the positive feedback I received from the non-technical volunteer, was that they appreciated the step by step format as it allowed clear display of what was to be achieved before the explanation. They also liked the usage of images to separate each step and the wall of text, however some improvements could be made to this as explained above about the annotations/captioning.