IFS4102 LAB WEEK 4

OBJECTIVES

1. Inspect disk image file using TSK **(Task 1)**
2. Perform file signature analysis and fix mismatched extensions **(Task 2)**
3. Extract and view metadata of Microsoft office files and image files (**Task 3-A&B**)
4. Using Autopsy to extract deleted files and execute ingest modules **(Task 4-A,B,C&D)**

REMINDER: WEEK 4 GRADED LAB TASKS #2
SATURDAY, 11 FEBRUARY 2023, 23:59 SGT
USE THE GIVEN SAMPLE FILES

1. For questions 1 & 2, use **SuspectDrive1.E01**
2. For question 3, use **SecretFile.docx**

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

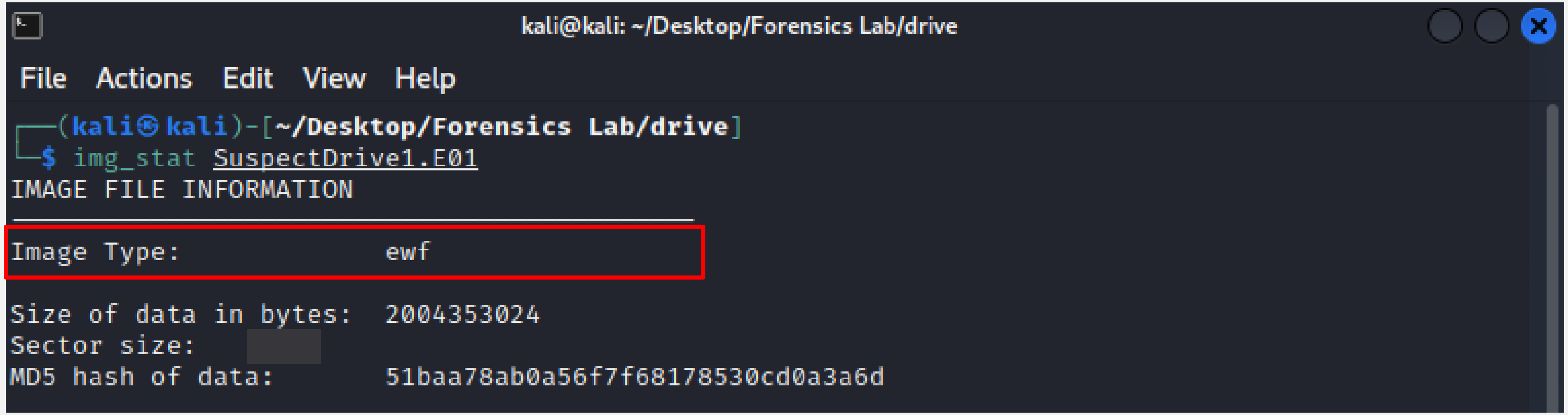
- Linux forensic workstation
- Open-sourced, CLI toolkit for analysing Microsoft and UNIX file systems and disk
- Autopsy is the GUI of TSK

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- TSK is a command line utility:
- More like a toolkit rather than an application:
  - img_stat
  - mmstat
  - mmls
  - fsstat
  - fls
  - and more at https://github.com/sleuthkit/sleuthkit/tree/develop/man

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- Tool: `img_stat`
- What it does? – Displays information of a disk image file
- Typical usage:
  - `img_stat <imagefile>`
  - Output:
    - **Image type**
    - Size of data
    - Sector size
    - MD5 Hash
- Output of `img_stat` is used as input for other tools in TSK, should always run `img_stat` first.
- Note down image type


kali@kali: ~/Desktop/Forensics Lab/drive
File Actions Edit View Help

```
┌──(kali㉿kali)-[~/Desktop/Forensics Lab/drive]
└─$ img_stat SuspectDrive1.E01
IMAGE FILE INFORMATION
────────────────────────────────────────────
Image Type:               ewf

Size of data in bytes:    2004353024
Sector size:    [redacted]
MD5 hash of data:         51baa78ab0a56f7f68178530cd0a3a6d
```

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

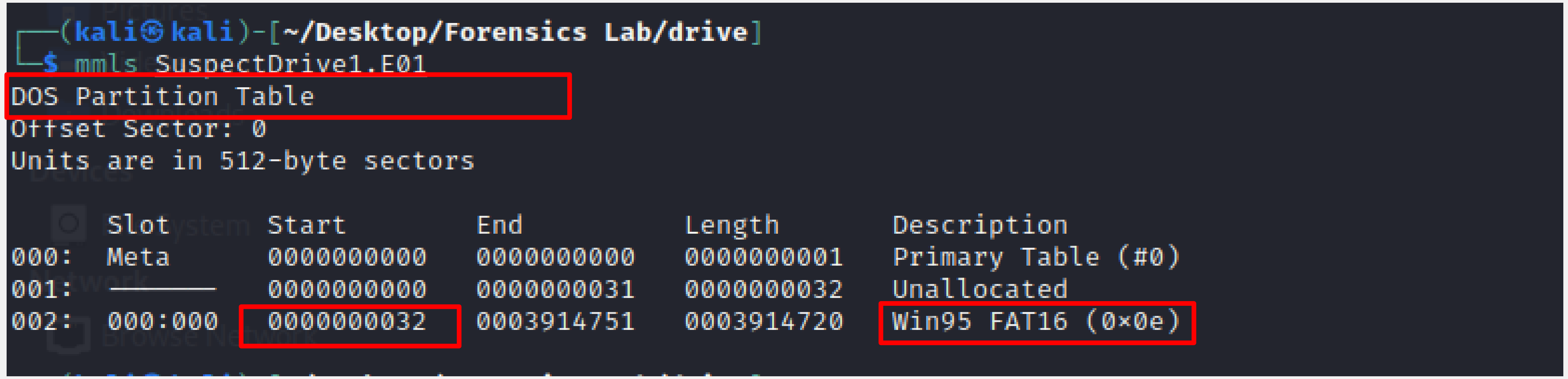
- Tool: `mmstat`
- What it does? – Displays detail about the volume system (i.e., partition table)
- Typical usage:
  - `mmstat <`**`imagefile`**`>`
  - Output: partition table type (e.g., dos)
- Usually I use `mmls` to check the type of partition, but can use both to double check make sure the type matches.

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- Tool: `mmls`
- What it does? – Displays partition layout of a volume system, including unallocated spaces.
- Typical usage:
  - `mmls <`**`imagefile`**`>`
  - Output:
    - type of partition table (e.g., dos)
    - Offset sector
    - Sector size
    - Layout of the disk in table form
      - Index
      - **Start** and end sector number
      - Length
      - **File System Type**

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- Tool: `mmls`
- Output of `mmls` is used as input for other tools in TSK, should always run `mmls`.
- Note down the file system type and associated start sector number



```
┌──(kali㉿kali)-[~/Desktop/Forensics Lab/drive]
└─$ mmls SuspectDrive1.E01
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------   0000000000   0000000031   0000000032   Unallocated
002:  000:000   0000000032   0003914751   0003914720   Win95 FAT16 (0x0e)
```

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

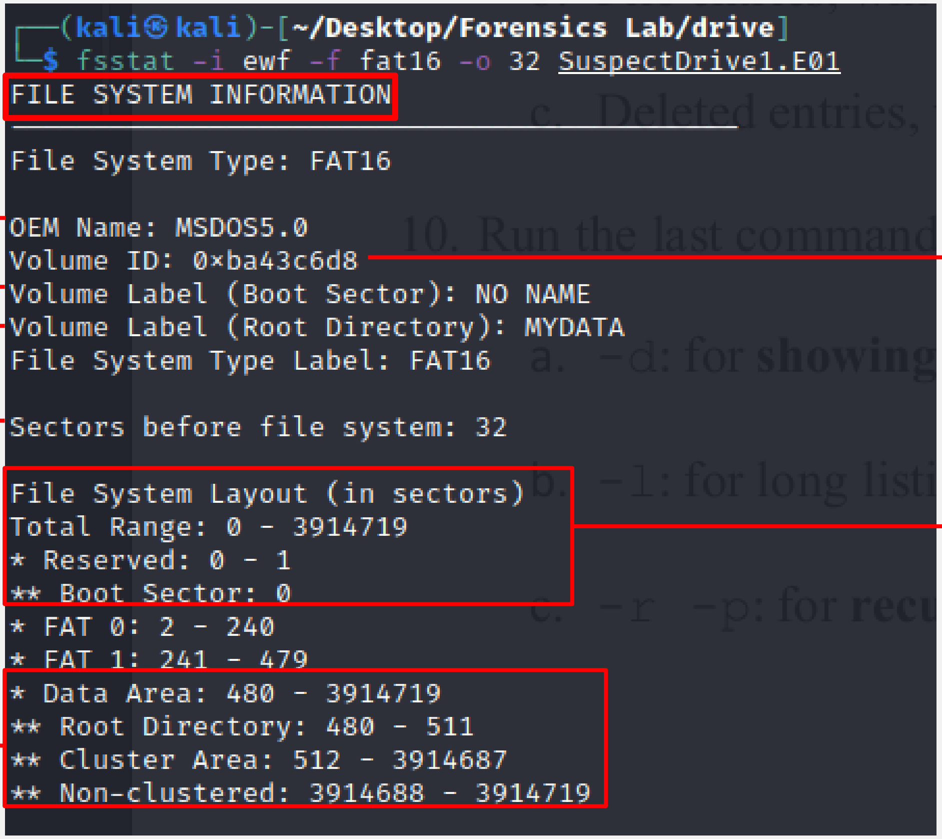
- Tool: `fsstat`
- What it does? – Displays the general details of a file system
- From `img_stat` and `mmls`, identified image type, file system type and start offset.
- Typical usage:
  - `fsstat –i list`
  - `fsstat –f list`
  - `fsstat –i <`**`image type`**`> -f <`**`file system type`**`> -o <`**`starting offset`**`> <`**`imagefile`**`>`
  - `fsstat –i ewf -f fat16 -o 32 SuspectDrive1.E01`

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- Output: 4 sections
  - File system information
  - Metadata information
  - Content information
  - FAT content

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- File system information



```
┌──(kali㉿kali)-[~/Desktop/Forensics Lab/drive]
└─$ fsstat -i ewf -f fat16 -o 32 SuspectDrive1.E01
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT16

OEM Name: MSDOS5.0
Volume ID: 0xba43c6d8
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): MYDATA
File System Type Label: FAT16

Sectors before file system: 32

File System Layout (in sectors)
Total Range: 0 - 3914719
* Reserved: 0 - 1
** Boot Sector: 0
* FAT 0: 2 - 240
* FAT 1: 241 - 479
* Data Area: 480 - 3914719
** Root Directory: 480 - 511
** Cluster Area: 512 - 3914687
** Non-clustered: 3914688 - 3914719
```

Typically, the OS or application which formatted the file system

Two location to store volume label

How many sectors exist before the file system. I.e., starting offset of the file system

NOT the serial ID. Assigned by the system that formatted the drive.

All in sectors!

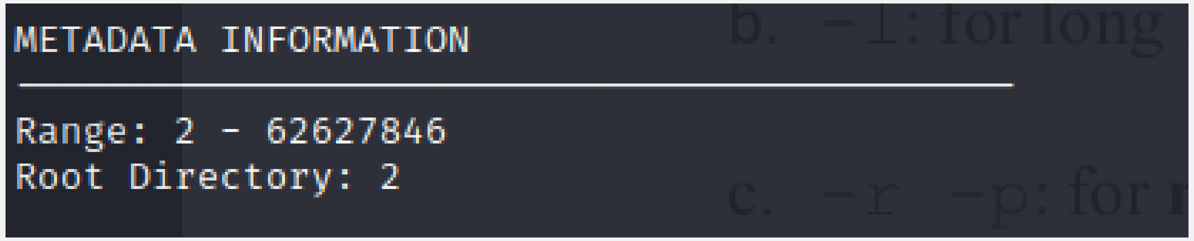
- 3914720 sectors in file system.
- Sector 0 to 1 are in the reserve area.
- Boot sector in sector 0.
- Information displayed in a tree like format ('*' shows how deep)

All in sectors!

- Sector 480 to 3914719 data area
- Sector 480 to 511 root directory
- Sector 512 to 3914687 cluster area
- 32 sectors not allocated to a cluster (Sector 3914688 to 3914719)

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

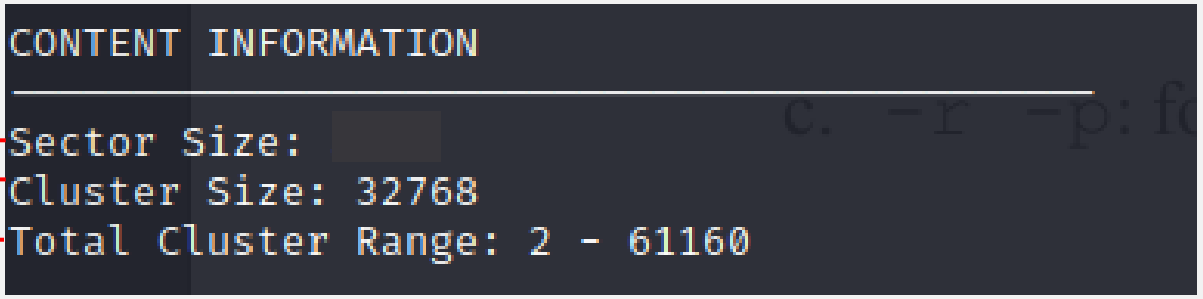
- Metadata information
  - Valid range of addresses to use with `icat` or `istat`
  - Based on total number of sectors in the file system
  - Not very important for us, this slide is FYI.



```
METADATA INFORMATION
--------------------------------------------
Range: 2 - 62627846
Root Directory: 2
```

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- Content information



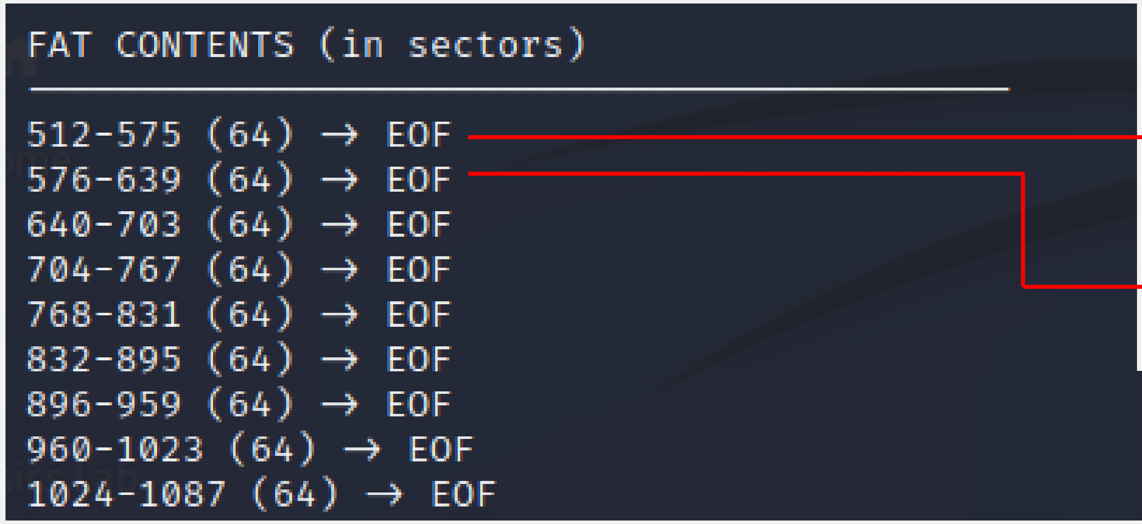
```
CONTENT INFORMATION
--------------------------------------------
Sector Size: [redacted]
Cluster Size: 32768
Total Cluster Range: 2 - 61160
```

Size in bytes

Total addressable cluster range, even though TSK show all addresses in sectors.

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

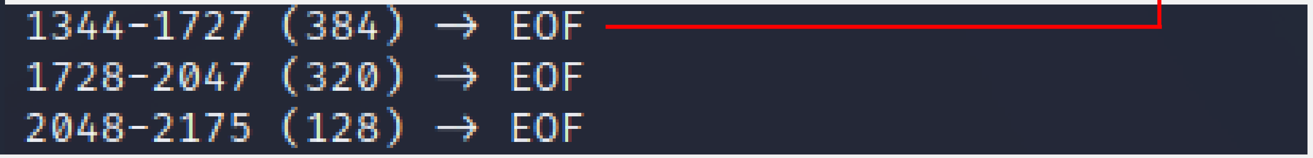
- FAT contents (to be discussed in Lecture 5)
  - Graphical representation of the primary FAT structure.
  - Each line corresponds to a "cluster run"
  - If cluster is not allocated then no entry in this section



```
FAT CONTENTS (in sectors)
________________________________________
512-575 (64) -> EOF
576-639 (64) -> EOF
640-703 (64) -> EOF
704-767 (64) -> EOF
768-831 (64) -> EOF
832-895 (64) -> EOF
896-959 (64) -> EOF
960-1023 (64) -> EOF
1024-1087 (64) -> EOF
```

- Sector 512 to 575 = 64 sectors
- For Cluster 2

- Sector 576 to 639 = 64 sectors
- For Cluster 3



```
1344-1727 (384) -> EOF
1728-2047 (320) -> EOF
2048-2175 (128) -> EOF
```

- Not always 1 cluster
- Sector 1344 to 1727 = 384 sectors
- This file allocated 6 consecutive cluster (384/64=6)
- Grouped together if consecutive, otherwise, instead of → EOF, will see → (sector number)
- For cluster X, X+1, X+2 ... X+N

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

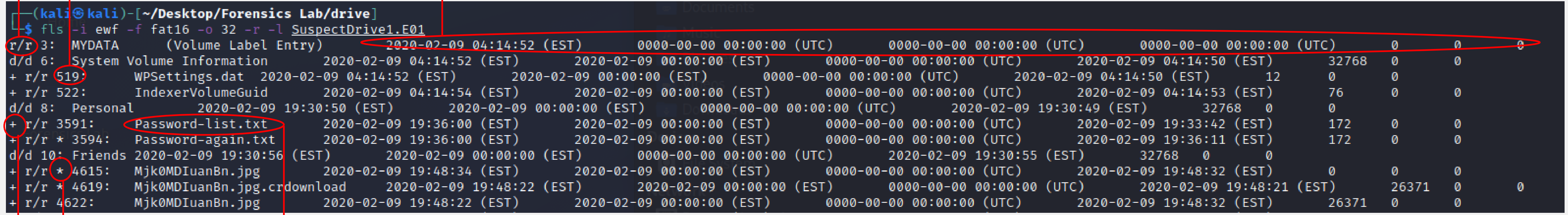
- Tool: `fls`
- What it does? – List file and directory names in a disk image
- From `img_stat` and `mmls`, identified image type, file system type and start offset.
- Typical usage (similar to `fsstat`):
  - `fls -i list`
  - `fls -f list`
  - `fls -i <`**`image type`**`> -f <`**`file system type`**`> -o <`**`starting offset`**`> <`**`imagefile`**`>`
  - Above command only show directory listings
  - `-r` show all files, `-p` show full pathnames, -l show MAC time (-l is lower case L).

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

File type. First letter is saved in file name structure. Second letter saved in metadata structure.
For allocated files, should always be equal. For delete files, could be different.
Types: **r = regular file, d = directory**, c = character device, b = block device, l = symbolic link, p = named FIFO, s = shadow,
h = socket, w = whiteout, **v = TSK virtual file/directory (not a real directory, created by TSK for convenience)**

Metadata address

'-l' (lower case L) show, in order, last modified time, last accessed time, last changed time, created time, size in bytes, UID, GID



```
┌──(kali㉿kali)-[~/Desktop/Forensics Lab/drive]
└─$ fls -i ewf -f fat16 -o 32 -r -l SuspectDrive1.E01
r/r 3:  MYDATA      (Volume Label Entry)        2020-02-09 04:14:52 (EST)       0000-00-00 00:00:00 (UTC)       0000-00-00 00:00:00 (UTC)       0000-00-00 00:00:00 (UTC)       0       0       0
d/d 6:  System Volume Information       2020-02-09 04:14:52 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 04:14:50 (EST)       32768   0       0
+ r/r 519:      WPSettings.dat  2020-02-09 04:14:52 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 04:14:50 (EST)       12      0       0
+ r/r 522:      IndexerVolumeGuid       2020-02-09 04:14:54 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 04:14:53 (EST)       76      0       0
d/d 8:  Personal        2020-02-09 19:30:50 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 19:30:49 (EST)       32768   0       0
+ r/r 3591:     Password-list.txt       2020-02-09 19:36:00 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 19:33:42 (EST)       172     0       0
+ r/r * 3594:   Password-again.txt      2020-02-09 19:36:00 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 19:36:11 (EST)       172     0       0
d/d 10: Friends 2020-02-09 19:30:56 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 19:30:55 (EST)       32768   0       0
+ r/r * 4615:   Mjk0MDIuanBn.jpg        2020-02-09 19:48:34 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 19:48:32 (EST)       0       0       0
+ r/r * 4619:   Mjk0MDIuanBn.jpg.crdownload     2020-02-09 19:48:22 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 19:48:21 (EST)       26371   0       0
+ r/r 4622:     Mjk0MDIuanBn.jpg        2020-02-09 19:48:22 (EST)       2020-02-09 00:00:00 (EST)       0000-00-00 00:00:00 (UTC)       2020-02-09 19:48:32 (EST)       26371   0       0
```

'*' means deleted

File/directory name

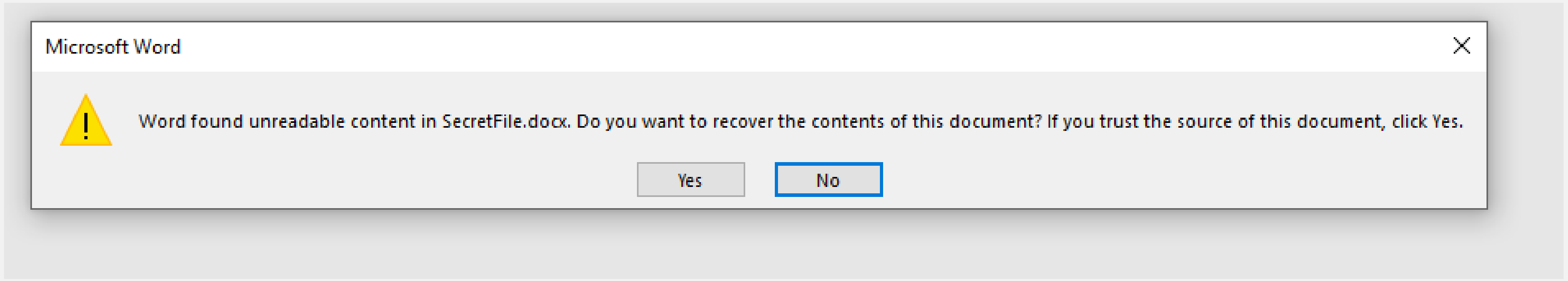
If –r is used, '+' is added to the front of each entry to show how deep the file is.
In this example, password-list.txt is 1 directory deep, in the personal directory

1. INSPECT DISK IMAGE FILE USING TSK **(TASK 1)**

- Documentation
  - https://github.com/sleuthkit/sleuthkit/tree/develop/man
  - Use the tool without argument, example: *`img_stat`*

2. PERFORM FILE SIGNATURE ANALYSIS AND FIX MISMATCHED EXTENSIONS **(TASK 2)**

- Use hex editor – WinHex or FTK Imager
- Use online library https://www.garykessler.net/library/file_sigs.html
- Using the online library, find specific patterns in the header and/or footer which will help identify the correct extension.


Microsoft Word

Word found unreadable content in SecretFile.docx. Do you want to recover the contents of this document? If you trust the source of this document, click Yes.

3. EXTRACT AND VIEW METADATA OF MICROSOFT OFFICE FILES AND IMAGE FILES (**TASK 3-A&B**)

Task 3A

- Change Microsoft Office file to .zip extension
- Open and navigate to docProps > core.xml
- Inspect metadata

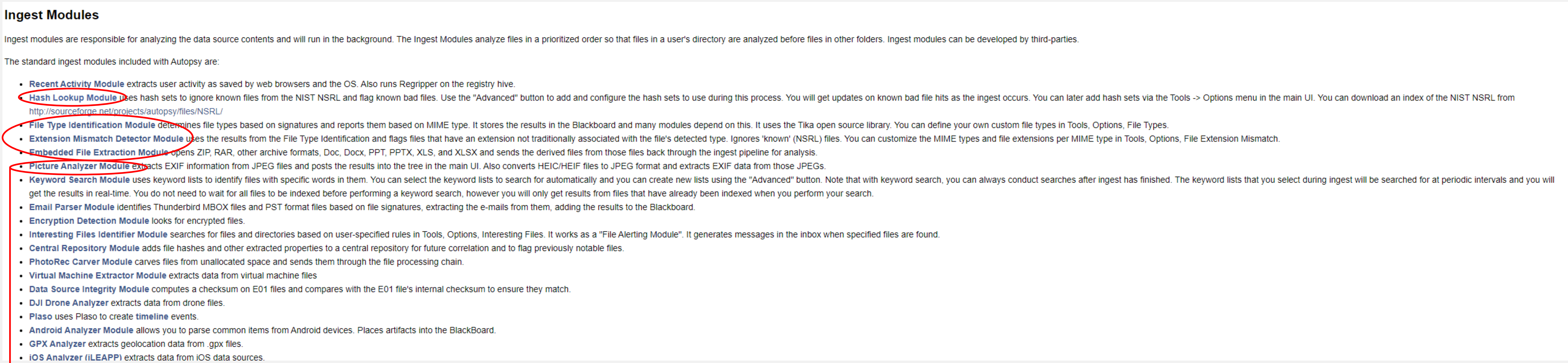
Task 3B

- Use web-based tool http://exif.regex.info/exif.cgi

4. USING AUTOPSY TO EXTRACT DELETED FILES AND EXECUTE INGEST MODULES (**TASK 4-A,B,C&D**)

- Introduced Autopsy last week, manual analysis
- Automated analysis using Ingest modules
- Demo with “SuspectDrive1.E01”
  - **Task 4-A:** Filter all deleted files (Automatic once add data source, no need to run any ingest modules)
  - **Task 4-B:** Perform File Type identification based on internal signatures (Ingest module: File Type Identification)
  - **Task 4-C:** Identify mismatched extensions (Ingest module: File Type Identification and Extension Mismatch Detector)
  - **Task 4-D:** Find if a file exist using hash-lookup (Ingest module: Hash Lookup)

⭕ = demo just now/inside lab notes



Ingest Modules

Ingest modules are responsible for analyzing the data source contents and will run in the background. The Ingest Modules analyze files in a prioritized order so that files in a user's directory are analyzed before files in other folders. Ingest modules can be developed by third-parties.

The standard ingest modules included with Autopsy are:

- **Recent Activity Module** extracts user activity as saved by web browsers and the OS. Also runs Regripper on the registry hive.
- **Hash Lookup Module** uses hash sets to ignore known files from the NIST NSRL and flag known bad files. Use the "Advanced" button to add and configure the hash sets to use during this process. You will get updates on known bad file hits as the ingest occurs. You can later add hash sets via the Tools -> Options menu in the main UI. You can download an index of the NIST NSRL from http://sourceforge.net/projects/autopsy/files/NSRL/
- **File Type Identification Module** determines file types based on signatures and reports them based on MIME type. It stores the results in the Blackboard and many modules depend on this. It uses the Tika open source library. You can define your own custom file types in Tools, Options, File Types.
- **Extension Mismatch Detector Module** uses the results from the File Type Identification and flags files that have an extension not traditionally associated with the file's detected type. Ignores 'known' (NSRL) files. You can customize the MIME types and file extensions per MIME type in Tools, Options, File Extension Mismatch.
- **Embedded File Extraction Module** opens ZIP, RAR, other archive formats, Doc, Docx, PPT, PPTX, XLS, and XLSX and sends the derived files from those files back through the ingest pipeline for analysis.
- **Picture Analyzer Module** extracts EXIF information from JPEG files and posts the results into the tree in the main UI. Also converts HEIC/HEIF files to JPEG format and extracts EXIF data from those JPEGs.
- **Keyword Search Module** uses keyword lists to identify files with specific words in them. You can select the keyword lists to search for automatically and you can create new lists using the "Advanced" button. Note that with keyword search, you can always conduct searches after ingest has finished. The keyword lists that you select during ingest will be searched for at periodic intervals and you will get the results in real-time. You do not need to wait for all files to be indexed before performing a keyword search, however you will only get results from files that have already been indexed when you perform your search.
- **Email Parser Module** identifies Thunderbird MBOX files and PST format files based on file signatures, extracting the e-mails from them, adding the results to the Blackboard.
- **Encryption Detection Module** looks for encrypted files.
- **Interesting Files Identifier Module** searches for files and directories based on user-specified rules in Tools, Options, Interesting Files. It works as a "File Alerting Module". It generates messages in the inbox when specified files are found.
- **Central Repository Module** adds file hashes and other extracted properties to a central repository for future correlation and to flag previously notable files.
- **PhotoRec Carver Module** carves files from unallocated space and sends them through the file processing chain.
- **Virtual Machine Extractor Module** extracts data from virtual machine files
- **Data Source Integrity Module** computes a checksum on E01 files and compares with the E01 file's internal checksum to ensure they match.
- **DJI Drone Analyzer** extracts data from drone files.
- **Plaso** uses Plaso to create **timeline** events.
- **Android Analyzer Module** allows you to parse common items from Android devices. Places artifacts into the BlackBoard.
- **GPX Analyzer** extracts geolocation data from .gpx files.
- **iOS Analyzer (iLEAPP)** extracts data from iOS data sources.

a.k.a. Exif parser

http://sleuthkit.org/autopsy/docs/user-docs/4.19.2/quick_start_guide.html

QUESTIONS?

REMINDER: WEEK 4 GRADED LAB TASKS #2 SATURDAY, 11 FEBRUARY 2023, 23:59 SGT USE THE GIVEN SAMPLE IMAGE FILE

1. For questions 1 & 2, use **SuspectDrive1.E01**
2. For question 3, use **SecretFile.docx**

Also, Week 3's graded task if you have not submitted.