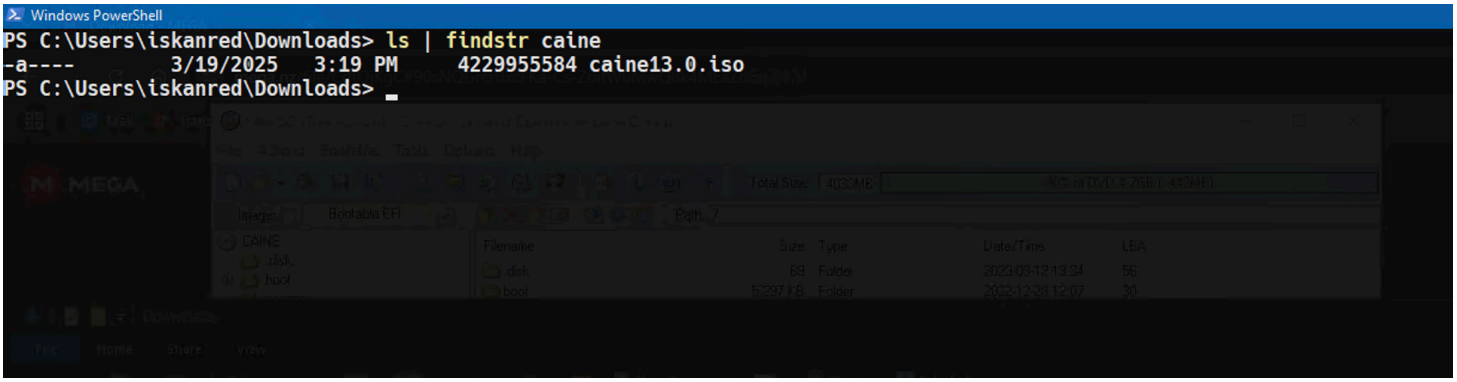
- **Name**: Iskander Nafikov
- **E-mail**: i.nafikov@innopolis.university
- **GitHub**: https://github.com/iskanred

> **Task description**
>
> This lab will introduce you to forensic imaging and data handling using a live environment.
> You also will work in groups (2 persons per group) in Task 3

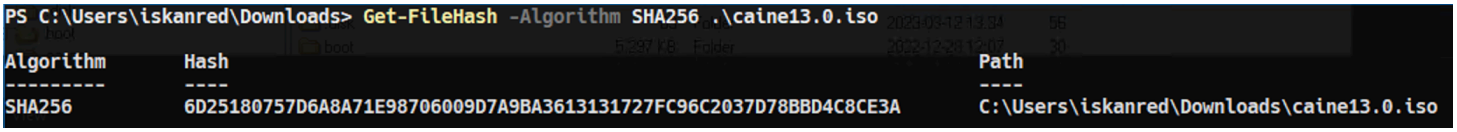
Task 1 - Setting up your environment

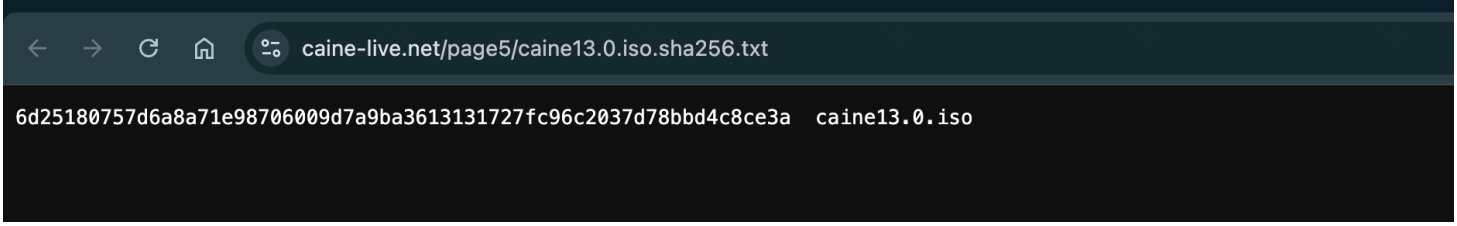
> **Task description**
>
> You need to prepare 2 USB drives. The first one should have a CAINE live environment that will be used to collect evidence. On the second one (this drive will be called drive A) you should deploy that disk image. The disk image is compressed with special utility that preserves original bits intact. You should uncompress it (using FTK imager) and burn it on the flash drive (do not forget about unallocated space).

- So, I prepared two USB drives
- Firstly, I downloaded CAINE ISO images from the official website



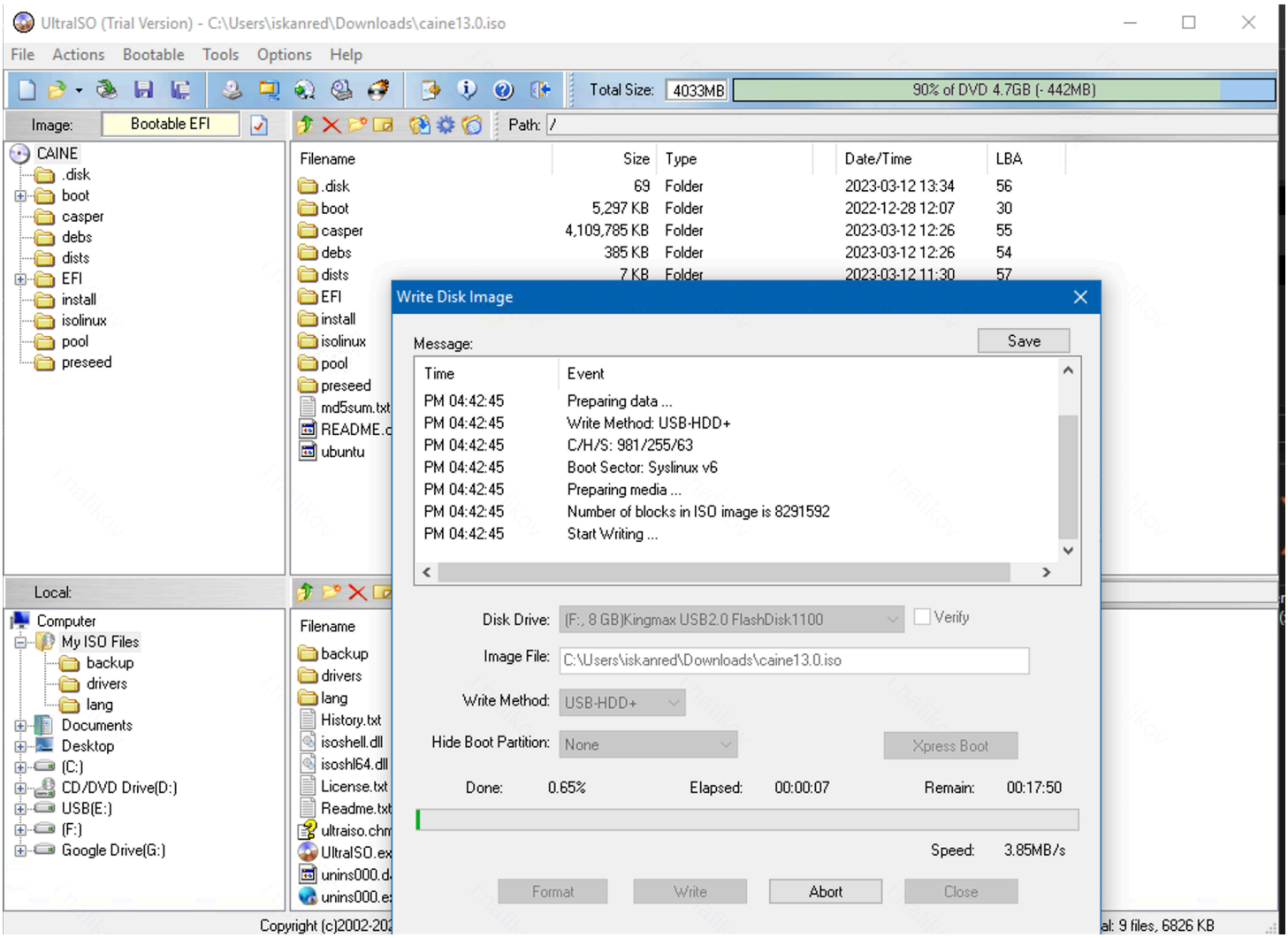
- The instruction on the website stated that it's better to compare checksums to prove image's integrity



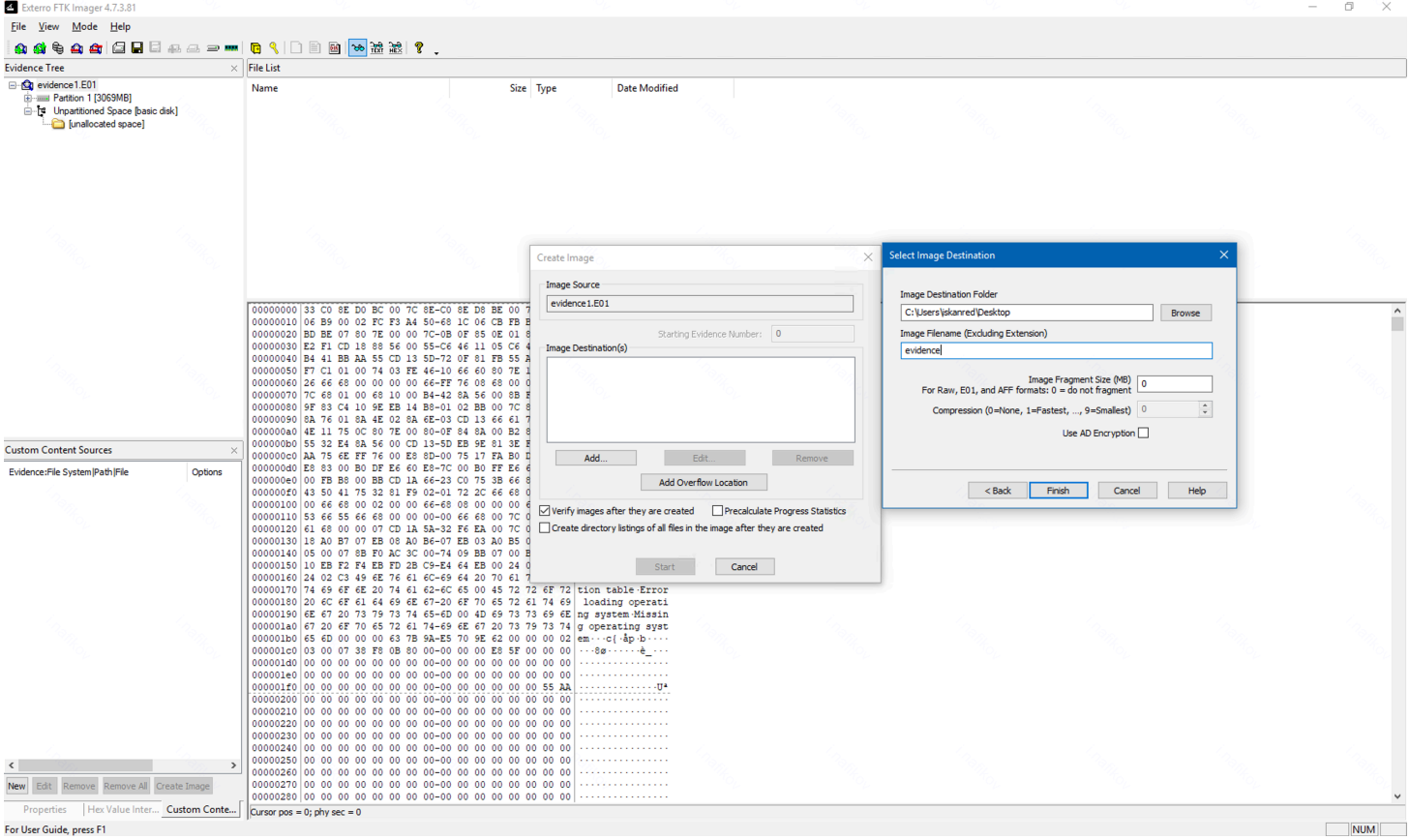


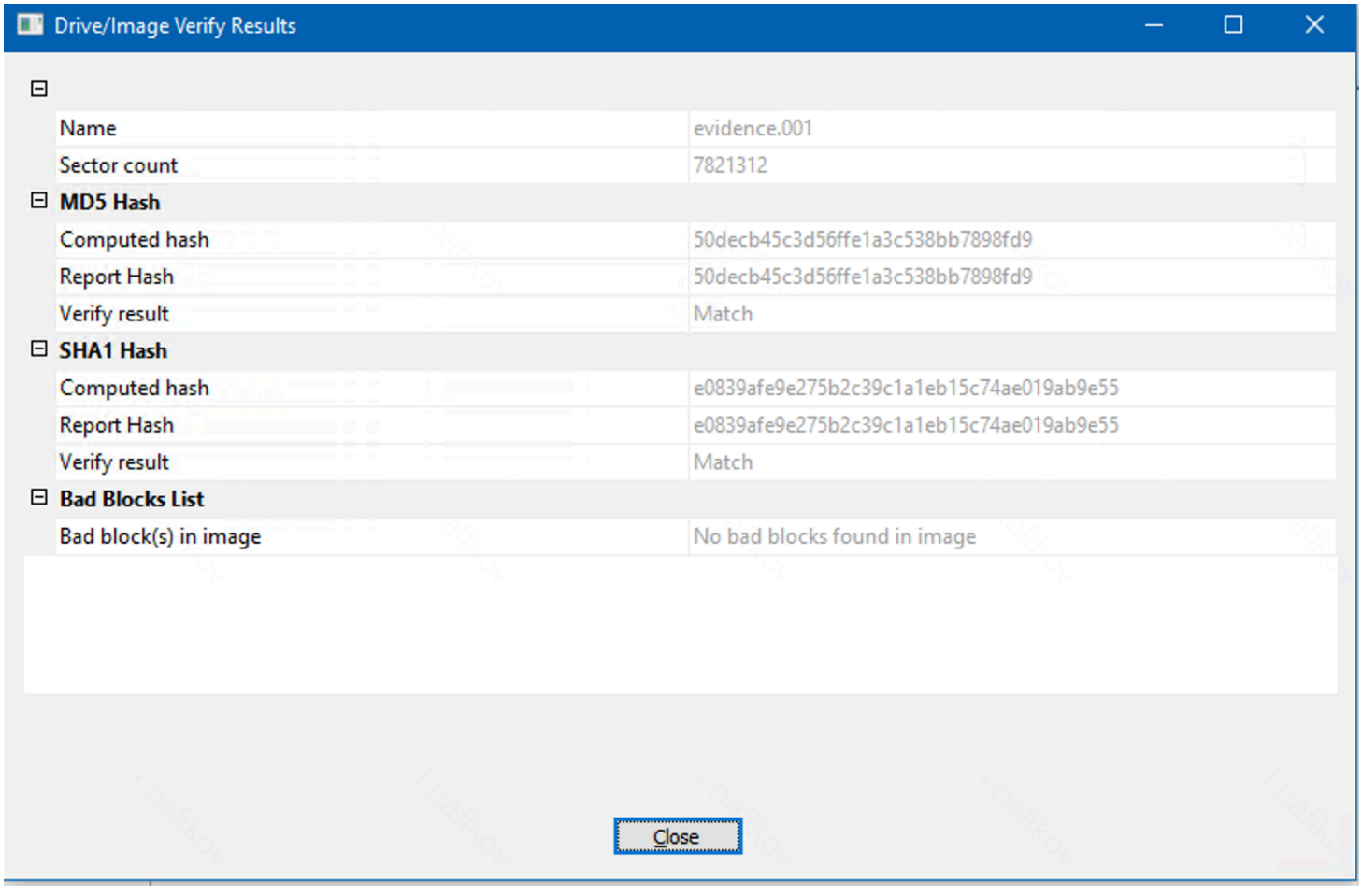
✅

- Then I wrote CAINE live CD/DVD image to the first one drive using [UltraISO]



- Secondly, I uncompressed the image of the evidence to my host machine using `dd` method to keep unallocated space too.


Exterro FTK Imager 4.7.3.81
File View Mode Help
Evidence Tree
evidence1.E01
Partition 1 [3069MB]
Unpartitioned Space [basic disk]
[unallocated space]
File List
Name Size Type Date Modified
Custom Content Sources
Evidence:File System|Path|File Options
New Edit Remove Remove All Create Image
Properties Hex Value Inter... Custom Conte...
Create Image
Image Source
evidence1.E01
Starting Evidence Number: 0
Image Destination(s)
Add... Edit... Remove
Add Overflow Location
Verify images after they are created
Precalculate Progress Statistics
Create directory listings of all files in the image after they are created
Start Cancel
Select Image Destination
Image Destination Folder
C:\Users\skanred\Desktop
Browse
Image Filename (Excluding Extension)
evidence
Image Fragment Size (MB)
For Raw, E01, and AFF formats: 0 = do not fragment
0
Compression (0=None, 1=Fastest, ..., 9=Smallest)
0
Use AD Encryption
< Back Finish Cancel Help
Cursor pos = 0; phy sec = 0
For User Guide, press F1
NUM

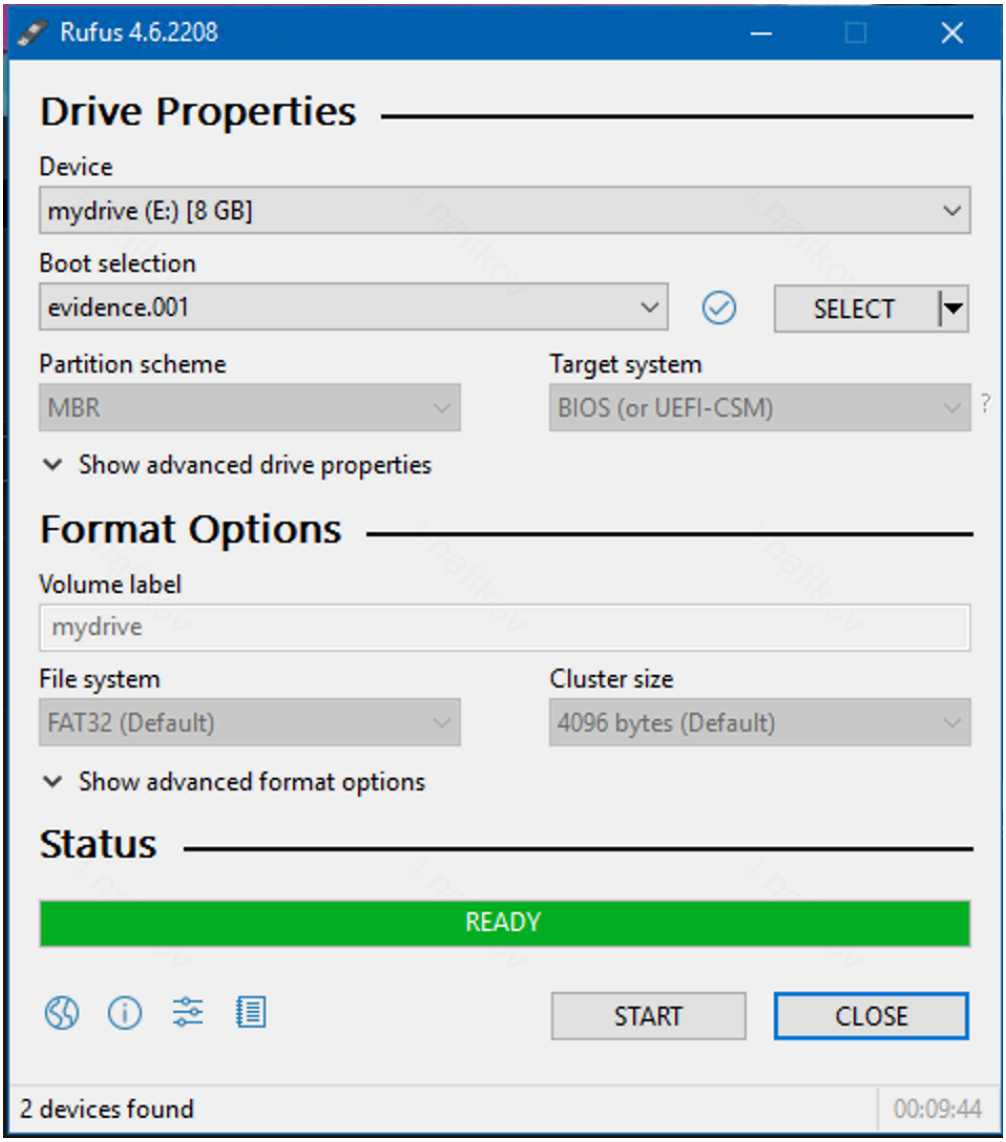

Drive/Image Verify Results
Name
evidence.001
Sector count
7821312
MD5 Hash
Computed hash
50decb45c3d56ffe1a3c538bb7898fd9
Report Hash
50decb45c3d56ffe1a3c538bb7898fd9
Verify result
Match
SHA1 Hash
Computed hash
e0839afe9e275b2c39c1a1eb15c74ae019ab9e55
Report Hash
e0839afe9e275b2c39c1a1eb15c74ae019ab9e55
Verify result
Match
Bad Blocks List
Bad block(s) in image
No bad blocks found in image
Close

evidence.001.txt

```
Created By Exterro® FTK® Imager 4.7.3.81

Case Information:
Acquired using: ADI4.7.3.81
Case Number: 1
Evidence Number: 1
Unique description: cf-lab-01
Examiner: Iskander Nafikov
Notes: cf-lab-01

------------------------------------------------------------

Information for C:\Users\iskanred\Desktop\evidence:

Physical Evidentiary Item (Source) Information:
[Device Info]
 Source Type: Physical
[Verification Hashes]
 MD5 verification hash: 50decb45c3d56ffe1a3c538bb7898fd9
[Drive Geometry]
 Bytes per Sector: 512
 Sector Count: 7,821,312
[Image]
 Image Type: E01
 Case number: Lab-1
 Evidence number: 0001
 Examiner: Emil A. Sharifullin
 Notes: 4C532000060816106053
 Acquired on OS: Linux deft8 3.5.0-30-generic #51-Ubuntu SMP Tue May 14 18:47:48 UTC 2013 x86_64
 Acquired using: guymager 0.7.3-1
 Acquire date: 1/9/2017 11:50:17 AM
 System date: 1/9/2017 11:50:17 AM
 Unique description: Red evidence flash drive
 Source data size: 3819 MB
 Sector count:    7821312
[Computed Hashes]
 MD5 checksum:    50decb45c3d56ffe1a3c538bb7898fd9
 SHA1 checksum:   e0839afe9e275b2c39c1a1eb15c74ae019ab9e55

Image Information:
 Acquisition started:   Sun Mar 23 17:08:16 2025
 Acquisition finished:  Sun Mar 23 17:08:32 2025
 Segment list:
  C:\Users\iskanred\Desktop\evidence.001
 COMPUTED HASH :  50decb45c3d56ffe1a3c538bb7898fd9
 COMPUTED HASH :  e0839afe9e275b2c39c1a1eb15c74ae019ab9e55

Image Verification Results:
 Verification started:  Sun Mar 23 17:08:32 2025
 Verification finished: Sun Mar 23 17:08:44 2025
 MD5 checksum:    50decb45c3d56ffe1a3c538bb7898fd9 : verified
 SHA1 checksum:   e0839afe9e275b2c39c1a1eb15c74ae019ab9e55 : verified

```

Normal text file | length : 1,699 lines : 53 | Ln : 14 Col : 1 Pos : 331 | Windows (CR LF) | UTF-8-BOM | INS

- Finally, I wrote this disk image to my second USB flash drive using [Rufus]



Task 2 - Imaging

1.

> **Task description**
>
> Discuss how you can retrieve an image from an, currently off-line, USB stick in a forensically sound manner. Create and describe this method.

1. **Boot CAINE**: Start your system using the CAINE Live CD/USB. Ensure you're using a read-only medium or a properly installed CAINE to prevent altering the evidence.
2. **Connect the USB Stick**: Insert the USB stick into the machine running CAINE. Make sure to note which device it is using `lsblk` (e.g., `/dev/sdb`).
3. **Create a Forensic Image**: We can use `dd` or `Guymager` to create a bit-by-bit copy of the USB stick. This process is crucial as it preserves the original data by creating a forensic image. It's better to choose `E01` image format since `E01` images contain a lot of additional integrity data that makes it easy to check if the image file is damaged or modified and where.
4. **Calculate and Verify Checksums**: Finally, we should check image's integrity comparing all the necessary `MD5` and `SHA1` checksums which are computed by `Guymanager` automatically or we can do this manually using `md5sum`, `sha1sum`. This is also a crucial step to be sure that the `E01` image was stored without any issues and it contains exactly the same data as USB drive does.
5. **Start analysis and documentation**: Register the timestamp right after creating the `E01` image and start analysis and reporting the image.

2.

> **Task description**
>
> Write a one-line description, or note a useful feature for the following tools included in CAINE: `Guymager`, `Disk Image Mounter`, `dcfldd` / `dc3dd`, `kpartx`.

- **Guymager**: A graphical tool for creating and verifying forensic disk images, supporting multiple image formats and hash verification for ensuring data integrity.
- **Disk Image Mounter**: This tool allows for easy mounting of disk images as virtual filesystems, enabling forensic analysis of the contents without altering the original image.
- **dcfldd / dc3dd**: Enhanced versions of `dd` that include features for hashing on-the-fly, progress monitoring, and error recovery, making them ideal for forensic imaging.
- **kpartx**: A utility that creates device mappings for partitions within disk images, allowing users to easily access and interact with individual partitions for forensic analysis.
- **Autopsy**: A web-based forensic analysis tool.
- **The Sleuth Kit**: Command-line analysis tools for filesystems.

- **TestDisk**: For data recovery from lost partitions.
- **dmesg**: For viewing kernel messages, which are useful in troubleshooting.

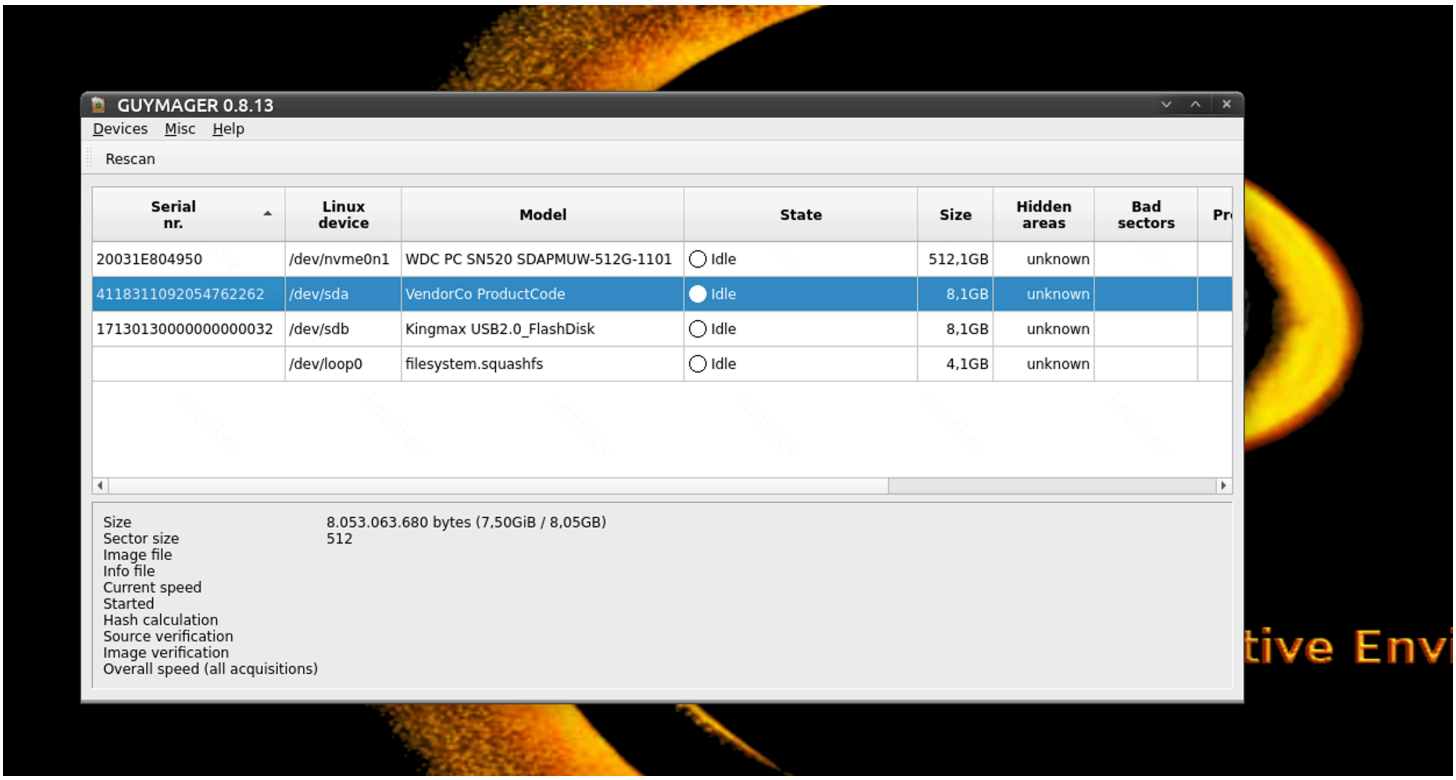
3.

Task description

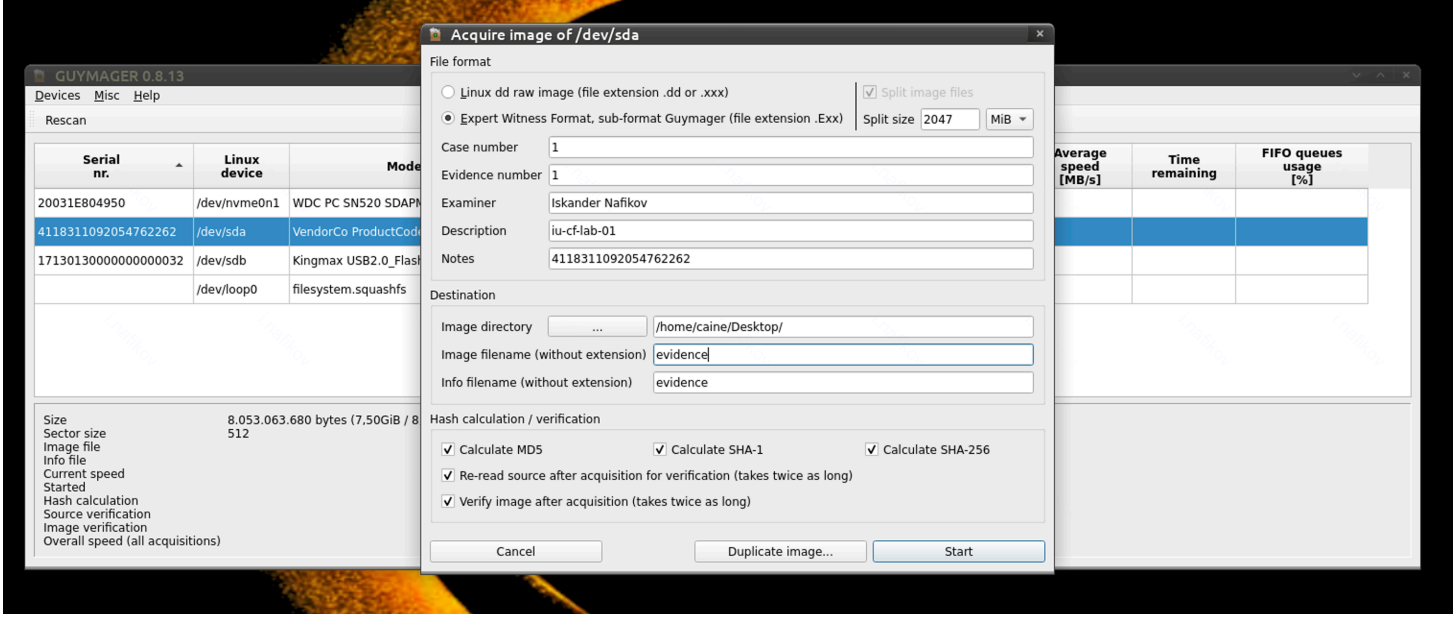
Follow your method to retrieve the image from drive A. Please use timestamps, explain every tool and note down the version. For the purpose of speed. Make sure both team members have access to the retrieved image. You can use your PCs as an evidence sharing platform

- I ran `lsblk` and detected that `/dev/sda` is my USB flash drive since it has `sda1` partition with 3 Gb of memory which is exactly the size of the burnt image

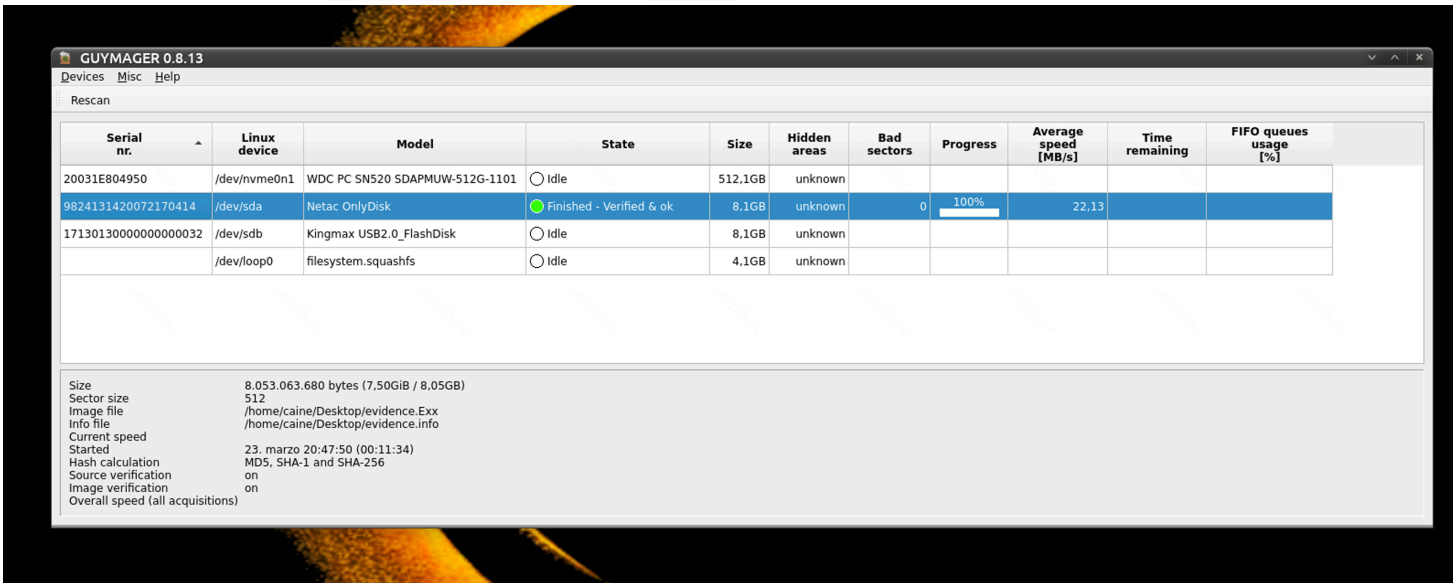
- So then I ran Guymager tool



- I acquired the image, input necessary fields and selected toggles: MD5, SHA-1, SHA-256 and re-read source for double verification.



- I started acquisition at `20:47:50` and it took `22:13` minutes.



- We already see that the image was verified successfully but I still checked the `evidence.info` to check hash values manually

```
No bad sectors encountered during acquisition.
No bad sectors encountered during verification.
State: Finished successfully

MD5 hash                   : dbfd4b481ed5f36e80d7257f0bec9af2
MD5 hash verified source   : dbfd4b481ed5f36e80d7257f0bec9af2
MD5 hash verified image    : dbfd4b481ed5f36e80d7257f0bec9af2
SHA1 hash                  : 30803eb20470bfd2e8186d3046baa8cd6d3ffee2
SHA1 hash verified source  : 30803eb20470bfd2e8186d3046baa8cd6d3ffee2
SHA1 hash verified image   : 30803eb20470bfd2e8186d3046baa8cd6d3ffee2
SHA256 hash                : c29ac03609f66a13d3aa702f775fbaa6c3c51057b538a9df99ef568873df1d06
SHA256 hash verified source: c29ac03609f66a13d3aa702f775fbaa6c3c51057b538a9df99ef568873df1d06
SHA256 hash verified image : c29ac03609f66a13d3aa702f775fbaa6c3c51057b538a9df99ef568873df1d06
Source verification OK. The device delivered the same data during acquisition and verification.
Image verification OK. The image contains exactly the data that was written.
```

- So now we see that everything was OK ✅

4.

Task description

Read about CAINE Linux and its features while waiting on the dump to finish.

- **CAINE** (Computer Aided INvestigative Environment) is a Linux-based distribution designed for digital forensics, providing a comprehensive suite of tools for data recovery and analysis.
- It operates as a **live system**, allowing users to boot from USB or DVD without altering the host machine, which is crucial for preserving evidence integrity.
- CAINE **defaults to read-only mounting** of storage media to ensure that original data remains unaltered during investigations, and it **supports a variety of filesystems**, enabling forensic examination across different storage devices.
- The distribution includes **user-friendly forensic tools** such as Guymager, Autopsy, and The Sleuth Kit, facilitating both novice and expert users in their analysis.
- With **extensive documentation and community support**, CAINE is an essential tool for forensic investigators and law enforcement professionals executing nonintrusive evidence collection and thorough analyses.

4.1.

Task description

Why would you use a Forensic distribution and what are the main differences between a regular distribution?

1. **Toolset**: Forensic distributions come preloaded with specialised forensic tools for data recovery and analysis, while regular distributions include general-purpose software that often requires separate installations for similar tools.
2. **Default Behavior**: Forensic distributions typically mount drives in read-only mode to prevent data alteration, whereas regular distributions generally use read-write access by default.
3. **User Interface**: Forensic distributions feature interfaces optimised for investigative workflows, including guided processes and reporting, while regular distributions focus on general usability for everyday tasks.
4. **Security Features**: Forensic distributions prioritise maintaining evidence integrity and often include built-in security measures, while regular distributions lack these specialised protections.
5. **Compliance Standards**: Forensic distributions are designed to comply with legal standards and best practices in forensic investigations, unlike regular distributions, which may not consider these requirements.

4.2.

Task description

When would you use a live environment and when would you use an installed environment?

- You would use a **live environment** for CAINE when conducting forensic investigations on a potentially compromised machine, as it allows you to operate without altering the system or its data, preserving the integrity of the evidence. This setup is ideal for initial data collection and analysis in the field or when working with sensitive evidence.
- An **installed environment** would be more suitable for prolonged analysis, extensive data processing, or when utilizing advanced tool configurations, as it provides enhanced hardware support and allows for faster access to resources. Additionally, an installed environment is beneficial for users who require a stable operating platform for ongoing forensic training or complex investigations that demand multiple tools and applications.

4.3.

Task description

What are the policies of CAINE?

1. **Preservation of Evidence**: CAINE emphasises the importance of maintaining the integrity of digital evidence. Tools and processes are designed to prevent any alteration of original data during forensic analysis.
2. **Non-Intrusiveness**: The live environment allows users to conduct investigations without interacting with the host operating system, which helps protect the evidence from being compromised.
3. **Open Source Philosophy**: CAINE adheres to the principles of open source software, promoting transparency, collaboration, and community-driven development. Users are encouraged to contribute to the improvement of tools and solutions.
4. **Compliance with Legal Standards**: The tools and practices supported by CAINE are intended to align with legal standards and best practices in digital forensics, ensuring that evidence can be utilised in legal contexts without issues regarding admissibility.
5. **User Education and Support**: CAINE prioritises education through documentation, tutorials, and community support, enabling users, particularly those new to forensic investigations, to understand and effectively utilize the tools available.
6. **Community Contribution**: The CAINE community encourages feedback, contributions, and collaboration among users to continuously enhance the functionality and usability of the distribution.

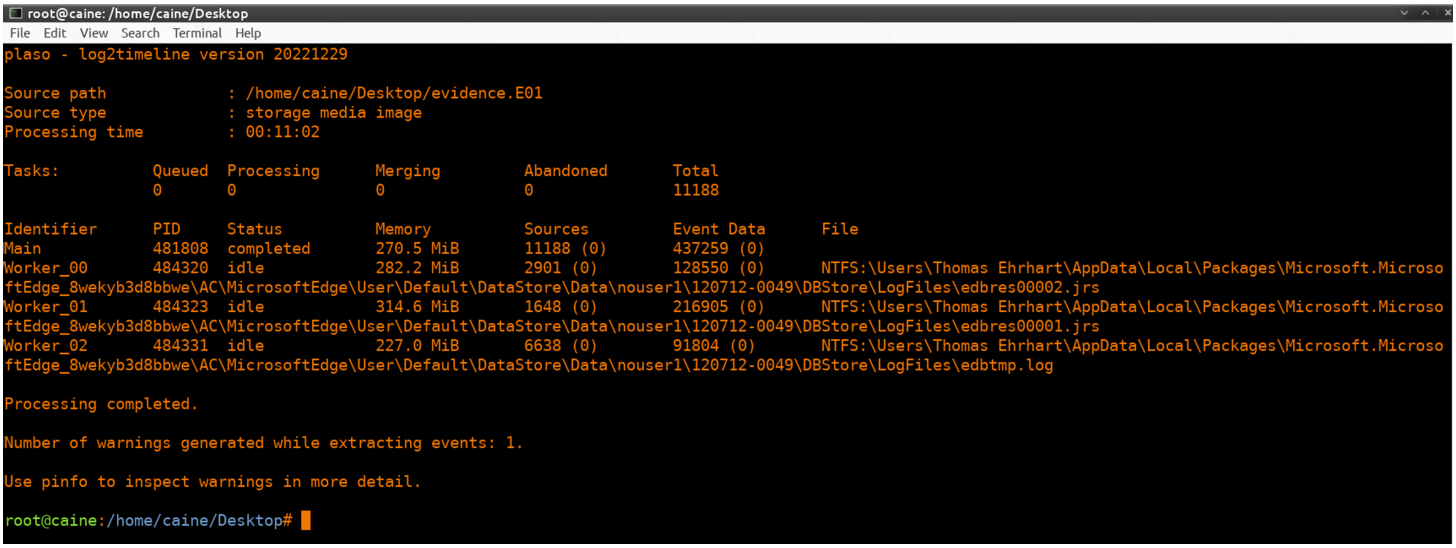
5.

Task description

As soon as your dump finishes, start a tool to create a timeline on the image. You will need this timeline later in the assignment. Hints: `log2timeline.py`.

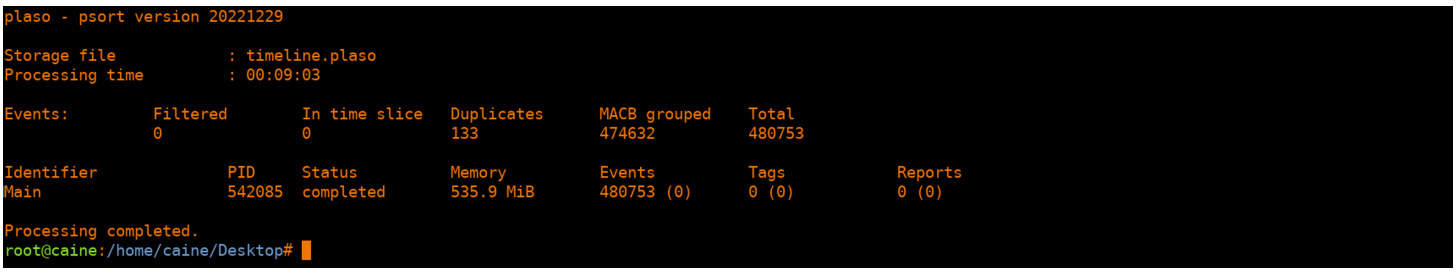
- I used the pre-installed `log2timeline.py` tool from the Plaso forensics analysis framework. This tool extracts timestamped events from the image and generates a timeline in the Plaso format.

```
log2timeline.py --partitions all --storage_file timeline.plaso evidence.E01
```



- I generated the Plaso timeline and exported it to CSV format it using `psort.py`

```
psort.py -w timeline.csv timeline.plaso
```



- Finally, I was able to see these logs. There are almost 490,000 logs are here, so it must be difficult to analyze

E100070 fx [HKEY_LOCAL_MACHINE\Software\Classes\CLSID\{361778EC-D741-4CE1-90F9-743435F56E13}] (default): [REG_SZ] Remove Device Menu

	A	B	C	D	E
476	1979-12-31T23:00:00.000000+00:00	Last Access	LNK	Windows Shortcut	File size: 0 File attribute flags: 0x00000010 Drive type: 3 Drive serial number: 0xb4319435 V
477	1984-01-11T08:00:00.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://1.gravatar.com/avatar/12d879c17d9678f85facefc4e2ff1064?s=38&d=wavatar
478	2000-03-16T03:47:22.000000+00:00	Creation Ti	META	Open XML Metadata	Application: Microsoft Excel 16.0300 Last saved by: X Author: TMJCC Security flags: 0x0000(
479	2001-01-01T08:00:00.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://www.facebook.com/rsrc.php/v2/y1/r/LVx-xkvaJ0b.png Access count: 1 Synchr
480	2001-01-01T08:00:00.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://www.facebook.com/rsrc.php/v3iSgH4/yu/l/en_US/gtpDOdy6aml.js Access cou
481	2001-11-26T21:23:10.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://static01.nyt.com/ads/blank.gif Access count: 1 Synchronization count: 0 Filena
482	2002-07-22T01:56:56.000000+00:00	Last Printe	META	Open XML Metadata	Application: Microsoft Excel 16.0300 Last saved by: X Author: TMJCC Security flags: 0x0000(
483	2007-09-19T08:50:25.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://static.criteo.net/js/ld/ld.js Access count: 1 Synchronization count: 0 Filename:
484	2008-12-05T21:34:37.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://images2.fanpop.com/images/photos/2900000/LOL-CATS-lol-cats-2985219-772
485	2009-02-19T12:00:39.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://images2.fanpop.com/images/photos/4200000/audi-cars-audi-4294883-1280-1
486	2009-03-05T08:52:01.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://images.fanpop.com/images/image_uploads/Lightning-McQueen-disney-pixar-c
487	2009-07-13T23:27:26.000000+00:00	Content M	PE	PE/COFF file	PE Type: Dynamic Link Library (DLL) Import hash: 0da87250a394ebd0e4a2e03fba21e9ed Ex
488	2009-07-13T23:27:26.000000+00:00	Content M	PE	PE/COFF file	PE Type: Dynamic Link Library (DLL) Import hash: 0da87250a394ebd0e4a2e03fba21e9ed Ex
489	2009-07-14T01:11:00.000000+00:00	Creation Ti	PE	PE/COFF file	PE Type: Dynamic Link Library (DLL) Import hash: 0da87250a394ebd0e4a2e03fba21e9ed Ex
490	2009-07-14T01:11:00.000000+00:00	Creation Ti	PE	PE/COFF file	PE Type: Dynamic Link Library (DLL) Import hash: 0da87250a394ebd0e4a2e03fba21e9ed Ex
491	2009-11-22T13:26:26.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://t.wallpaperweb.org/wallpaper/animals/1024x768/Cats2005cal09ccnan.jpg Acc
492	2009-12-05T22:50:46.000000+00:00	Creation Ti	PE	PE/COFF file	PE Type: Executable (EXE) Import hash: 099c0646ea7282d232219f8807883be0
493	2010-04-15T17:07:29.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://c.betrad.com/a/4.gif Access count: 1 Synchronization count: 0 Filename: 4[1].g
494	2010-05-19T19:29:43.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://www.hdwallpapers.in/walls/ronn_motor_scorpion_super_car-wide.jpg Access
495	2010-07-14T10:02:22.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://images2.fanpop.com/image/photos/13800000/farrari-sports-cars-13821367-1
496	2010-08-05T17:06:30.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://www.zastavki.com/pictures/1680x1050/2010/Animals_Cats_Cat_023761_.jpg
497	2010-10-04T23:48:28.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://ajax.aspnetcdn.com/ajax/jquery.templates/beta1/jquery.tmpl.min.js Access cc
498	2010-10-07T20:06:06.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://images4.fanpop.com/image/photos/16000000/Beautiful-Cat-and-Kitten-cats-1
499	2010-10-07T20:24:12.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://images4.fanpop.com/image/photos/16000000/Beautiful-Cat-cats-16096437-1
500	2010-10-07T22:37:55.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://dailycollegian.com/media/2010/10/cats.jpg Access count: 1 Synchronization co
501	2010-11-04T01:42:54.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://pbs.twimg.com/profile_images/1980294624/DJT_Headshot_V2_normal.jpg A
502	2010-11-12T00:18:36.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://ajax.aspnetcdn.com/ajax/jquery/jquery-1.4.4.min.js Access count: 2 Synchroni
503	2010-11-18T17:49:13.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.6/jquery-ui.min.js Access count: 2 Sync
504	2010-11-22T18:36:30.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://www.freemovieposters.net/posters/cars_2006_5_poster.jpg Access count: 1 Sy
505	2011-02-14T11:09:26.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://load.s3.amazonaws.com/pixel.gif Access count: 1 Synchronization count: 0 Filer
506	2011-03-01T22:54:04.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://i1.nyt.com/images/loaders/loading-grey-lines-circle-18.gif Access count: 2 Sync
507	2011-03-08T08:49:03.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://stanzedicinema.files.wordpress.com/2011/03/cars-2-mcqueen.jpg Access cour
508	2011-06-24T08:11:28.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://www.hdwallpapers.in/walls/cars_2_race-wide.jpg Access count: 1 Synchronizat
509	2011-07-20T08:03:04.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://www.thecatsbreeds.net/gallery/deaf-cat/deaf_cat.jpg Access count: 1 Synchror
510	2011-08-05T06:45:12.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://1.gravatar.com/avatar/dbba0a79ed3c899888135edd06d93a3a?s=38&d=wavat
511	2011-10-26T13:06:12.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://pics.chaikoffee.com/wp-content/uploads/2011/10/spyker-cars-wallpapers.jpg
512	2012-03-01T17:13:25.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: https://static01.nyt.com/packages/js/multimedia/libs/jquery-1.7.1.min.js Access coun
513	2012-03-05T09:16:56.000000+00:00	Content M	WEBHIST	MSIE WebCache container record	URL: http://www.wintuts.com/modules/comment_notify/comment_notify.css?C Access cc

Task 3 - Verification

Task description

Verification of the retrieved evidence is also required. You are going to exchange your evidence between your group of students. This can be done by sharing your USB device (drive A) with your teammate.

1.

Task description

Create and describe a method that enables the verification of your method. Write this down in steps that your teammate can follow.

Preparation steps for a sender

1. First, I ran the following command to convert `.ewf` (`.e0{N}`) file into a `.raw` file.

```
ewfexport evidence.E01 # in the suggested options name the output file
`verify.raw`
```

2. Now the sender has the `verify.raw` file, so he should burn this file to the flash drive using `dcfldd` also computing the checksum hash values:

```
sudo dcfldd if=verify.raw of=/dev/sd{device}{partition} hash=md5,sha256
md5log=md5.txt sha256log=sha256.txt conv=sync,noerror bs=512 hashconv=after
```

   - **`dcfldd`** : It is a modified version of `dd` that provides additional features for forensic purposes, such as hashing and logging.
   - **`if=evidence.raw`** : The `if` (input file) option specifies the input file to be read, which in this case is `evidence.raw` .
   - **`of=/dev/sdb`** : The `of` (output file) option specifies the output destination. In our case it must be the USB flash drive we want to transfer to our partner.
   - **`hash=md5,sha256`** : This option indicates that `dcfldd` should calculate both MD5 and SHA256 hashes of the input data as it is being processed.
   - **`md5log=md5.txt sha256log=sha256.txt`** : These options specify the filenames (`md5.txt` , `sha256.txt`) where the MD5 and SHA256 hash logs will be written.
   - **`conv=sync,noerror`** : These options tell `dcfldd` to pad the output blocks with zeroes if they are less than the specified block size, ensuring that the output remains consistent and aligned with no error produced, which is important for forensic purposes .
   - **`bs=512`** : This option sets the block size for reading and writing to 512 bytes. This is a common block size for disk operations and can improve performance during the copying process.
   - **`hashconv=after`** : This option indicates that the hashing should be computed after the data transfer is complete; this ensures that the entire output is hashed rather than hashing as the data is being read and written simultaneously.
3. Now the image is burnt to the USB flash drive (`/dev/sd{device}{partition}`) and its MD5 and SHA256 checksums are stored in the `md5.txt` and `sha256.txt` correspondingly. The sender now should share the USB drive with the receiver, along with the hash values (from `md5.txt` and `sha256.txt`).

Verification steps for a receiver

1. Receive suspected USB drive, `md5.txt` and `sha256.txt` files.
2. Insert the USB drive and determine USB name (e.g., `/dev/sdb`) using `lsblk` or `dmesg` :

```
sudo dmesg | tail tee
```

3. Display the the information about the drive. It should be the same as I provided (look at the screenshot below)

```
cat /proc/scsi/usb-storage/{number}
```

```
root@caine:~# cat /proc/scsi/usb-storage/33
   Host scsi33: usb-storage
       Vendor: Kingmax
      Product: USB2.0 FlashDisk
Serial Number: 17130130000000000032
     Protocol: Transparent SCSI
    Transport: Bulk
       Quirks:
root@caine:~# ^C
```

4. Check the size of the block device (USD drive) and other parameters. They should be the same as I provided (look at the screenshot below)

```
sudo fdisk -l /dev/sdb
```

```
root@caine:~# fdisk -l /dev/sdb
Disk /dev/sdb: 7,52 GiB, 8074035200 bytes, 15769600 sectors
Disk model: USB2.0 FlashDisk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x629e70e5

Device     Boot Start     End Sectors Size Id Type
/dev/sdb1        128 6285439 6285312   3G  7 HPFS/NTFS/exFAT
root@caine:~#
```

5. Create directories for further mount and logs

```
mkdir /evidence      # for hash logs
mkdir /mnt/evidence  # to mount copied disk for analysis
```

6. Now copy the content of the flash drive to a new disk image and compute MD5 and SHA256 hash values using `dcfldd`

```
dcfldd if=/dev/sdb1 of=/mnt/evidence/image.raw hash=md5,sha256
md5log=/evidence/md5.txt sha256log=/evidence/sha256.txt sizeprobe=if
conv=sync,noerror hashconv=after
```

7. Compare the calculated hashes in `/evidence/md5.txt` and `/evidence/sha256.txt` with those provided by me (from `md5.txt` and `sha256.txt` correspondingly).
8. If they match everything is okay! If not => evidence was corrupted, do not trust it!

2.

Task description

Exchange USB images with your partner. Verify the procedure that he used and the resulting image. Write a small paragraph of max 200 words. Write as if you were verifying the evidence gathering procedure for a court case.

We did this task together with with Mohamad Bahja (m.bahja@innopolis.university)

My actions as a sender

As a sender I completed all the steps I described above ⬆️

1. I started with decompressing `E01` to `raw`:

```
root@caine:/home/caine/Desktop# ewfexport evidence1.E01
ewfexport 20140812

Information for export required, please provide the necessary input
Export to format (raw, files, ewf, smart, ftk, encase1, encase2, encase3, encase4, encase5, encase6, encase7, encase7-v2, linen5, linen6, linen7, ewfx) [raw]:
Target path and filename without extension or - for stdout: verify
Evidence segment file size in bytes (0 is unlimited) (0 B <= value <= 7.9 EiB) [0 B]:
Start export at offset (0 <= value <= 4004511744) [0]:
Number of bytes to export (0 <= value <= 4004511744) [4004511744]:

Export started at: Mar 24, 2025 21:20:19
This could take a while.

Status: at 15%.
        exported 586 MiB (614498304 bytes) of total 3.7 GiB (4004511744 bytes).
        completion in 22 second(s) with 146 MiB/s (154019682 bytes/second).

Status: at 37%.
        exported 1.4 GiB (1515159552 bytes) of total 3.7 GiB (4004511744 bytes).
        completion in 13 second(s) with 181 MiB/s (190691035 bytes/second).

Status: at 59%.
        exported 2.2 GiB (2386460672 bytes) of total 3.7 GiB (4004511744 bytes).
        completion in 8 second(s) with 190 MiB/s (200225587 bytes/second).

Status: at 82%.
        exported 3.0 GiB (3310682112 bytes) of total 3.7 GiB (4004511744 bytes).
        completion in 3 second(s) with 201 MiB/s (210763776 bytes/second).

Export completed at: Mar 24, 2025 21:20:37

Written: 3.7 GiB (4004511744 bytes) in 18 second(s) with 212 MiB/s (222472874 bytes/second).
MD5 hash calculated over data:          50decb45c3d56ffe1a3c538bb7898fd9
ewfexport: SUCCESS
root@caine:/home/caine/Desktop#
```

2. Then I burned the image to my USB flash drive with computing its SHA256 and MD5 hash values

```
root@caine:/home/caine/Desktop# dcfldd if=verify.raw of=/dev/sdb hash=md5,sha256 md5log=md5.txt sha256log=sha256.txt conv=sync,noerror bs=512 hashconv=after
7821312 blocks (3819Mb) written.
7821312+0 records in
7821312+0 records out
root@caine:/home/caine/Desktop#
```

3. Finally, I provided the flash drive to Mohamad alongside with the `md5.txt` and `sha256.txt` files.

My actions as a receiver

As a receiver I again completed all the steps I described above ⬆️

1. I received a USB stick from Mohamad together with his `md5.txt` and `sha256.txt`

2. I plugged it to my PC, ran `dmesg` to figure the name of this block device. The name was `dev/sdb` and a partition was `/dev/sdb1`

```
root@caine:~# dmesg | tail
[ 6001.700017] sd 33:0:0:0: Attached scsi generic sg2 type 0
[ 6001.705634] sd 33:0:0:0: [sdb] 15769600 512-byte logical blocks: (8.07 GB/7.52 GiB)
[ 6001.710183] sd 33:0:0:0: [sdb] Write Protect is off
[ 6001.710189] sd 33:0:0:0: [sdb] Mode Sense: 43 00 00 00
[ 6001.714679] sd 33:0:0:0: [sdb] No Caching mode page found
[ 6001.714687] sd 33:0:0:0: [sdb] Assuming drive cache: write through
[ 6001.753858]  sdb: sdb1
[ 6001.770043] sd 33:0:0:0: [sdb] Attached SCSI removable disk
[ 7044.722452] perf: interrupt took too long (2836 > 2500), lowering kernel.perf_event_max_sample_rate to 70500
[ 9440.916577] perf: interrupt took too long (3552 > 3545), lowering kernel.perf_event_max_sample_rate to 56250
root@caine:~#
```

3. After, I displayed information about this flash drive in `/proc/scsi/usb-storage/` and ensured that this info was the same: Netac OnlyDisk.

```
root@caine:~# cat /proc/scsi/usb-storage/2
    Host scsi2: usb-storage
        Vendor: Netac
       Product: OnlyDisk
Serial Number: 98241314200721704l4
      Protocol: Transparent SCSI
     Transport: Bulk
        Quirks:
root@caine:~#
```

4. I checked the size of the image and also ensured it was the same: `15728640` sectors

```
root@caine:~# fdisk -l /dev/sdb
Disk /dev/sdb: 7,5 GiB, 8053063680 bytes, 15728640 sectors
Disk model: OnlyDisk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x629e70e5

Device     Boot Start     End Sectors Size Id Type
/dev/sdb1        128 6285439 6285312   3G  7 HPFS/NTFS/exFAT
root@caine:~#
```

5. I created directories for further mount and hash logs

```
root@caine:~# mkdir /evidence
root@caine:~# mkdir /mnt/evidence
root@caine:~# ls /evidence
root@caine:~# ll /mnt/evidence
total 0
drwxr-xr-x 2 root root 40 mar 24 20:53 ./
drwxr-xr-x 1 root root 60 mar 24 20:53 ../
root@caine:~#
```

6. Then I copied the content of the flash drive to a new disk image and computed MD5 and SHA256 hash values using `dcfldd`:

```
root@caine:~# dcfldd if=/dev/sdb1 of=/mnt/evidence/image.raw hash=md5,sha256 md5log=md5.txt sha256log=sha256.txt sizeprobe=if conv=sync,noerror ha
conv=after
[29% of 3069Mb] 29184 blocks (912Mb) written. 00:04:13 remaining.
29402+0 records in
29401+0 records out
root@caine:~# cat /mnt/
cat: /mnt/: Is a directory
root@caine:~# ls /mnt/evidence/
image.raw
```

7. I compared the hashes I computed and the hashes from Mohamad and found they match, so everything is okay and USB flash drive was not corrupted!

```
root@caine:~# cat md5.txt
50decb45c3d56ffe1a3c538bb7898fd9  -
root@caine:~#
root@caine:~# cat sha256.txt
69bf6a0a408347390fcbebaa99aef95b1e1df81ca24643186daff935b1e88429  -
root@caine:~#
```

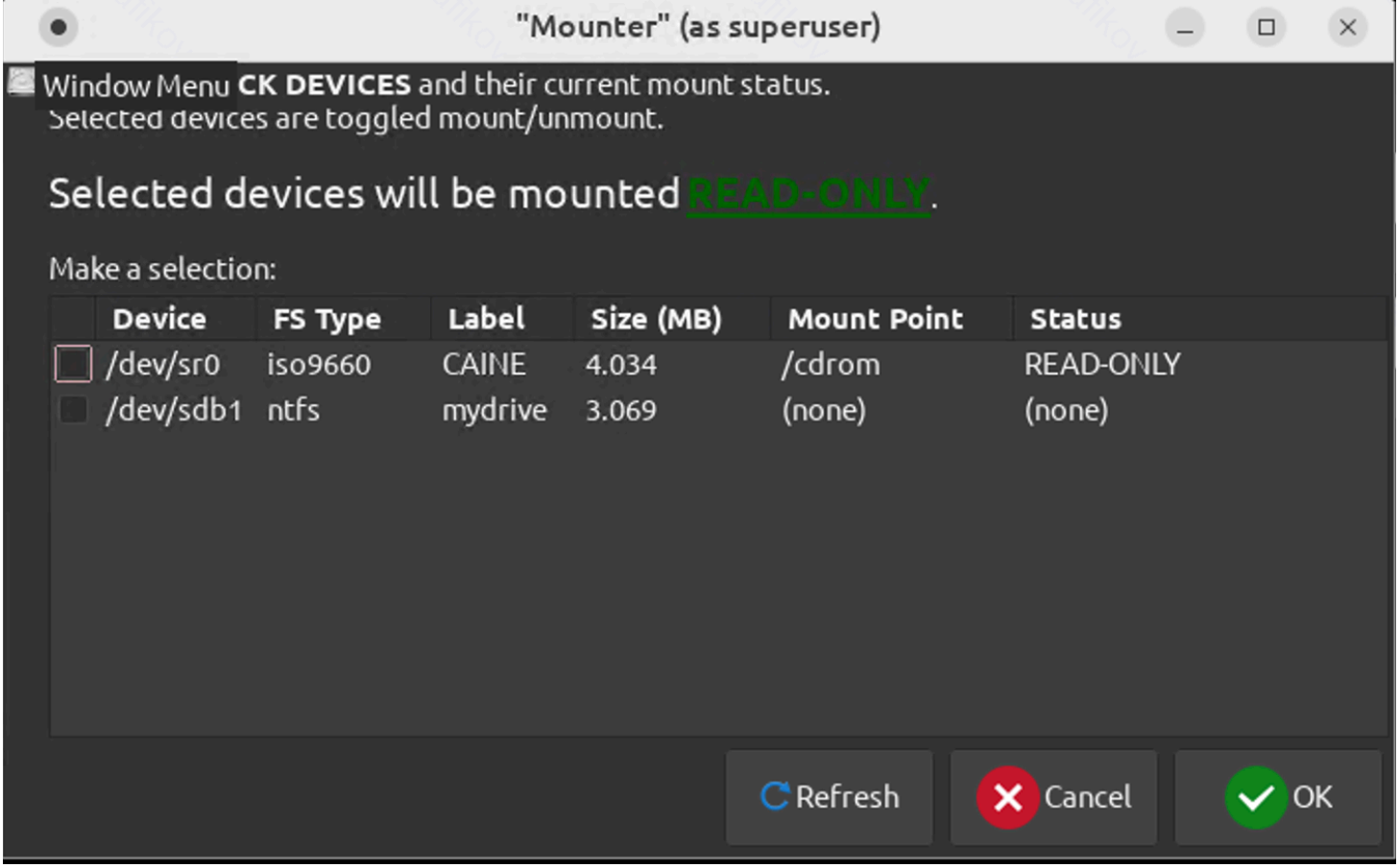
Task 4 - Technical analysis

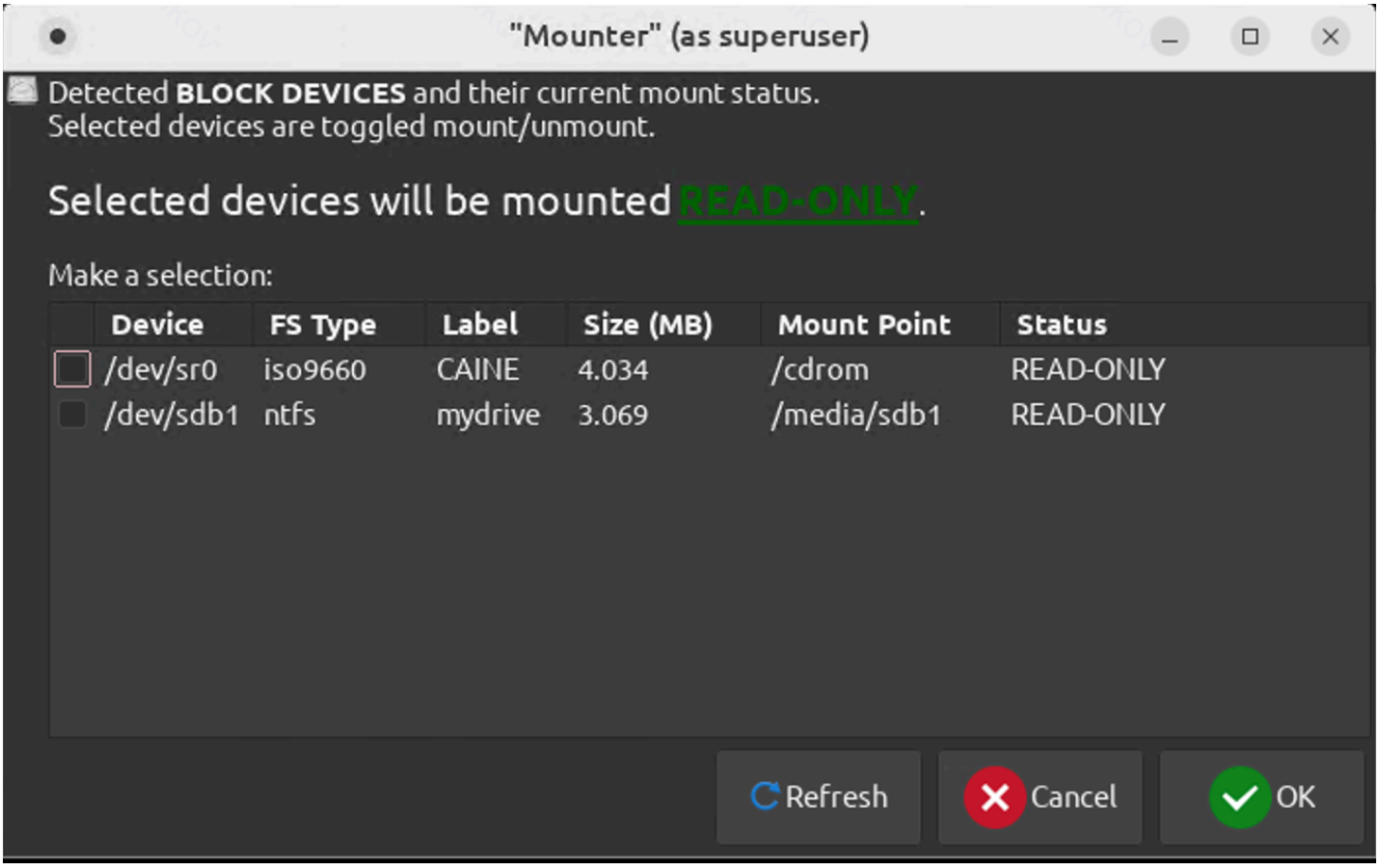
1.

Task description

Mount your image (image of drive A) and make sure that it is mounted as read-only.

- To mount the suspected drive to the system I used the easisest method with Mounter pre-installed application





- Finally, I checked that my device is mounted in read-only now

```
cat /proc/mounts
```

```
root@caine:~# cat /proc/mounts | grep sdb
/dev/loop1 /media/sdb1 fuseblk ro,nodev,noexec,noatime,user_id=0,group_id=0,default_permissions,allow_other,blksize=4096 0 0
root@caine:~#
```

> Note: This flags (`ro` , `noexec` , etc) are essential not to alter the evidence

- We could do the same with `mount` command specifying flags and calculating offsets and sizelimits but it is simply harder :)

2.

Task description

Identify and write a small paragraph of max 200 words about what kind of image it is. Don't go into file specific details just yet. This includes but is not limited to:

- What is the size of the image?
- What partition type(s) does this image have?
- Does it have an MBR/GPT?
- etc.

- To check the size of image I used `fdsk` . The size is 7.52 GiB

```
sudo fdisk -l /dev/sdb
```

```
root@caine:~# fdisk -l /dev/sdb
Disk /dev/sdb: 7,52 GiB, 8074035200 bytes, 15769600 sectors
Disk model: USB2.0 FlashDisk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x629e70e5

Device     Boot Start     End Sectors Size Id Type
/dev/sdb1          128 6285439 6285312   3G  7 HPFS/NTFS/exFAT
root@caine:~#
```

- To check partition type(s) I used `blkid` . The partition type is NTFS, the block size is default $= 512$

```
sudo blkid /dev/sdb1
```

```
root@caine:~# blkid /dev/sdb1
/dev/sdb1: LABEL="mydrive" BLOCK_SIZE="512" UUID="84E6CF86E6CF76C8" TYPE="ntfs" PARTUUID="629e70e5-01"
root@caine:~#
```

- To check MBR/GPT record I used `gdisk` . We see MBR was there but GPT was invalid and there is an overlap.

```
sudo gdisk /dev/sdb1
```

```
root@caine:~# gdisk /dev/sdb1
GPT fdisk (gdisk) version 1.0.8

Partition table scan:
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: not present


***************************************************************
Found invalid GPT and valid MBR; converting MBR to GPT format
in memory. THIS OPERATION IS POTENTIALLY DESTRUCTIVE! Exit by
typing 'q' if you don't want to convert your MBR partitions
to GPT format!
***************************************************************

Exact type match not found for type code 7200; assigning type code for
'Linux filesystem'
Exact type match not found for type code 6C00; assigning type code for
'Linux filesystem'

Warning! Secondary partition table overlaps the last partition by
3883396021 blocks!
You will need to delete this partition or resize it in another utility.
```

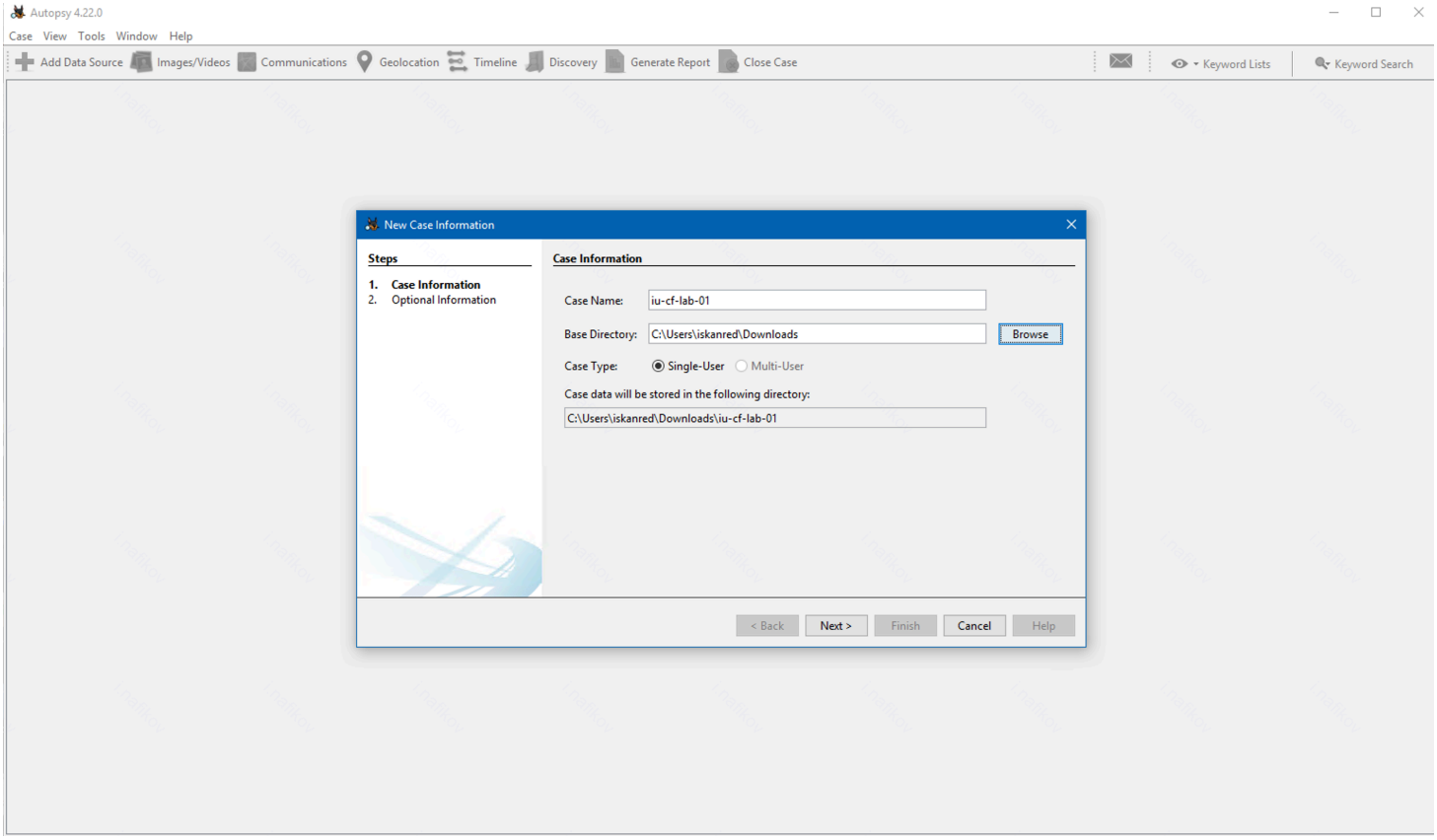
- In total I can state that:
  The 3 GiB disk image contains a corrupted MBR partition table with nonsensical entries, such as 866 GiB partitions on a 3 GiB disk. Although I found the presence of an NTFS filesystem, `gdisk` reports invalid overlaps and alignment problems with the GPT. The discrepancies between the MBR and GPT structures point to significant corruption, most likely resulting from unsuccessful partitioning or formatting attempts. In its current condition, the disk's metadata is damaged and cannot be utilised.

3.

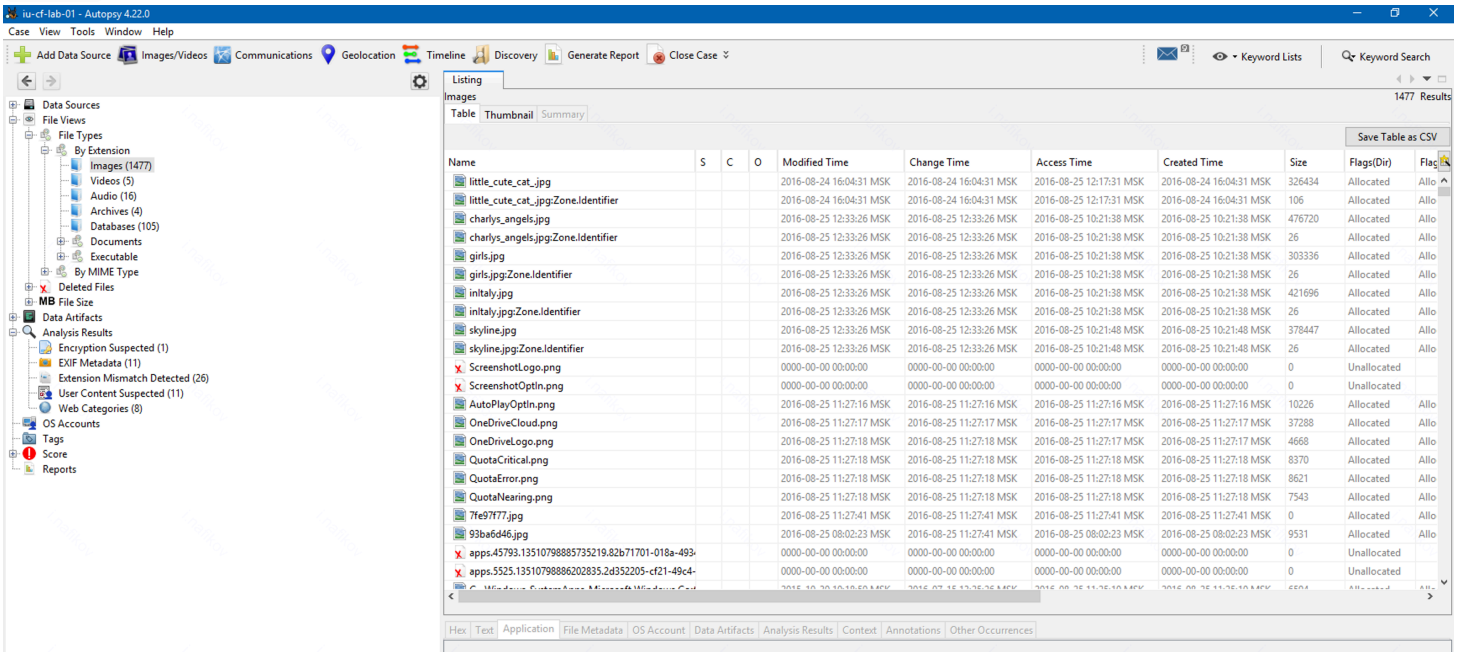
Task description

Using the information from the timeline you create above, write a small paragraph on what you think happened on this specific USB device. The device owner is suspected in a crime. Try to find the evidence that can support this accusation. Please remain objective, as you would be preparing evidence for a court case. Make it a maximum of 300 words, and use timestamps

- First, I downloaded AutoPsy for Windows since it is more modern that is default used in Caine.



- `22:40` : I entered the system and started the search



- `22:52` I felt into user's search history and found something interesting. Why he would need `trucrypt` and `veracrypt`

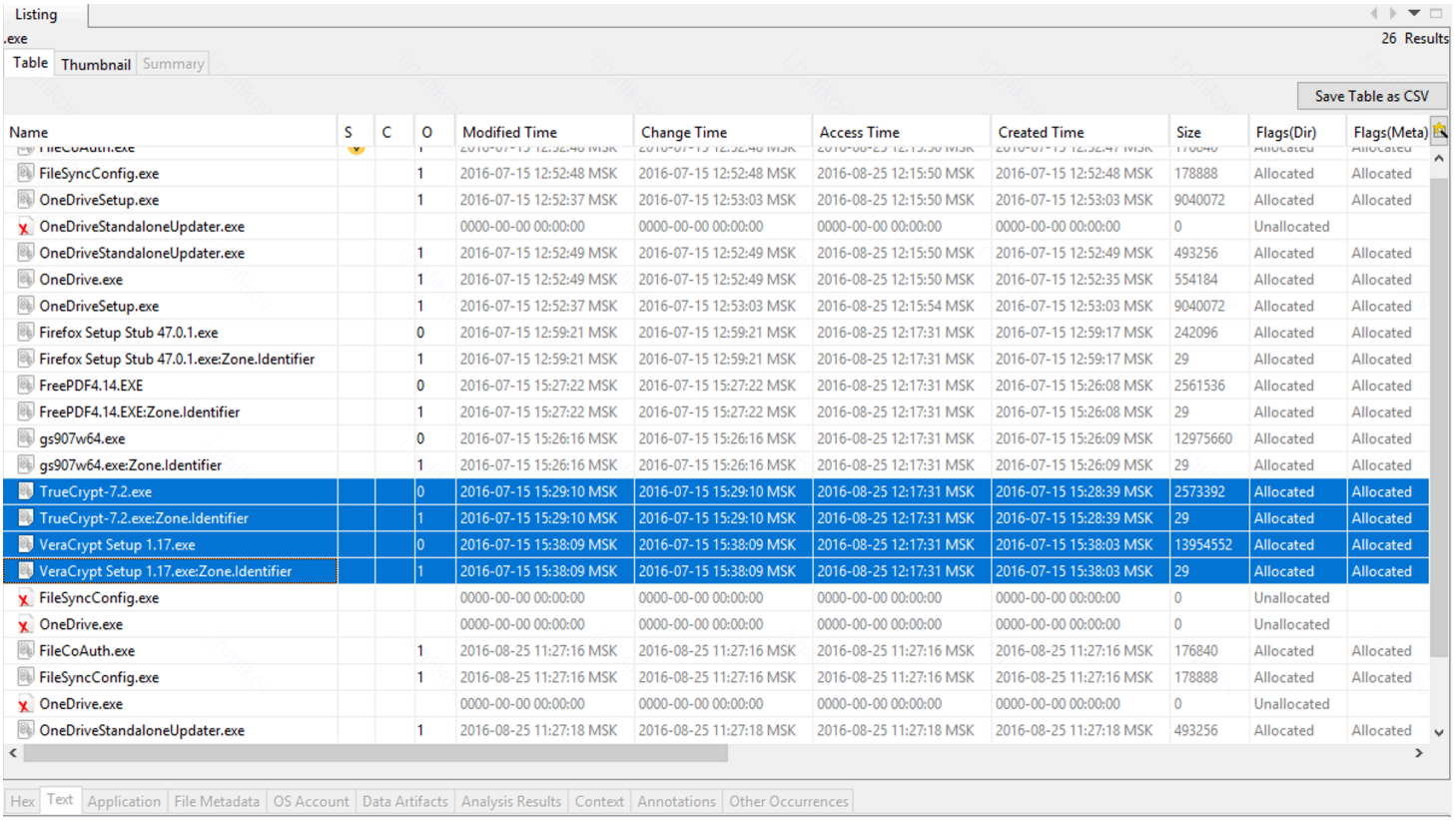
Recent Web Searches

Search String	Date Accessed	Translated
cars	2016/08/25 09:32:52	
freepdf	2016/08/25 09:32:52	
newyork times	2016/08/25 09:32:52	
newyorktimes	2016/08/25 09:32:52	

Recent Web Searches

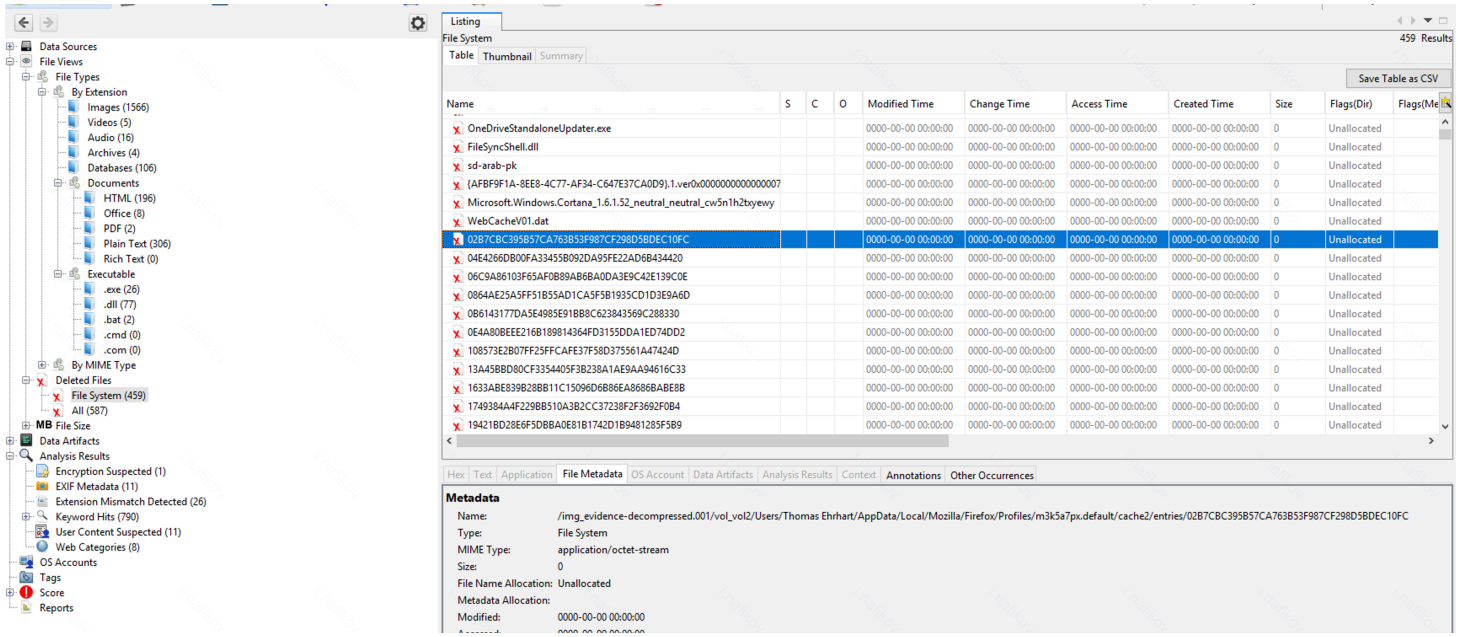
Search String	Date Accessed	Translated
firefox	2016/08/25 09:32:52	
truecrypt	2016/08/25 09:32:52	
veracrypt	2016/08/25 09:32:52	
newyor times	2016/08/25 09:32:52	

- `22:55` I made an assumption that the user actually downloaded and installed these crypto programs



- `23:32` I found he searched for artifact hiding, therefore I made an assumption he wanted to hide something after the some crime. It was one day before the installing crypto programs (`2016-08-25`).

- `23:41` Also, I found that the user removed Mozilla cache file for some reason



- `23:56` I found many text and HTML files that contained JavaScript snippets to extract some info from the client such as cookies

Listing

Plain Text 306 Resu

Table | Thumbnail | Summary

Save Table as CSV

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Know
T2O7PA6D.txt				2016-08-25 11:45:58 MSK	2016-08-25 11:45:58 MSK	2016-08-25 11:45:58 MSK	2016-08-25 11:45:58 MSK	0	Allocated	Allocated	unkno
wr[1].txt			1	2016-08-25 12:27:38 MSK	2016-08-25 12:27:38 MSK	2016-08-25 12:27:38 MSK	2016-08-25 12:27:38 MSK	1	Allocated	Allocated	unkno
wr[2].txt				2016-08-25 12:27:42 MSK	2016-08-25 12:27:42 MSK	2016-08-25 12:27:42 MSK	2016-08-25 12:27:42 MSK	0	Allocated	Allocated	unkno
167IJUAR.txt				2016-08-25 11:46:11 MSK	2016-08-25 11:46:11 MSK	2016-08-25 11:46:11 MSK	2016-08-25 11:46:11 MSK	0	Allocated	Allocated	unkno
f[2].txt			0	2016-07-15 13:00:48 MSK	2016-07-15 13:00:48 MSK	2016-08-25 12:16:37 MSK	2016-07-15 13:00:48 MSK	180197	Allocated	Allocated	unkno
f[1].txt			0	2016-08-25 12:06:49 MSK	2016-08-25 12:06:49 MSK	2016-08-25 12:16:37 MSK	2016-07-15 13:00:48 MSK	1675	Allocated	Allocated	unkno
f[3].txt			0	2016-08-25 12:06:51 MSK	2016-08-25 12:06:51 MSK	2016-08-25 12:16:37 MSK	2016-07-15 13:00:49 MSK	46774	Allocated	Allocated	unkno
f[4].txt			0	2016-07-15 15:25:51 MSK	2016-07-15 15:25:51 MSK	2016-08-25 12:16:37 MSK	2016-07-15 15:25:51 MSK	70008	Allocated	Allocated	unkno
f[5].txt			0	2016-08-25 12:06:51 MSK	2016-08-25 12:06:51 MSK	2016-08-25 12:16:37 MSK	2016-08-25 12:06:51 MSK	70843	Allocated	Allocated	unkno
wr[1].txt			1	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	1	Allocated	Allocated	unkno
wr[2].txt			1	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	1	Allocated	Allocated	unkno
wr[3].txt			1	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	1	Allocated	Allocated	unkno
wr[4].txt			1	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	1	Allocated	Allocated	unkno
wr[5].txt			1	2016-08-25 12:27:40 MSK	2016-08-25 12:27:40 MSK	2016-08-25 12:27:40 MSK	2016-08-25 12:27:40 MSK	1	Allocated	Allocated	unkno
f[1].txt			0	2016-07-15 15:25:51 MSK	2016-07-15 15:25:51 MSK	2016-08-25 12:16:44 MSK	2016-07-15 15:25:51 MSK	158910	Allocated	Allocated	unkno
f[2].txt			0	2016-08-25 12:06:48 MSK	2016-08-25 12:06:48 MSK	2016-08-25 12:16:44 MSK	2016-08-25 12:06:48 MSK	43724	Allocated	Allocated	unkno
f[3].txt			0	2016-08-25 12:06:48 MSK	2016-08-25 12:06:48 MSK	2016-08-25 12:16:44 MSK	2016-08-25 12:06:48 MSK	26674	Allocated	Allocated	unkno
wr[1].txt			1	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	2016-08-25 12:27:37 MSK	1	Allocated	Allocated	unkno
wr[2].txt				2016-08-25 12:27:42 MSK	2016-08-25 12:27:42 MSK	2016-08-25 12:27:42 MSK	2016-08-25 12:27:42 MSK	0	Allocated	Allocated	unkno
f[1].txt			0	2016-07-15 15:25:52 MSK	2016-07-15 15:25:52 MSK	2016-08-25 12:16:52 MSK	2016-07-15 15:25:52 MSK	4008	Allocated	Allocated	unkno
f[2].txt			0	2016-07-15 15:25:52 MSK	2016-07-15 15:25:52 MSK	2016-08-25 12:16:52 MSK	2016-07-15 15:25:52 MSK	96123	Allocated	Allocated	unkno
f[3].txt			0	2016-08-25 12:06:49 MSK	2016-08-25 12:06:49 MSK	2016-08-25 12:16:52 MSK	2016-08-25 12:06:49 MSK	173440	Allocated	Allocated	unkno
f[5].txt			0	2016-08-25 12:08:11 MSK	2016-08-25 12:08:11 MSK	2016-08-25 12:16:52 MSK	2016-08-25 12:08:11 MSK	19	Allocated	Allocated	unkno

Hex | Text | Application | File Metadata | OS Account | Data Artifacts | Analysis Results | Context | Annotations | Other Occurrences

Strings | Extracted Text | Translation

Page: 1 of 1 Page | Matches on page: - of - Match | 100% | Reset | Text Source: File Text

(function(opts_){window.__gcse=window.__gcse||{};window.__gcse.ct=(new Date).getTime();window.__gcse.scb=function(){var a=window.__gcse;a.plainStyle&&delete opts_.rawCss;google.search.cse.element.init(opts_)&&("explicit"!=a.parsetags?"complete"==document.readyState||"interactive"==document.readyState?(google.search.cse.element.go(),a.callback&&a.callback()):google.setOnLoadCallback(function(){google.search.cse.element.go();a.callback&&a.callback()},!0):a.callback&&a.callback())};
var b=document.createElement("script"),c=opts_.protocol+"://"+opts_.uds+"/jsapi?autoload=",d=encodeURIComponent,e='{"name":"search","version":"1.0","callback":"__gcse.scb"',f=window.__gcse;if(!f||!f.plainStyle)var g=opts_.protocol+(opts_.uiOptions&&opts_.uiOptions.enableMobileLayout?"://www.google.com/cse/style/look/mobile/":"://www.google.com/cse/style/look/")+opts_.theme.toLowerCase().replace("v2_","v2/")+".css",e=e+(',"style":"'+g+'"');opts_.language&&(e+=',"language":"'+opts_.language+'"');
e+="}";b.src=c+d('{"modules":['+e+"]}")+"";b.type="text/javascript";document.getElementsByTagName("head")[0].appendChild(b);
})({"cx":"007671019323140114033:nvfwtfochm8","language":"en","theme":"V2_DEFAULT","uiOptions":{"resultsUrl":"http://www.tenforums.com/gtsearch.php","enableAutoComplete":true,"enableImageSearch":false,"imageSearchLayout":"popup","resultSetSize":"filtered_cse","enableOrderBy":true,"orderByOptions":[{"label":"Relevance","key":""},{"label":"Date","key":"date"}],"overlayResults":false,"queryParameterName":"q","enableMobileLayout":false,"numTopRefinements":2147483647,"isSitesearch":false,"enableSpeech":true},"protocol":"http","uds":"www.google.com","rawCss":"\n"});

Listing

HTML 196 Result

Table | Thumbnail | Summary

Save Table as CSV

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(M
download[1].htm			0	2016-07-15 15:28:31 MSK	2016-07-15 15:28:31 MSK	2016-08-25 12:16:37 MSK	2016-07-15 15:28:31 MSK	382	Allocated	Allocate
emily[2].htm			0	2016-07-15 13:00:49 MSK	2016-07-15 13:00:49 MSK	2016-08-25 12:16:37 MSK	2016-07-15 13:00:48 MSK	30885	Allocated	Allocate
imsync[1].htm			1	2016-07-15 15:29:18 MSK	2016-07-15 15:29:18 MSK	2016-08-25 12:16:37 MSK	2016-07-15 15:29:18 MSK	17	Allocated	Allocate
index[1].htm			0	2016-08-25 12:06:58 MSK	2016-08-25 12:06:58 MSK	2016-08-25 12:16:37 MSK	2016-08-25 12:06:58 MSK	85390	Allocated	Allocate
new[1].htm			0	2016-07-15 12:59:16 MSK	2016-07-15 12:59:16 MSK	2016-08-25 12:16:39 MSK	2016-07-15 12:59:16 MSK	34831	Allocated	Allocate
Pa new[1].htm			1	2016-07-15 15:25:32 MSK	2016-07-15 15:25:32 MSK	2016-08-25 12:16:39 MSK	2016-07-15 15:25:32 MSK	318	Allocated	Allocate
Passport[2].htm			0	2016-07-15 13:00:24 MSK	2016-07-15 13:00:24 MSK	2016-08-25 12:16:39 MSK	2016-07-15 13:00:24 MSK	320	Allocated	Allocate
Passport[3].htm			1	2016-08-24 16:04:13 MSK	2016-08-24 16:04:13 MSK	2016-08-25 12:16:39 MSK	2016-08-24 16:04:13 MSK	318	Allocated	Allocate
postmessageRelay[1].htm			0	2016-08-25 12:08:08 MSK	2016-08-25 12:08:08 MSK	2016-08-25 12:16:39 MSK	2016-08-25 12:08:08 MSK	506	Allocated	Allocate
s2[1].htm			0	2016-07-15 15:25:53 MSK	2016-07-15 15:25:53 MSK	2016-08-25 12:16:39 MSK	2016-07-15 15:25:53 MSK	1424	Allocated	Allocate
s2[2].htm			0	2016-07-15 15:28:36 MSK	2016-07-15 15:28:36 MSK	2016-08-25 12:16:39 MSK	2016-07-15 15:28:36 MSK	305	Allocated	Allocate
Suggestions[1].htm			0	2016-07-15 13:00:12 MSK	2016-07-15 13:00:12 MSK	2016-08-25 12:16:40 MSK	2016-07-15 13:00:12 MSK	5911	Allocated	Allocate
s[2].htm			0	2016-08-25 12:09:54 MSK	2016-08-25 12:09:54 MSK	2016-08-25 12:16:40 MSK	2016-08-25 12:09:54 MSK	143	Allocated	Allocate
tweet_button.d73d0c4cb6af3df0ea22b7c11dbc87d2			0	2016-08-25 12:08:12 MSK	2016-08-25 12:08:12 MSK	2016-08-25 12:16:41 MSK	2016-08-25 12:08:12 MSK	30306	Allocated	Allocate
uN4_cXtJDGb[1].htm			0	2016-08-25 12:08:11 MSK	2016-08-25 12:08:11 MSK	2016-08-25 12:16:41 MSK	2016-08-25 12:08:10 MSK	33528	Allocated	Allocate
YC4GBXZY.htm			0	2016-07-15 15:28:06 MSK	2016-07-15 15:28:06 MSK	2016-08-25 12:16:41 MSK	2016-07-15 15:28:06 MSK	9272	Allocated	Allocate
zrt_lookup[1].htm			0	2016-07-15 13:00:49 MSK	2016-07-15 13:00:49 MSK	2016-08-25 12:16:41 MSK	2016-07-15 13:00:48 MSK	13796	Allocated	Allocate
MsaCallback[1].htm			0	2016-08-25 11:46:18 MSK	2016-08-25 11:46:18 MSK	2016-08-25 11:46:18 MSK	2016-08-25 11:46:18 MSK	2627	Allocated	Allocate
MsaCallback[2].htm			0	2016-08-25 12:31:38 MSK	2016-08-25 12:31:38 MSK	2016-08-25 12:31:38 MSK	2016-08-25 12:31:38 MSK	2624	Allocated	Allocate
msasso[1].htm			0	2016-08-25 12:27:36 MSK	2016-08-25 12:27:36 MSK	2016-08-25 12:27:35 MSK	2016-08-25 12:27:35 MSK	756	Allocated	Allocate
958737122[1].htm			0	2016-08-25 12:27:35 MSK	2016-08-25 12:27:35 MSK	2016-08-25 12:27:35 MSK	2016-08-25 12:27:35 MSK	451	Allocated	Allocate
ads[1].htm			0	2016-07-15 13:00:49 MSK	2016-07-15 13:00:49 MSK	2016-08-25 12:16:42 MSK	2016-07-15 13:00:49 MSK	109972	Allocated	Allocate

Hex | Text | Application | File Metadata | OS Account | Data Artifacts | Analysis Results | Context | Annotations | Other Occurrences

Strings | Extracted Text | Translation

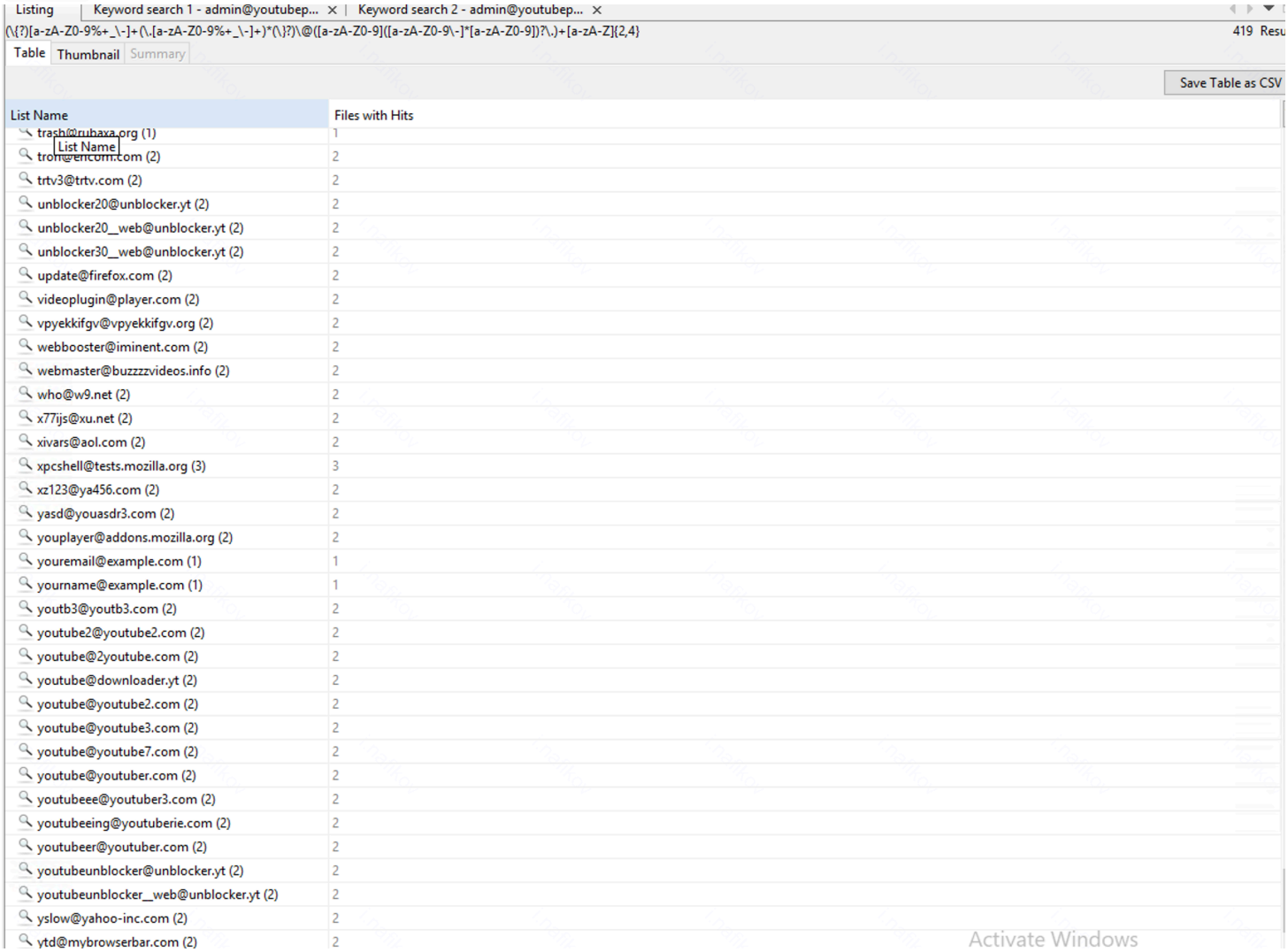
Page: 1 of 1 Page | Matches on page: - of - Match | 100% | Reset | Text Source: File Text

Loading...

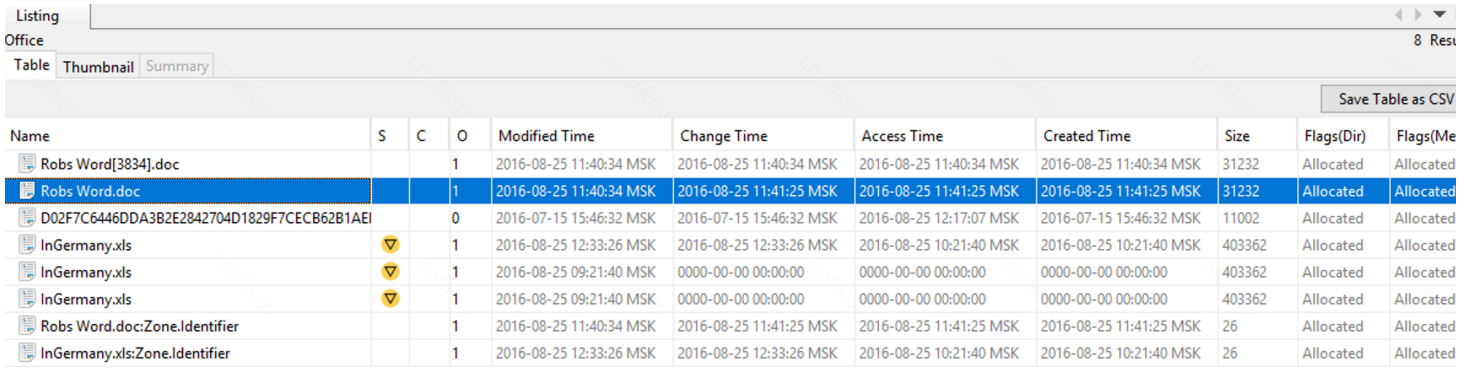
----------METADATA----------

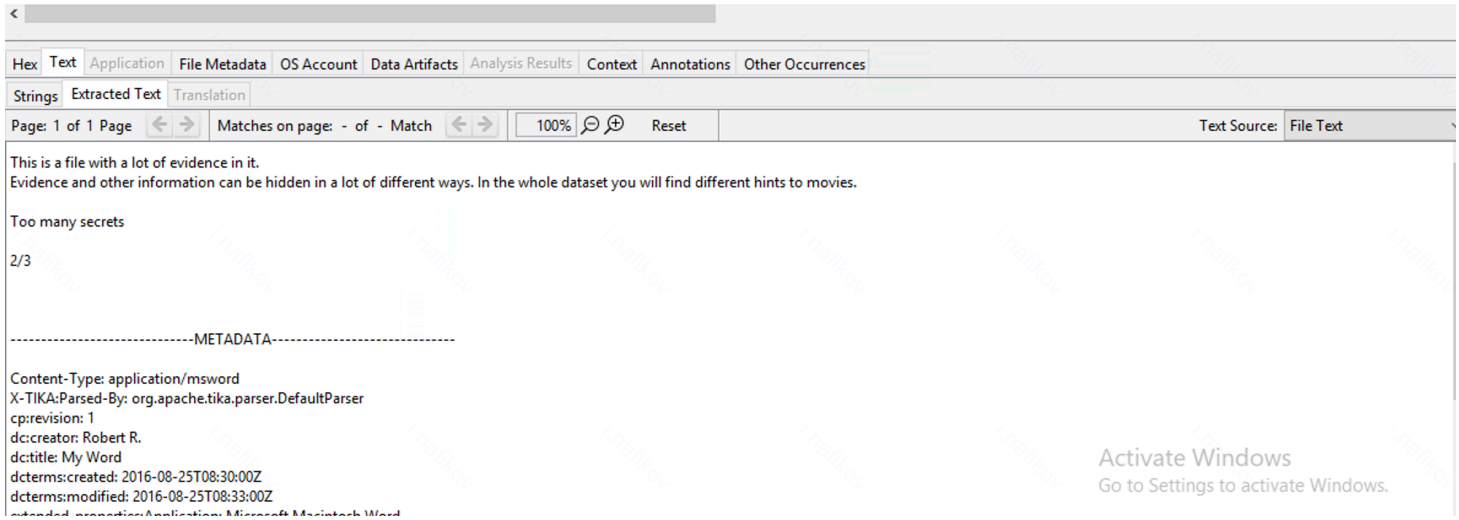
Scripts:
1) type="text/javascript" //<![CDATA[
var _w = window; var o = _w.opener; var mainWindow; (mainWindow = o) || (mainWindow = _w.parent); if (mainWindow) {mainWindow.sj_evt && mainWindow.sj_evt.fire("wl:cancel"); };if (o) _w.close();;

- `00:13` Also, I found many suspicious addresses

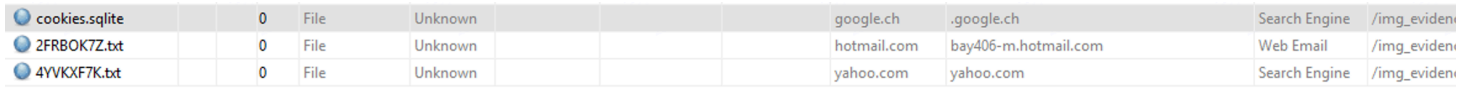


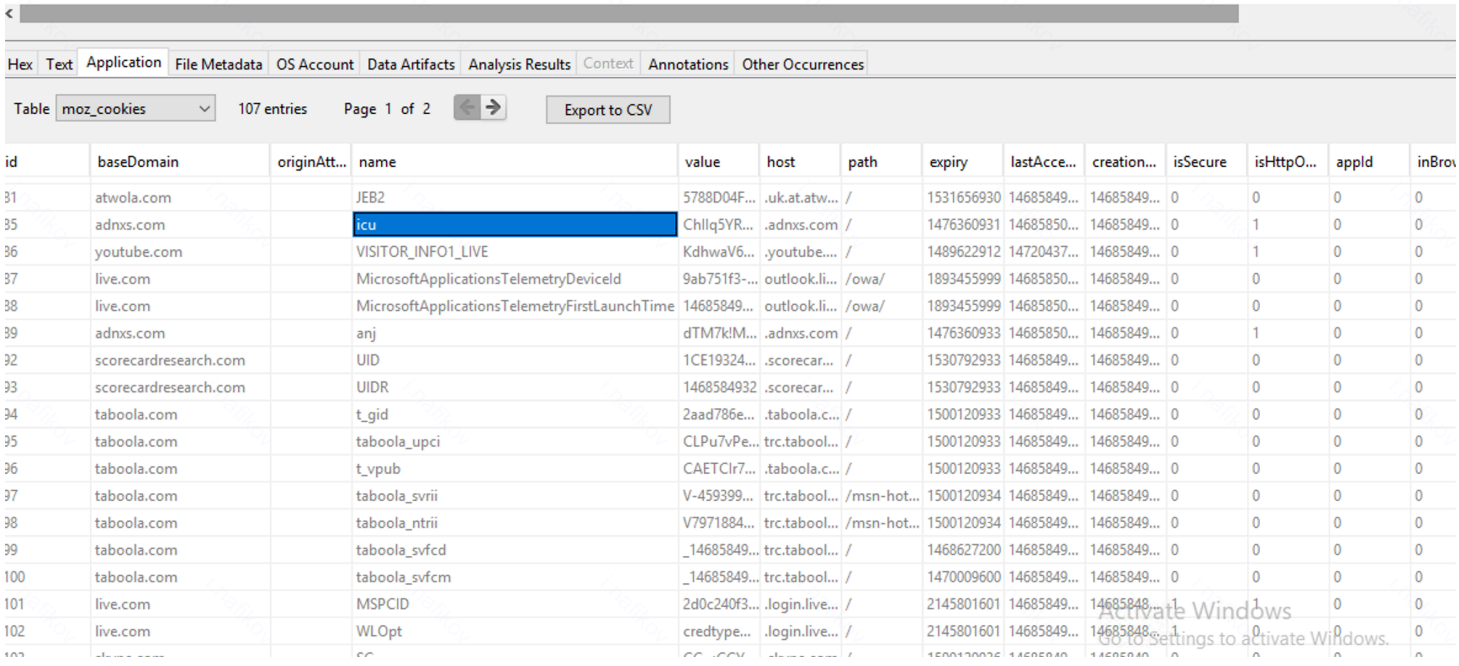
- `00:16` Also, I found funny hints





- `00:21` Finally, many suspicious cookies was found with repetitions and non-real usernames





- `00:34` So, based on all this information I made an assumption that:

```
The user was involved in cookie theft, as evidenced by a significant collection of
cookies from various websites. These cookies were likely exploited to create
fraudulent sessions impersonating victims. His Google search history suggests
attempts to obfuscate his activities, indicating a conscious effort to hide his
actions.

Additionally, numerous suspicious JavaScript scripts were found embedded in HTML
pages, along with fake email addresses in his address book, hinting at possible
spamming schemes. He downloaded security tools like TrueCrypt and VeraCrypt while
on Windows 10, which may have been used for file encryption or concealment.

There is a strong likelihood that he distributed binary files to capture cookies
and manipulated the Zone Identifier to alter Alternate Data Streams on victims'
systems. His deliberate erasure of web cache further points to a sophisticated
understanding of digital forensics and cybercrime. Lastly, his use of OneDrive
indicates potential file transfer between multiple devices.
```

4.

Task description

What would help to investigate this evidence further?

- Analysis of files and directories for deeper insights.
- Examination of deleted files and those with Unicode-based names.
- Reviewing raw file content and extract hexadecimal and ASCII strings.
- Performing a hash database check to assess potential malicious activity.
- Establishing a timeline of file activity to reveal system interactions and highlight evidence.
- Analyzing timestamps for valuable information on the sequence of events and data recovery.
- Conducting metadata analysis to recover deleted content and identify file locations.
- Correlating time-based events (file modifications, IDS alerts, firewall logs) for a comprehensive view of network activity.