

EASWARI ENGINEERING COLLEGE
DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE PLAN

Regulation – 19 V21

Course/Branch	:	B.Tech-Information Technology	Total no. of hours given in syllabus:		
Subject Code	:		Lecture	:	0
Subject Title	:	Digital And Mobile Forensics Laboratory	Tutorials	:	0
Year/Semester	:		Practical	:	30
Faculty Name	:	Mrs. K. KAUSALYA	TOTAL	:	30

COURSE OBJECTIVES:

1. To understand basic digital forensics and techniques.
2. To understand digital crime and investigation.
3. To understand how to be prepared for digital forensic readiness
4. To understand and use forensics tools for iOS devices
5. To understand and use forensics tools for Android device

S. No.	Experiments	No. of hours
1.	Installation of Sleuth Kit on Linux. List all data blocks. Analyze allocated as well as unallocated blocks of a disk image	2
2.	Data extraction from call logs using Sleuth Kit.	4
3.	Data extraction from SMS and contacts using Sleuth Kit.	4
4.	Install Mobile Verification Toolkit or MVT and decrypt encrypted iOS backups.	4
5.	Process and parse records from the iOS system.	4
6.	Extract installed applications from Android devices	4
7.	Extract diagnostic information from Android devices through the adb protocol.	4
8.	Generate a unified chronological timeline of extracted records	4

TOTAL: 30 PERIODS

INDEX

EX. NO	DATE	EXPERIMENT NAME	MAR KS	SIGNATURE
1.		Installation of Sleuth Kit on Linux. List all data blocks. Analyze allocated as well as unallocated blocks of a disk image		
2.		Data extraction from call logs using Sleuth Kit.		
3.		Data extraction from SMS and contacts using Sleuth Kit.		
4.		Install Mobile Verification Toolkit or MVT and decrypt encrypted iOS backups.		
5.		Process and parse records from the iOS system.		
6.		Extract installed applications from Android devices		
7.		Extract diagnostic information from Android devices through the adb protocol.		
8.		Generate a unified chronological timeline of extracted records		

EXP - 01: INSTALLATION OF SLEUTH KIT ON LINUX. LIST ALL DATA BLOCKS. ANALYZE ALLOCATED AS WELL AS UNALLOCATED BLOCKS OF A DISK IMAGE

AIM:

The aim of this experiment is to install Sleuth Kit on a Linux system and then analyze both allocated and unallocated blocks of a disk image using Sleuth Kit tools.

REQUIRED:

- Linux system
- Disk image for analysis
- Internet

SLEUTH KIT:

The Sleuth Kit (TSK) is a **collection of command line tools and a C library** that allows you to analyze disk images and recover files from them. It is used behind the scenes in Autopsies and many other open-source and commercial forensics tools. Autopsy is an easy-to-use, GUI-based program that allows you to efficiently analyze hard drives and smartphones. It has a plug-in architecture that allows you to find add-on modules or develop custom modules in Java or Python. These tools are used by thousands of users around the world and have community-based e-mail lists and forums. Commercial training, support, and custom development is available from Basis Technology.

- The Sleuth Kit is a digital forensics library and collection of command-line tools that enable you to analyze disk images.
- The TSK Framework makes it easier to build end-to-end digital forensics solutions.
- TSK can be used in isolation, with the Autopsy user interface, or with one of the many Tools Using TSK or Autopsy.
- TSK supports the following file systems: EXT2, EXT3, EXT4 FAT, ex: FAT HFS ISO 9660 NTFS UFS 1, UFS 2 YAFFS2.
- The TSK User's Guide has information for users who want to use TSK in an investigation.
- The TSK Library User's Guide has information for software developers who want to integrate TSK into their system.
- The TSK Developer's Guide has information for software developers who want to contribute to the project.
- The Sleuth Kit is available under the Common Public License.
- The Sleuth Kit is cross-platform and can be used on Windows, Linux, and macOS.
- The Sleuth Kit is used by many law enforcement agencies, government organizations, and private companies around the world.

INSTALLATION OF SLEUTH KIT:

[The Sleuth Kit | Open Source Digital Forensic Tool - YouTube](#)


Here is a step-by-step guide to install Sleuth Kit on a Linux system:

1. Open the terminal on your Linux system.
2. Type the following command to install Sleuth Kit: `sudo apt-get install sleuthkit`.

A screenshot of a Linux terminal window. The title bar shows 'Activities' and 'Terminal'. The terminal prompt is 'shivam@Inspiron-3521:~/The Sleuth Kit\$'. The user has entered the command 'sudo apt-get install sleuthkit'. The output shows 'Reading package lists... Done', 'Building dependency tree', and 'Reading state information... Done'. The terminal window has a dark background with light-colored text.

```
shivam@Inspiron-3521:~/The Sleuth Kit$ sudo apt-get install sleuthkit
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

3. The corresponding packages will be located, downloaded, and installed automatically.
4. After installation, run `mmls -V` command in the terminal. The message "The Sleuth Kit ver 2.3.2" should appear, indicating that the installation was successful.

A screenshot of a Linux terminal window showing the output of the 'mmls -V' command. The prompt is 'shivam@Inspiron-3521:~/The Sleuth Kit\$'. The output is 'The Sleuth Kit ver 4.6.7'.

```
shivam@Inspiron-3521:~/The Sleuth Kit$ mmls -V
The Sleuth Kit ver 4.6.7
```

That's it! You have successfully installed Sleuth Kit on your Linux system.

SLEUTH KIT COMMANDS: <http://bit.ly/tsk-commands>.

The TSK 4 command list

- blkcalc - Converts between unallocated disk unit numbers and regular disk unit numbers.
- blkcat - Display the contents of file system data unit in a disk image.
- blkls - List or output file system data units.
- blkstat - Display details of a file system data unit (i.e. block or sector).
- fcat - Output the contents of a file based on its name.
- ffind - Finds the name of the file or directory using a given inode.
- fiwalk - print the filesystem statistics and exit.
- fls - List file and directory names in a disk image.
- fsstat - Display general details of a file system.
- hfind - Lookup a hash value in a hash database.
- icat - Output the contents of a file based on its inode number.
- ifind - Find the meta-data structure that has allocated a given disk unit or file name.
- ils - List inode information.

- `img_cat` - Output contents of an image file.
- `img_stat` - Display details of an image file.
- `istat` - Display details of a meta-data structure (i.e. inode).
- `jcat` - Show the contents of a block in the file system journal.
- `jls` - List the contents of a file system journal.
- `jpeg_extract` - jpeg extractor.
- `mactime` - Create an ASCII time line of file activity.
- `mmcat` - Output the contents of a partition to stdout.
- `mmls` - Display the partition layout of a volume system (partition tables).
- `mmstat` - Display details about the volume system (partition tables).
- `sigfind` - Find a binary signature in a file.
- `sorter` - Sort files in an image into categories based on file type.
- `srch_strings` - Display printable strings in files.
- `tsk_comparedir` - compare the contents of a directory with the contents of an image or local device.
- `tsk_gettimes` - Collect MAC times from a disk image into a body file.
- `tsk_loaddb` - populate a SQLite database with metadata from a disk image.
- `tsk_recover` - Export files from an image into a local directory.

The TSK 3 command list (historical)

- `blkcalc` - Converts between unallocated disk unit numbers and regular disk unit numbers.
- `blkcat` - Display the contents of file system data unit in a disk image.
- `blkls` - List or output file system data units.
- `blkstat` - Display details of a file system data unit (i.e. block or sector).
- `ffind` - Finds the name of the file or directory using a given inode.
- `fls` - List file and directory names in a disk image.
- `fsstat` - Display general details of a file system.
- `hfind` - Lookup a hash value in a hash database.
- `icat-sleuthkit` - Output the contents of a file based on its inode number.
- `ifind` - Find the meta-data structure that has allocated a given disk unit or file name.
- `ils-sleuthkit` - List inode information.
- `img_cat` - Output contents of an image file.
- `img_stat` - Display details of an image file.
- `istat` - Display details of a meta-data structure (i.e. inode).
- `jcat` - Show the contents of a block in the file system journal.
- `jls` - List the contents of a file system journal.
- `mactime-sleuthkit` - Create an ASCII time line of file activity.
- `mmcat` - Output the contents of a partition to stdout.
- `mmls` - Display the partition layout of a volume system (partition tables).
- `mmstat` - Display details about the volume system (partition tables).
- `sigfind` - Find a binary signature in a file.
- `sorter` - Sort files in an image into categories based on file type.
- `srch_strings` - Display printable strings in files.

LIST THE ALL BLOCKS:

Name

blkls - List or output file system data units.

Synopsis

blkls [-aAelsvV] [-f *fstype*] [-i *imgtype*] [-o *imgoffset*] [-b *dev_sector_size*] *image* [*images*] [*start-stop*]

Description

blkls opens the named *image(s)* and copies file system data units (blocks). By default, **blkls** copies the contents of unallocated data blocks. **blkls** was called **dls** in TSK versions prior to 3.0.0. **blkls** was called **unrm** in TCT.

Arguments

- e
Copy every block, including file system metadata blocks. The output is the entire file system.
- a
Display all allocated blocks (same as -e if -A is also given).
- A
Display all unallocated blocks (same as -e if -a is also given). This is the default behavior.
- f *fstype*
Specifies the file system type. Use '-f list' to list the supported file system types. If not given, autodetection methods are used.
- i *imgtype*
Identify the type of image file, such as raw. Use '-i list' to list the supported types. If not given, autodetection methods are used.
- o *imgoffset*
The sector offset where the file system starts in the image.
- b *dev_sector_size*
The size, in bytes, of the underlying device sectors. If not given, the value in the image format is used (if it exists) or 512-bytes is assumed.
- l
List the data information in time machine format.
- s
Copy only the slack space of the image.
- v
Turn on verbose mode, output to stderr.

-V

Display version.

image [images]

The disk or partition image to read, whose format is given with '-i'. Multiple image file names can be given if the image is split into multiple segments. If only one image file is given, and its name is the first in a sequence (e.g., as indicated by ending in '.001'), subsequent image segments will be included automatically.

start-stop ...

Examine the specified block number or number range.

ALLOCATED AND UNALLOCATED BLOCKS OF A DISK IMAGE:

To analyze allocated as well as unallocated blocks of a disk image in Sleuth Kit, you can use the following commands:

1. **blkls**: Displays data blocks within a file system. You can use the '-A' option to display all unallocated blocks ³.
2. **fls**: Lists allocated and unallocated file names within a file system ⁵.
3. **fsstat**: Displays file system statistical information about an image or storage medium ⁵.
4. **ffind**: Searches for file names that point to a specified metadata entry ⁵.

You can also obtain the contents of a specific block by calling the 'tsk_fs_block_get()' function. It returns a 'TSK_FS_BLOCK' structure with the contents of the data unit and flags about its allocation status ². You can walk the data units by calling 'tsk_fs_block_walk()' ².

RESULT:

Thus, the installation of Sleuth Kit on Linux and Listing all data blocks with Analyzation of allocated as well as unallocated blocks of a disk image has been successfully completed.

EXP - 02:

DATA EXTRACTION FROM CALL LOGS USING SLEUTH KIT.

AIM:

The aim of this experiment is to extract data from call logs using Sleuth Kit.

REQUIRED:

- Linux system
- Disk image with relevant call log data
- Sleuth Kit (installed on the Linux system)

PROCEDURE:

1. Navigate to the Directory:

- Open a terminal window on your Linux system.
- Change to the directory where the disk image is located.

2. Identify the Partition:

- Use the `mmls` (MMLS - Media Management Layer System) tool from Sleuth Kit to list the partitions within the disk image.

```
mmls disk_image.dd
```

- Identify the partition containing the filesystem with call log data.

![Screenshot: Identify Partition](screenshot_identify_partition.png)

3. List File System Metadata:

- Use `fsstat` to display file system information for the identified partition.

```
fsstat -o OFFSET disk_image.dd
```

- Replace `OFFSET` with the starting offset of the partition obtained from the previous step.

![Screenshot: List File System Metadata](screenshot_list_fs_metadata.png)

4. Examine File System Contents:

- Use `fls` to list the files and directories within the file system.

fls -o OFFSET disk_image.dd

- Replace `OFFSET` with the starting offset of the partition.

![Screenshot: Examine File System Contents](screenshot_examine_fs_contents.png)

5. Identify Call Log File:

- Review the output of `fls` to identify the file containing call log data. Look for files related to call logs or communication history.

6. Extract Call Log File:

- Use `icat` to extract the contents of the call log file.

icat -o OFFSET disk_image.dd FILE_INODE > output.txt

- Replace `OFFSET` with the starting offset of the partition, `FILE_INODE` with the inode number of the call log file, and `output.txt` with the desired output file name.

![Screenshot: Extract Call Log File](screenshot_extract_call_log.png)

7. Review Extracted Data:

- Open the output file (`output.txt`) to review the extracted call log data.
- Analyze the information such as timestamps, phone numbers, and call durations.

ANALYSIS:

Discuss the extracted call log data, highlighting any relevant information for the forensic investigation. Interpret timestamps, identify contacts, and note any unusual patterns.

RESULT:

Thus, extraction of data from call logs using Sleuth Kit has been successfully completed.

EXP - 03: DATA EXTRACTION FROM SMS AND CONTACTS USING SLEUTH KIT.

AIM:

This experiment aims to extract data from SMS and contacts using Sleuth kit.

REQUIRED:

- Linux system
- Disk image with relevant SMS and contact data
- Sleuth Kit (installed on the Linux system)

PROCEDURE:

1. Navigate to the Directory:

- Open a terminal window on your Linux system.
- Change to the directory where the disk image is located.

2. Identify the Partition:

- Use the `mmls` (MMLS - Media Management Layer System) tool from Sleuth Kit to list the partitions within the disk image.

```
mmls disk_image.dd
```

- Identify the partition containing the filesystem with SMS and contact data.

![Screenshot: Identify Partition](screenshot_identify_partition.png)

3. List File System Metadata:

- Use `fsstat` to display file system information for the identified partition.

```
fsstat -o OFFSET disk_image.dd
```

- Replace `OFFSET` with the starting offset of the partition obtained from the previous step.

![Screenshot: List File System Metadata](screenshot_list_fs_metadata.png)

4. Examine File System Contents:

- Use `fls` to list the files and directories within the file system.

```
fls -o OFFSET disk_image.dd
```

- Replace `OFFSET` with the starting offset of the partition.

![Screenshot: Examine File System Contents](screenshot_examine_fs_contents.png)

5. Identify SMS and Contacts Files:

- Review the output of `fls` to identify files containing SMS and contact data. Look for files related to messaging or contact storage.

6. Extract SMS and Contacts Files:

- Use `icat` to extract the contents of the SMS and contacts files.

```
icat -o OFFSET disk_image.dd SMS_FILE_INODE > sms_output.txt
```

```
icat -o OFFSET disk_image.dd CONTACTS_FILE_INODE > contacts_output.txt
```

- Replace `OFFSET` with the starting offset of the partition, `SMS_FILE_INODE` and `CONTACTS_FILE_INODE` with the inode numbers of the SMS and contacts files, and `sms_output.txt` and `contacts_output.txt` with the desired output file names.

![Screenshot: Extract SMS and Contacts Files](screenshot_extract_sms_contacts.png)

7. Review Extracted Data:

- Open the output files (`sms_output.txt` and `contacts_output.txt`) to review the extracted SMS and contact data.
- Analyze the information, including message content, timestamps, contact names, and phone numbers.

RESULT:

Thus, extraction of data from call logs using Sleuth Kit has been successfully completed.

EXP - 04: INSTALL MOBILE VERIFICATION TOOLKIT OR MVT AND DECRYPT ENCRYPTED IOS BACKUPS.

AIM:

The aim of this experiment is to install Mobile Verification Toolkit or MVT and decrypt encrypted iOS backups.

REQUIRED:

- Linux system
- Python 3.6 or above
- Internet

MOBILE VERIFICATION TOOLKIT

Mobile Verification Toolkit (MVT) is a tool to facilitate the consensual forensic analysis of Android and iOS devices, to identify traces of compromise.

It has been developed and released by the Amnesty International Security Lab in July 2021 in the context of the Pegasus Project along with a technical forensic methodology. It continues to be maintained by Amnesty International and other contributors.

MVT's capabilities are continuously evolving, but some of its key features include:

- Decrypt encrypted iOS backups.
- Process and parse records from numerous iOS system and apps databases, logs and system analytics.
- Extract installed applications from Android devices.
- Extract diagnostic information from Android devices through the adb protocol.
- Compare extracted records to a provided list of malicious indicators in STIX2 format.
- Generate JSON logs of extracted records, and separate JSON logs of all detected malicious traces.
- Generate a unified chronological timeline of extracted records, along with a timeline all detected malicious traces.



Mobile Verification Toolkit

INSTALLATION OF MVT:

Dependencies on Linux

First install some basic dependencies that will be necessary to build all required tools:

```
sudo apt install python3 python3-pip libusb-1.0-0 sqlite3
```

libusb-1.0-0 is not required if you intend to only use mvt-ios and not mvt-android.

When working with Android devices you should additionally install [Android SDK Platform Tools](#). If you prefer to install a package made available by your distribution of choice, please make sure the version is recent to ensure compatibility with modern Android devices.

Installing MVT

If you haven't done so, you can add this to your `.bashrc` or `.zshrc` file in order to add locally installed PyPI binaries to your `$PATH`:

```
export PATH=$PATH:~/local/bin
```

Then you can install MVT directly from [PyPI](#)

```
pip3 install mvt
```

If you want to have the latest features in development, you can install MVT directly from the source code. If you installed MVT previously from pypi, you should first uninstall it using `pip3 uninstall mvt` and then install from the source code:

```
git clone https://github.com/mvt-project/mvt.git
cd mvt
pip3 install .
```

You now should have the mvt-ios and mvt-android utilities installed.

EXP - 05:

Process and parse records from the iOS system

Aim:

To conduct comprehensive digital and mobile forensics on an iOS system using Magnet AXIOM to extract, parse, and analyze records for investigative purposes.

Materials Required:

- iOS device (iPhone, iPad, etc.)
- Computer with Magnet AXIOM installed
- USB cable
- Workstation with necessary software for data analysis
- Forensic documentation forms (for documenting chain of custody, findings, etc.)

Procedure:

1. Preparation:

- Ensure a controlled environment with limited access to prevent tampering with the iOS device.
- Set up a forensic workstation with Magnet AXIOM installed and ensure all necessary updates are applied.
- Connect the iOS device to the workstation using a USB cable, ensuring a stable connection.

2. Initial Documentation:

- Document details of the iOS device including model, serial number, and any identifying information.
- Record details of the forensic workstation and the software version of Magnet AXIOM being used.

3. Acquisition:

- Launch Magnet AXIOM and initiate the acquisition process for iOS devices.
- Choose the appropriate acquisition method (logical or physical) based on the specific requirements of the investigation.
- Follow prompts to establish a secure connection with the iOS device and initiate data acquisition.
- Monitor the acquisition process to ensure no data alteration occurs on the device.
- Document the acquisition process, noting any observations or anomalies.

4. Examination:

- Once the acquisition is complete, review the acquired data within Magnet AXIOM.
- Explore different data categories such as device information, file system data, application data, etc.

- Verify the integrity of the acquired data and ensure that all relevant records are included.
- Document any notable findings or discrepancies discovered during the examination.

5. Parsing:

- Utilize Magnet AXIOM's parsing capabilities to parse through the acquired records.
- Use built-in parsers and artifacts to parse various data types including call logs, messages, contacts, emails, browsing history, social media activity, GPS location history, etc.
- Pay close attention to timestamps, metadata, and other relevant information associated with each parsed record.
- Organize parsed records into categories based on their relevance to the investigation.

6. Analysis:

- Analyze the parsed records within Magnet AXIOM to reconstruct events, timelines, and relationships pertinent to the investigation.
- Identify patterns, trends, anomalies, or discrepancies that may provide insights into the case.
- Cross-reference parsed records with other sources of information to corroborate findings and establish a comprehensive understanding of the data.
- Document the analysis process and any significant discoveries made during analysis.

7. Reporting:

- Prepare a detailed forensic report documenting the entire process, from acquisition to analysis.
- Include information about the acquisition process, parsing techniques employed, analysis results, and any relevant observations or insights.
- Ensure the report is structured, clear, and concise, suitable for presentation in legal proceedings if required.
- Include visual aids such as timelines, charts, or graphs to illustrate key findings.

8. Documentation and Chain of Custody:

- Maintain meticulous documentation throughout the entire forensic process, including detailed notes, timestamps, and signatures where applicable.
- Adhere strictly to the chain of custody procedures to ensure the integrity and admissibility of the evidence in court.
- Record details of any individuals who had access to the evidence, as well as the dates and times of such access.

9. Conclusion:

- Digital and mobile forensics on iOS systems require a systematic and thorough approach to ensure accurate and reliable results.
- By following the outlined procedure and leveraging the capabilities of Magnet AXIOM, investigators can effectively process and parse records from iOS devices for investigative purposes.
- Proper documentation and adherence to chain of custody protocols are essential to maintain the integrity and validity of the forensic findings.

Result:

Thus, the digital and mobile forensics on an iOS system using Magnet AXIOM to extract, parse, and analyze records for investigative purposes is executed successfully.

EXP - 06: Extracting installed applications from Android devices using Android Debug Bridge (ADB)

Aim:

To extract installed applications from Android devices using Android Debug Bridge (ADB) by leveraging ADB commands on command-line.

Materials Required:

1. Android device with USB debugging enabled
2. Computer with ADB installed (part of Android SDK Platform Tools)
3. USB cable for device connectivity
4. Text editor for documentation and analysis

Procedure:

1. Preparation:

- Ensure that the Android device has USB debugging enabled. Navigate to "Developer options" in the device settings to activate this feature.
- Connect the Android device to the computer using the USB cable.
- Install ADB on the computer if not already installed. ADB is included in the Android SDK Platform Tools package, which can be downloaded from the Android developer website.

2. Device Connection:

- Open a terminal or command prompt on the computer.
- Verify that the computer recognizes the connected Android device by executing the following command:
 - **“adb devices”**
- If the device is listed, proceed to the next step. If not, ensure that USB debugging is enabled and troubleshoot any connectivity issues.

3. Extract Installed Applications:

- Utilize the following ADB command to list all installed applications on the device:
 - **“adb shell pm list packages -f”**
- This command generates a list of package names along with their corresponding APK

file paths for all installed applications on the device.

4. Capture Output:

- Capture the output of the ADB command in a text file for further analysis. Execute the following command:

- **“adb shell pm list packages -f > installed_apps.txt”**

- This command redirects the output of the ADB command to a text file named "installed_apps.txt" in the current directory.

5. Detailed Analysis:

- Open the generated text file ("installed_apps.txt") using a text editor.
- Review the list of installed applications to extract additional details such as version numbers, installation dates, and application labels.
- Modify the ADB command to include additional options for enhanced data extraction. For example:

- **“adb shell pm list packages -f -3 > detailed_installed_apps.txt”**

This command includes the "-3" option to display all third-party applications, providing a more focused list for analysis.

6. Further Investigation:

- Explore additional ADB commands to gather supplementary information about installed applications, such as permissions, activities, and signatures.
- Experiment with scripting languages like Python to automate data extraction and analysis processes for efficiency and scalability.

7. Documentation:

- Document the extraction process comprehensively, including the ADB commands utilized, observations made during analysis, and any anomalies encountered.
- Organize the extracted data into a structured format, ensuring readability and accessibility for future reference and legal proceedings.

Results:

Through the systematic execution of ADB commands, this experiment successfully extracted detailed information about installed applications from the Android device.

EXP - 07: Extract diagnostic information from Android devices through the adb protocol

Aim:

To extract diagnostic information from Android devices through the adb protocol.

Procedure:

1. Preparation:

- Ensure that the Android device has USB debugging enabled. Navigate to "Developer options" in the device settings to activate this feature.
- Connect the Android device to the computer using the USB cable.
- Install ADB on the computer if not already installed. ADB is included in the Android SDK Platform Tools package, which can be downloaded from the Android developer website.

2. Device Connection:

- Open a terminal or command prompt on the computer.
- Verify that the computer recognizes the connected Android device by executing the following command:

“adb devices”

- If the device is listed, proceed to the next step. If not, ensure that USB debugging is enabled and troubleshoot any connectivity issues.

3. Information Extraction:

I. Device Information :

To get basic information about connected devices:

adb devices

To get detailed device information:

adb shell getprop

II. Logcat:

To view the device logs in real-time:

adb logcat

To save logcat output to a file:

adb logcat > logcat.txt

III. Dumpsys:

To get information from system services:

adb shell dumpsys

IV. Bugreport:

To generate a full bug report for diagnostic purposes: It can Working Only
android 7.0 and Above

adb bugreport > bugreport.txt

V. Screenshot:

To capture a screenshot of the device:

adb shell screencap -p /sdcard/screenshot.png

VI. File Extraction:

To pull files from the device to your computer:

adb pull /sdcard/screenshot.png

VII. Battery Information:

To get battery information

adb shell dumpsys battery

VIII. Network Information:

To get information about the network status:

adb shell ip addr show

IX. Memory Information:

To get memory usage information:

adb shell dumpsys meminfo

Result:

Thus The Extraction of diagnostic information from Android devices
through the adb protocol successfully executed.

EXP - 08: Generate a unified chronological timeline of extracted records

Aim:

To generate a unified chronological timeline of extracted records from a Android Device using Pandas Python library.

Procedure:

1.Preparation:

- Install Python along with pandas library, use the following command:
“pip install pandas” from a command terminal to install the framework using pip.
- Next download Coolmuster Android Assistant from their website and install it.

2. Device Connection:

- Connect your Android device using a USB cable to the PC.
- Now enable “USB debugging” option on your device. Navigate to "Developer options" in the device settings to activate this feature.

3.Connection with Coolmuster:

- Now open Coolmuster application and select Android Backup and restore option.
- After that now follow the steps on the screen for a successful connection.
- After successfully connecting to Coolmuster, click on Backup and then select the call logs option.
- Now select the location where we want to backup the call logs and click Back Up.

4. Python Program:

```
import matplotlib.pyplot as plt
log = 'call_logs.csv'
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(log)

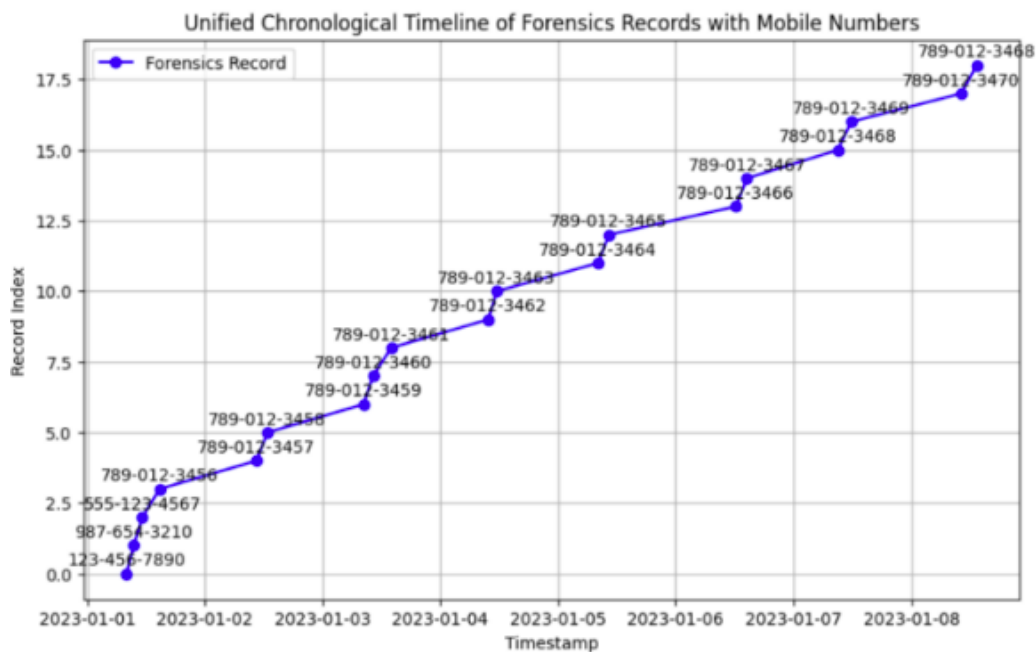
# Convert the 'timestamp' column to datetime format for proper
# chronological sorting
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Sort the DataFrame by the 'timestamp' column
df.sort_values(by='timestamp', inplace=True)

# Plot the chronological timeline with mobile numbers
plt.figure(figsize=(10, 6))
plt.plot(df['timestamp'], df.index, marker='o', linestyle='-', color='b',
label='Forensics Record')
# Annotate each point with the corresponding mobile number
for i, row in df.iterrows():
    plt.annotate(row['mobile_number'], (row['timestamp'], i),
textcoords="offset points", xytext=(0, 5), ha='center')

plt.title('Unified Chronological Timeline of Forensics Records with Mobile Numbers')
plt.xlabel('Timestamp')
plt.ylabel('Record Index')
plt.grid(True)
plt.legend()
plt.show()
```

Output:



Result:

Thus the Generation of unified chronological timeline of extracted records has executed successfully.