

Forensic Tool for Android Mobile Device

Kush Dabade¹, Parijat Dhalkar², Siddhesh Bandgar³, Anshul Shende⁴

^{1,2,3,4}B. Tech Student, Dept. of Computer Engineering and IT, VJTI College, Mumbai, Maharashtra, India.

Abstract - Mobile devices are the major platform for users to transfer and exchange diverse data for communication. Not just limited to communication, the rich set of diverse mobile applications have made mobile a widespread device all over the world. Market research reports from Forrester estimate the global mobile penetration to be around 50% in 2017 and is forecast to reach 66% by 2022. In India, Mobile penetration using smartphones has reached around 50% as of 2022 from 0.1% in 1998. The usability of mobile applications spreads across domains like banking, personal digital assistant, remote working, e-commerce, internet access, entertainment and medical usage. Due to the increasing usage of mobile devices, numerous mobile security issues and data privacy threats are challenging both manufacturers and users. Mobile devices are an ideal target for various security issues and data privacy threats in a mobile ecosystem. Also the need for forensic extraction of data from a particular mobile device is an important aspect for the evidence. The forensic analysis helps extract, analyze the data stored on a mobile device along with its metadata, path locations, databases etc. that are not visible to the mobile user otherwise. In this paper, we have presented a detailed study on mobile hardware/software architecture, Android mobile Operating System, its vulnerabilities and security model and forensic analysis of Android mobile devices. Finally, a forensic tool is developed to extract the data from an Android mobile device, classify extracted data as safe or malicious and present it in well-formatted reports.

Key Words: Android, forensics, data extraction, mobile, malicious, report

1.INTRODUCTION

Mobile phones have witnessed a remarkable growth in the past few years owing to convenience factors like portability, ubiquity, high performance and low power consumption of mobile processors and storage chips, high-resolution touchscreen, high-speed wireless networks and the convenient availability of diverse apps. Among the leading mobile OS platforms, Android is on the majority of smartphones in most countries in the world with a share of 82.8% with significant increase during the past two decades. Rich-content mobile applications further penetrate daily life beyond the basic communication use, which makes usage patterns shift from desktop to mobile devices at a vigorous pace making mobile devices and applications become an indispensable part of our daily life.

Thus, the awareness of the importance of privacy protection, data usage and details about the resident data on mobile

devices has driven the demand for Mobile security and Forensics. Although each smartphone OS developer has supported specific security measures, malicious codes have made their way to the mobile devices. The resident data has been the main target for these attacks and it is important to know and understand the domain of data protection along with data extraction and validation for investigation purposes.

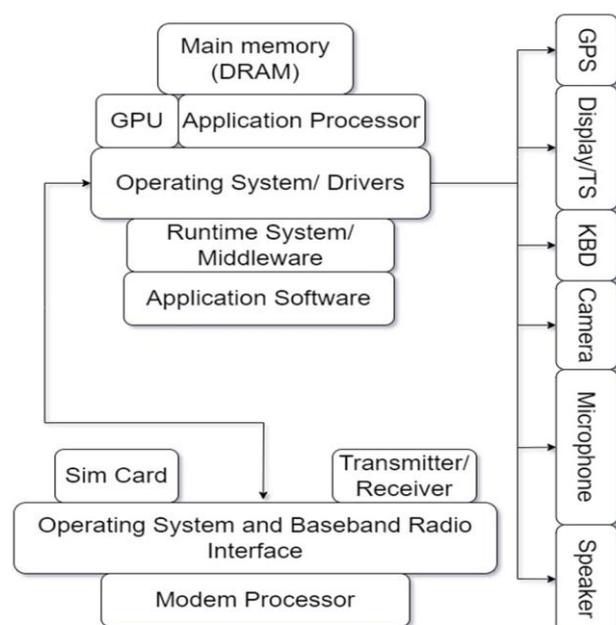
The science behind recovering digital evidence from mobile phones is called mobile forensics[1]. Digital evidence is defined as information and data that is stored on, received, or transmitted by an electronic device that is used for investigations.

This paper briefs about topics like mobile architecture, Android operating system, its vulnerabilities, attacks, security model and forensics. Lastly, a forensic tool has been implemented to extract the data from an Android mobile device with the help of a dump file. The tool further proceeds to classify extracted data as safe or malicious and present it in well-formatted reports.

2.LITERATURE REVIEW

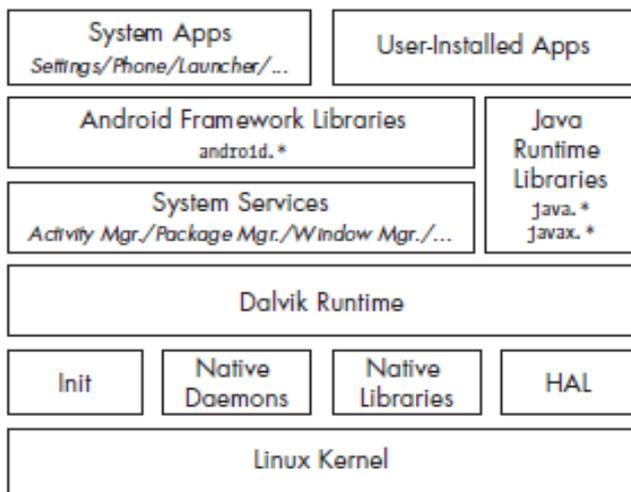
2.1 Mobile Architecture

- **Mobile Hardware Architecture:**



The main use case of a smartphone is to execute application software. The software gets executed with the help of an application processor often taking help from a graphics processor for rendering scenes and whenever the application processor of the graphics processor needs data it will access the main memory which is dynamic random-access memory. Between the application software and the application processor sit two elements one is the drivers that are necessary for handling various types of I/O calls and a runtime system or the middle layer which the application software takes help off to execute various types of application programs[2]. The I/O devices present are for example the Global Positioning System, Display and the Touchscreen, Keyboard, Camera, Microphone, Speaker and so on. The communication is handled by a modem processor which receives all the signals and passes them on to the operating system and the baseband radio interface. The radio interface communicates with the transmitter and receiver and also the SIM card. The two different operating systems residing on the application processor and the modern processor usually communicate with each other for handling data communication between these two modules.

• **Software Architecture**



Android architecture contains a number of components to support any android device needs. Android software contains an open-source Linux Kernel having a collection of C/C++ libraries which are exposed through an application framework. Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provides a platform for running an android application.

The main components of android architecture are following:-

1. Applications
2. Application Framework
3. Android Runtime

4. Platform Libraries
5. Linux Kernel

2.2 Study of Android Attacks and Security model

• **Android File System**

There are six primary logical partitions under the Android file system[3].

/boot partition consists of the Android kernel and the ramdisk.

/system partition accommodates the entire Android OS. This includes the Android GUI as well as the system apps that come pre-installed on Android devices.

/recovery partition is designed for backup and can be considered the alternative boot option or partition in an Android device.

/data partition consists of all of the user’s data, including contacts, settings, apps and messages.

/cache partition stores the frequently accessed app data and components. Clearing cache also frees up some space in your device and can also fix certain issues at times.

/misc partition contains all the miscellaneous system settings like on/off switches, carrier or region ID, USB configuration, and certain hardware settings.

• **Vulnerabilities, attacks on Android**

1. Attacks on Android architectural layers

The attacks at the kernel layer mainly target root privilege, sandboxing, memory, bootloader, device drivers etc. On the other hand, middleware layer attacks target Android architectural components like Hardware Abstraction Layer, libraries and native components. The application layer attacks occur during runtime of malicious applications.

2. Malware

Malware is the malicious software aimed at private specific information which disturbs users, may cause breakdown of the device and lead to results such as causing information and documents belonging to the user to be stolen or become unusable[4]. The different types of malwares include virus, worm, spyware, trojan horse, logic bomb, ransomware, backdoor, rootkit etc. A malware can steal mobile data, record calls, capture images, monitor location or even exfiltrate device information.

3. WiFi based Attacks

Wi-Fi connection is one of the security threats which can exploit the vulnerabilities in the Android operating system. The attacker can eavesdrop and access the content

of Android without the user permission. Intercepting traffic lets attackers read information that was previously assumed to be safely encrypted.

4. Bluetooth based Attacks

The vulnerability due to bluetooth connection allows remote attackers to run their own malicious code on vulnerable devices via Bluetooth LMP packets. The attackers can also use truncated, oversized, or out-of-order Bluetooth LMP packets to crash devices altogether. Bluejacking and bluebugging are two of the common examples of bluetooth based attacks.

- **Android OS security model**

Android security model consists of the following components[5]:

1. Sandboxing

Application sandbox is a security mechanism through which individual android applications run in their own "space" and cannot interact with other installed apps or the Android OS without proper permissions. The applications are isolated, or sandboxed, both at the process level and at the file level. The sandboxing is done at the kernel level to ensure that each application runs in an isolated environment.

2. Permission Mechanism

Permissions are access rights that can control access to hardware devices, internet connectivity, data, or OS services. The security sensitive interfaces are protected by Android permission such as PHONE_CALLS, INTERNET, and SEND_SMS. It means that the application must have the permission to perform the above tasks. Permissions support protection levels namely Normal, Dangerous, Signed, System and Signed.

3. Code Signing and Platform keys

These security features are implemented by comparing the signing certificate of the currently installed target app with the certificate of the update or related application. System applications are signed by a number of platform keys. Different system components can share resources and run inside the same process when they are signed with the same platform key.

4. Components Encapsulation

The components are declared as private as well as public. Within the same application, private components are accessible for each other. The public components are accessible by other applications which are not in the same sandboxing, having full accessible permission, but also restricted through with customized permission.

5. Multi user support

The composite structure, the target physical user's user ID and the app ID as effective UID guarantees multiple application instances installed by multiple users get their own sandbox. Additionally, Android also guarantees dedicated shared storage to each physical user.

6. SELinux

Security Enhanced Linux (SELinux) is a MAC implementation for the Linux kernel. Mandatory access control (MAC) ensures that access to resources conforms to a system-wide set of authorization rules called a policy that can only be changed by an administrator while users cannot override or bypass it.

7. Verified Boot

Android supports verified boot using the verity target of Linux's Device-Mapper. Verity provides transparent integrity checking of block devices using a cryptographic hash tree.

2.3 Android Forensic Analysis

Mobile forensics is a branch of digital forensics that is evolving in today's digital era and is constantly changing as new phones are released and operating systems are updated[10]. Android forensics deals with extracting, recovering, and analyzing data present on an Android device through various techniques. Due to the open nature of the Android operating system, these forensic techniques and methods can apply to more than just mobile phones: refrigerators, vehicle entertainment units, televisions, watches, and many more devices run Android. It's important to have a clear understanding of the platform and other fundamentals before we dive in and find out how to extract data.

Following are the necessary steps that need to be followed while performing android forensic analysis[6]:

1. Investigation preparation
2. Seizure and isolation
3. The acquisition phase
4. Examination and analysis
5. Reporting

Few android platform tools used in the process of forensic analysis are as follows:

1. Android Software Development Kit

The Android software development kit is the development resource needed to develop Android applications. It includes software libraries and APIs, reference materials, an emulator, and other tools.

2. Android Virtual Devices (Emulator)

Android Virtual Device is a virtual mobile device, or emulator, which runs on your computer. The emulator is especially helpful for developers for creating custom applications. The emulator takes considerable resources.

3. Android Debug Bridge

Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device.[7] The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device. It is a client-server program that includes three components: client, daemon and a server.

- **Forensics using ADB**

1. Phone Information

```

Command Prompt - adb shell
generic_x86:/ #
-e "hardware" -e "platform" -e "revision" -e "serialno" -e "product.name" -e "brand"
<
[ro.board.platform]: []
[ro.boot.android_dt_dir]: [/sys/bus/platform/devices/ANDR0001:00/properties/android/]
[ro.boot.hardware]: [ranchu]
[ro.boot.serialno]: [EMULATOR31X2X80]
[ro.build.version.sdk]: [25]
[ro.hardware]: [ranchu]
[ro.hardware.audio.primary]: [goldfish]
[ro.kernel.androidboot.hardware]: [ranchu]
[ro.kernel.androidboot.serialno]: [EMULATOR31X2X80]
[ro.product.brand]: [google]
[ro.product.manufacturer]: [Google]
[ro.product.model]: [Android SDK built for x86]
[ro.product.name]: [sdk_google_phone_x86]
[ro.revision]: [0]
[ro.serialno]: [EMULATOR31X2X80]
generic_x86:/ #
    
```

2. Messages

```

Command Prompt - adb shell
C:\Users\kushd>adb shell
generic_x86:/ #
generic_x86:/ # sqlite3 /data/user_de/0/com.android.providers.telephony/databases/mmsms.db
SQLite version 3.9.2 2015-11-02 18:31:45
Enter ".help" for usage hints.
sqlite> .tables
addr                pdu_restricted      threads
android_metadata   pending_msgs        words
attachments        rate                 words_content
canonical_addresses raw                   words_segdir
dms                  sms                  words_segments
pprt                 sms_restricted
pdu                  sp_pending
sqlite> select * from sms;
1|3|(959) 468-7497|1647756530933|0|1|-1|2||Hello|0|1|0|com.google.android.apps.messaging|1
2|4|(913) 790-0888||1648132654717|0|1|-1|2||Hello||0|1|0|com.google.android.apps.messaging|1
3|4|(913) 790-0888||1648132654717|0|1|-1|2||Kush Here||0|1|0|com.google.android.apps.messaging|1
sqlite>
    
```

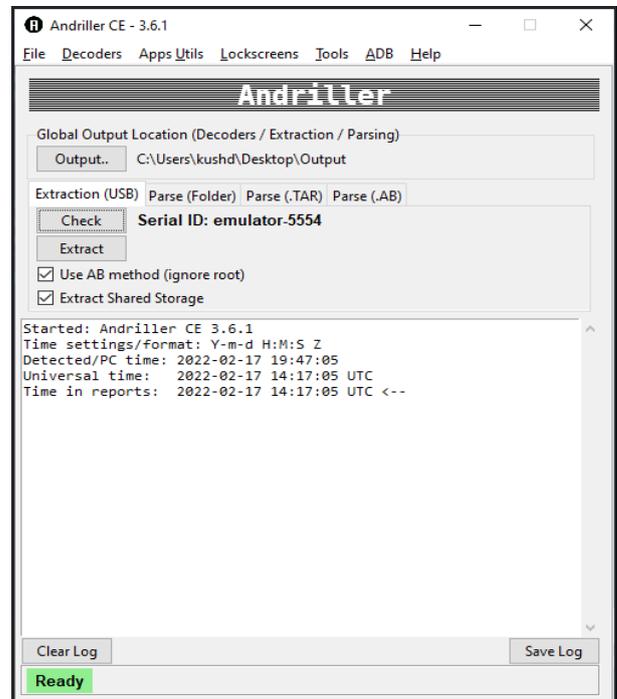
3. Call Logs

```

Command Prompt - adb shell
C:\Users\kushd>adb shell
generic_x86:/ #
generic_x86:/ # sqlite3 /data/data/com.android.providers.contacts/databases/calllog.db
SQLite version 3.9.2 2015-11-02 18:31:45
Enter ".help" for usage hints.
sqlite> .tables
android_metadata    calls                properties           voicemail_status
sqlite> select * from calls;
1|904087497|1|1647756492885|1|2|0|com.android.phone/com.android.services.telephony.TelephonyConnectionService|890141931
118510720|1555515554|0|-1|Kush D|1|US||content://com.android.contacts/contacts/lookup/484416e396e49f7
7435_3789e3-305140372f/3|(959) 468-7497|19594687497|0|(959) 468-7497|1647841909853|1|1|0|0
2|3|7900888|1|1648132654717|0|2|0|com.android.phone/com.android.services.telephony.TelephonyConnectionService|890141931
118510720|1555515554|0|-1|My New Number|2|US||Kansas|content://com.android.contacts/contacts/lookup/40444-41594331
55435141203148|4|(913) 790-0888|1917900888|0|(913) 790-0888|1648132504010|1|1|0|0
1913790088|1|1648132504010|1|2|0|com.android.phone/com.android.services.telephony.TelephonyConnectionService|890141931
118510720|1555515554|0|-1|My New Number|2|US||Kansas|content://com.android.contacts/contacts/lookup/40444-41594331
55435141203148|4|(913) 790-0888|1917900888|0|(913) 790-0888|1648132504010|1|1|0|0
4|123698547|1|164813250855|0|2|0|com.android.phone/com.android.services.telephony.TelephonyConnectionService|890141931
118510720|1555515554|0|-1|1|1|US||content://com.android.contacts/contacts/lookup/encoded/directory-922337203685477580
7f|display_name="123698547",display_name_source="3",vnd.android.cursor.item/vcontact":{"vnd.android.cursor.item/vphone_vc":{"data1":"123698547","data2":"83"}}|0|123698547|1|1648132521883|1|1|1|0|0
sqlite>
    
```

- **Forensics using Andriller**

Andriller Tool GUI



1. Phone Information

Andriller Report

Type	Data
Serial	emulator-5554
Status	device
Permission	root
Ro.Buidl.Version.Release	10
Ro.Buidl.Display.Id	sdk_gphone_x86-userdebug 10 QSR1.191030.002 5978551 dev-keys
WiFi.Mac	02:15:b2:00:00:00
Local.Time	2022-02-17 19:48:36 India Standard Time
Device.Time	2022-02-17 19:48:36 IST
Accounts	com.google.vulnerableapp@gmail.com
Application	Shared Storage (7)

2. Contacts

[Contacts]

#	Name	Number	Email	Other
1	Kush D	(959) 468-7497		group_membership: 1
3	Kush D	9594687497		vnd.com.whatsapp.profile: 919594687497@s.whatsapp.net vnd.com.whatsapp.voip.call: 919594687497@s.whatsapp.net vnd.com.whatsapp.video.call: 919594687497@s.whatsapp.net
4	My New Number	(913) 790-0888		group_membership: 1

3. Messages

[SMS Messages]

#	Number	Message	Type	Time
3	(913) 790-0888	Kush Here	Sent	2022-03-24 20:07:34
2	(913) 790-0888	Hello	Sent	2022-03-24 20:07:17
1	(959) 468-7497	Hello	Sent	2022-03-20 11:38:50

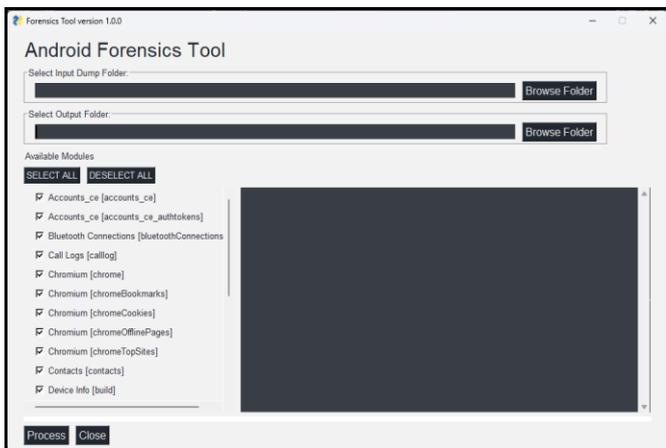
3. PROPOSED SYSTEM

3.1 Problem Statement

The smartphone market is growing higher and higher. With the drastic changes in technology and loads of extremely valuable data, data resident on mobile devices are becoming hotspots of target. The usability offered by a variety of mobile applications is adding sensitive data. The analysis of mobile phones and data extraction could reveal SMS, contacts, installed applications, GPS data, emails, deleted data, etc. that can offer significant insights. Android, being the leading mobile operating system, the design of an Android Forensic analysis tool is the focus of this paper with a brief look on topics discussed in Literature review.

The Android Forensic analysis tool should provide the resident data along with the details of data like databases, tables, timestamps, related attributes etc. Further the tool must classify data as safe or malicious. The results should be presented in the form of well-formatted reports. The tool should be easy to use and interactive with a usable and convenient interface.

3.2 Proposed Android Forensic Tool



Forensic Tool GUI

The proposed Android forensic tool should perform non destructive acquisition from rooted Android devices. It is a GUI based tool based on python language and libraries. It is proposed to produce results in the web pages. The dump file that contains the entire file system of the android device has to be created prior to the data extraction and analysis. The GUI of the tool expects an input path to the dump folder and an output path for the storage of generated data reports. Further, the user has to select the artifacts for which data should be processed allowing the user to customize the report as per need. Since the data is already extracted in the dump folder, the execution of the tool is fast and the report should be generated without much delay. The risky or potentially attack data is segregated from the normal legit

mobile data and classified accordingly by the tool. The results are finally displayed in interactive, well-formatted reports with a usable interface that can be viewed on any web browser.

3.3 Algorithms

1. Accounts

Purpose : To extract data about user account details

Input : accounts Database

Output : List of name, type, password for user accounts

Algorithm :

Step 1 : Search for the accounts table from database file in the dump

Step 2 : If accounts table is not empty:

Process accounts

Query name, type, password from accounts table

Add data to the respective columns of report

Else:

No accounts data found

Step 3 : Generate report

2. Call logs

Purpose : To extract call data

Input : call_logs Database

Output : Caller ID, Receiver ID, Start and End timestamp, call type

Algorithm :

Step 1 : Search for calls table in the call_logs database

Step 2 : Initialize empty lists for incoming, outgoing, missed calls.

Step 3 : If call_records not empty:

Iterate over call records.

case "Incoming": Append the call details to incoming list

case "Outgoing": Append the call details to outgoing list

case "Missed": Append the call details to missed calls list

Step 4 : Add data to the respective columns of report

Step 5 : Generate report

3. Messages (MMS-SMS)

Purpose : To extract message data from the dump file

Input : MMS-SMS.db Database

Output : Message timestamp, sender, receiver, content, type

Algorithm :

Step 1 : Look into MMS-SMS database

Step 2 : Perform SQL operations(Perform Join operation to extract data from multiple related tables and filter them) and queries on tables in the above database

Step 3 : Generate report

4. WhatsApp

Purpose : To extract data about WhatsApp application installed on the device

Input : wa.db, msgstore.db

Output : WhatsApp contacts data
WhatsApp message data

Algorithm :

Step 1 : Search for the WhatsApp database from the dump

Step 2 : If database is wa.db:

Query data from a table named wa_contacts.

If data not empty:

Append data to WhatsApp contacts list

Step 3 : If database is msgstore.db:

Query data from a table named messages.

If data not empty:

Append data to WhatsApp message data list

Step 4 : Add data to the respective columns of the report. Generate report.

5. Teams

Purpose : To perform forensics to get the data on Teams Application

Input : skype_teams.db

Output : Teams_messages, Teams_user, Teams_call-logs, Teams_activity-feed, Teams_file-info

Algorithm :

Step 1 : Get the required database skype_teams.db

Step 2 : If skype_teams.db not empty:

Retrieve Teams_messages,

Retrieve Teams_user,

Retrieve Teams_call-logs,

Retrieve Teams_activity-feed,

Retrieve Teams_file-info

Step 3 : Generate report

6. Chrome History

Purpose : To extract chrome history from Android device

Input : Chrome Media History folder

Output : ID, URL, title, last_updated_time, watchtime, visit_count

Algorithm :

Step 1 : Check browser details, OS and its type.

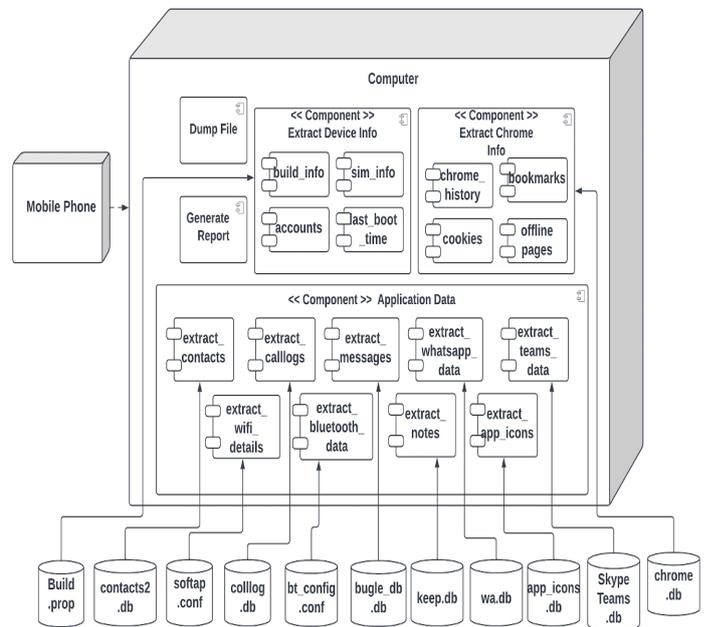
Step 2 : Open files from the input folder namely origin, playback, playback_session.

Step 3 : Retrieve data and map values with defined attribute keys.

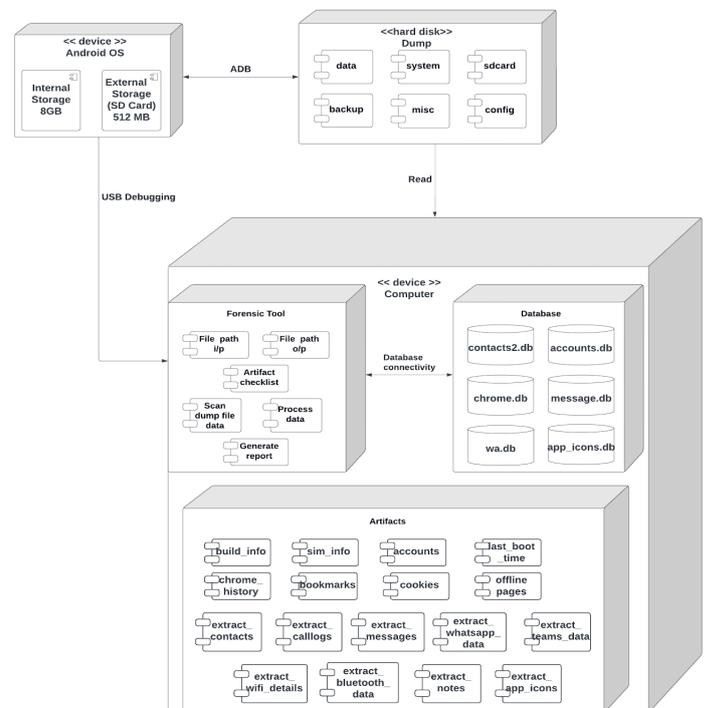
Step 4 : Generate report.

3.4 Software Architecture

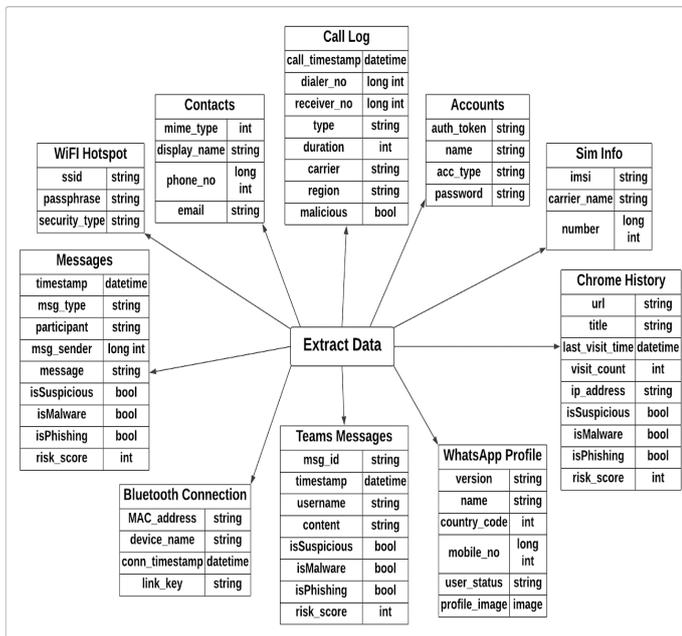
• Component Diagram



• Deployment Diagram



Data Structure



OS	Android 7.1 and higher
Internal Storage	32 GB
SD Card	512 MB

Software Requirements

1. Computer Software Specifications

Android SDK	Android Studio 2020.3.1
IDE	VS Code 1.67.2
Language	Python 3.9.0
Python Libraries	PySimpleGUI 4.16.0, MDB 4.13
Database	SQLite 3.32.2
Android SDK Platform	ADB 1.0.39, Emulator, Device File Explorer
CLI	Windows Command Prompt
Browser	Google Chrome, Firefox, Edge

4. IMPLEMENTATION

4.1 Experimental Setup

Hardware Requirements

We performed the forensic analysis on the Android Emulator of Google Pixel 4. The Emulator and Forensic tool is run in a computer having Windows OS.

1. Computer Specifications

RAM	8 GB (Min)
Processor	Intel i5 8th gen, 2.3 GHz
OS	Windows 10, 11
Hard Disk	128 GB SSD
USB Port	USB 3.0

2. Mobile Specifications

RAM	4 GB (Min)
Processor	Quad core processor, 2.0 GHz
CPU/ABI	x86
Resolution	1080 * 2280: 440 dpi

2. Mobile Software Specifications

Android API Level	25 or higher
Pre-installed Applications	Contacts, SMS, Calls, Google Play
Applications Installed Using Google Play	Whatsapp, MS Teams

4.2 Environmental Setup

Create Android Emulator

Step 1: In the Android Studio Menu bar navigate to Tools -> AVD Manager

Step 2: In the Virtual Device Manager, click on Create Virtual Device Button

Step 3: In the Virtual Device Configuration Window, Select the Category as Phone and Select the Pixel 4 Phone or any other phone having google play support and click Next.

Step 4: Select the system image and click Next.

Step 5: Give name to the AVD, Verify the configurations and click on Finish button.

Step 6: We can see a new Emulator created in the AVD Manager. Run this Emulator by pressing Play Button in Actions

- **Install Required Packages For Forensic Tool**

Step 1: Navigate to project folder in the Command Prompt

Step 2: Install the requirements using the following command

pip3 install -r requirements.txt

4.3 Android Forensics Tool Execution

- **Connect Android device to the computer**

Step 1: Turn on the Android device or emulator. Connect the mobile device to the computer using USB and turn on USB debugging in the settings.



Turn on Mobile Device

Turn on USB Debugging

Step 2: Check the connected Android devices to the computer by running the following adb command in the command prompt

adb devices

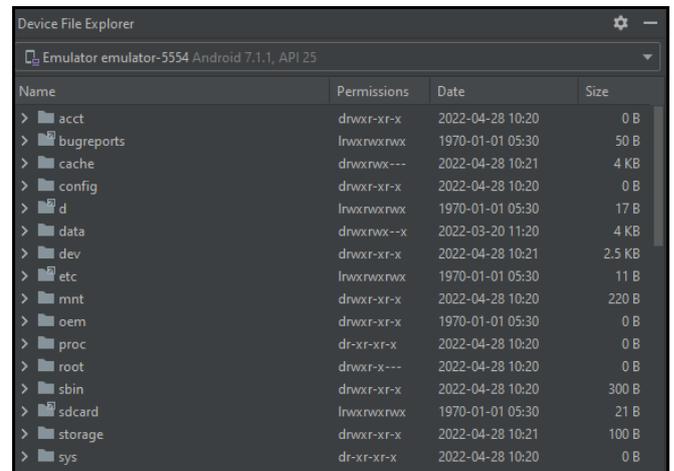
```
C:\Users\kushd>adb devices
List of devices attached
emulator-5554    device
```

Step 3: Get the superuser access to the Android device from the computer, run the adb shell and enter command su and then enter id.

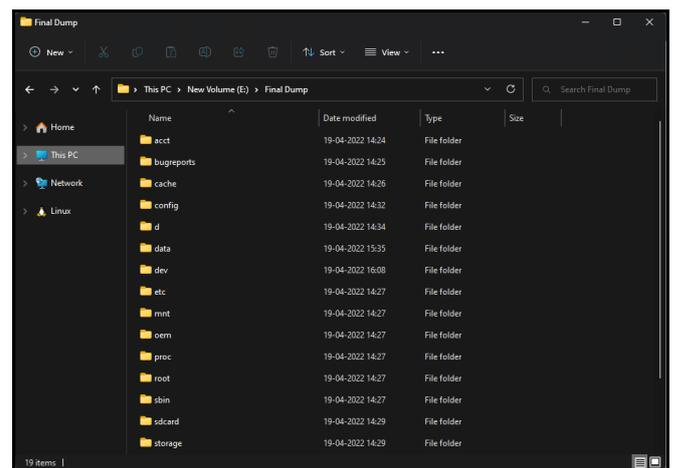
```
C:\Users\kushd>adb shell
generic_x86:/ $ su
generic_x86:/ # id
uid=0(root) gid=0(root) groups=0(root) context=u:r:magisk:s0
generic_x86:/ #
```

- **Create the dump file**

Step 1: In the Android Studio Menu Bar navigate to View -> Tool Windows -> Device File Explorer



Step 2: Select the folders required for dump and save them by providing dump folder path. Contents of the dump folder are shown in next screenshot



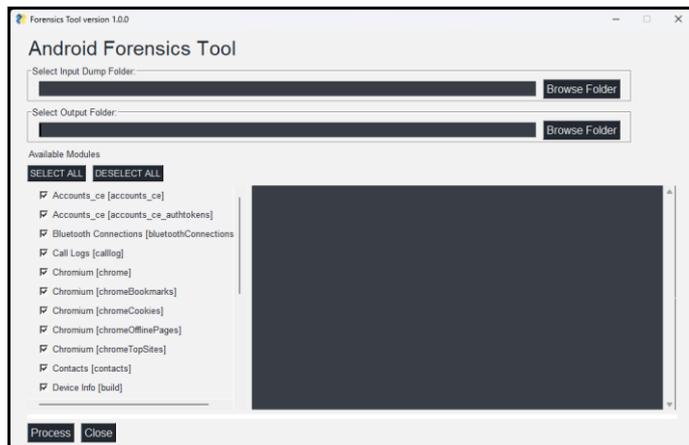
- **Use Android Forensics Tool**

Step 1: Open Command Prompt navigate to the forensic project folder and run the forensic tool using the following command

`python .\forensicsTool.py`

```
E:\>cd Android-Forensics-Tool
E:\Android-Forensics-Tool>python forensicsTool.py
```

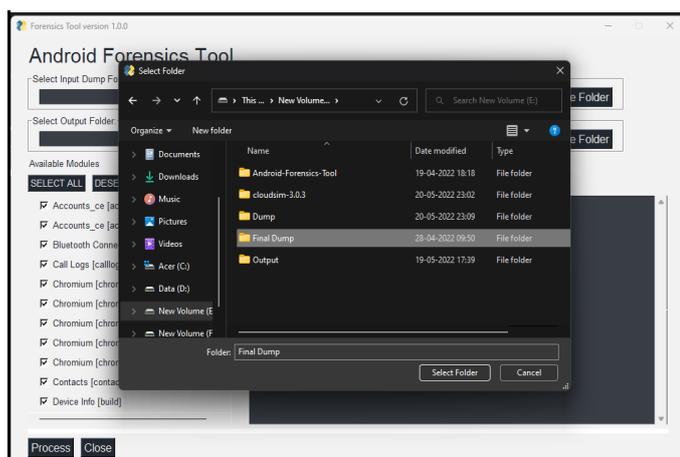
GUI window of Android Forensics Tool opens



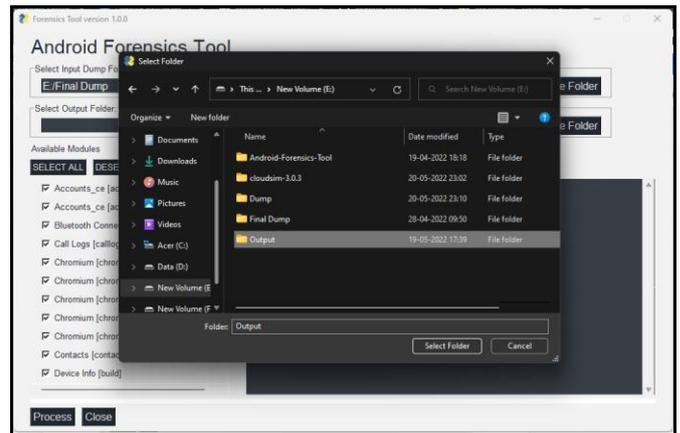
Forensic Tool GUI has four main components:

- i) Input path: To choose input dump folder path
- ii) Output path: To choose folder to store the output/report of the forensic analysis
- iii) Artifacts List: To select among the artifacts available for the forensics
- iv) Execution log display: To see the log of the processing of the forensics tool.

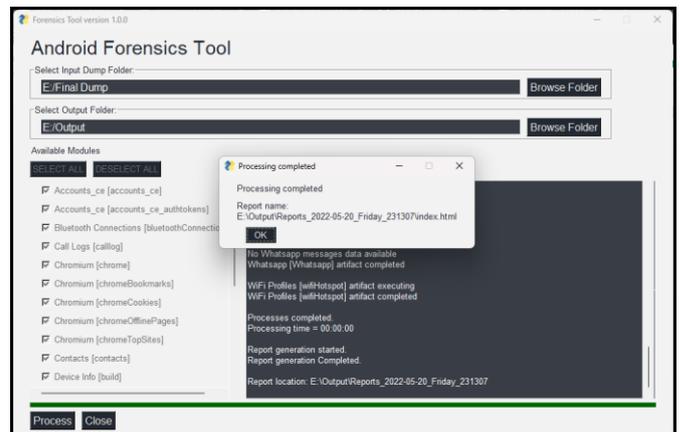
Step 2: Click on the Browse Folder button in Input Dump Folder and choose the Dump Folder.



Step 3: Select output folder to save the forensic analysis output:



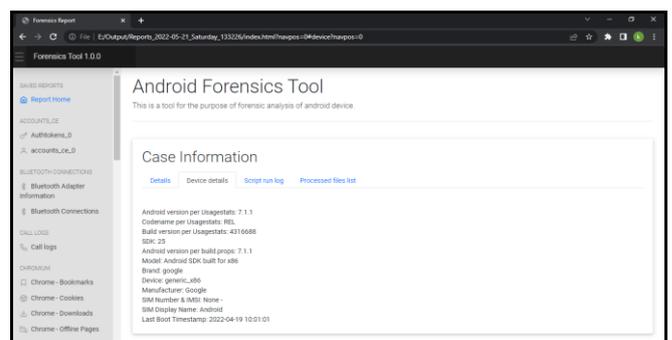
Step 4: After forensic analysis processing is completed we get the pop up window indicating processing complete message and report file name and path. Click the OK button we can see the report in the browser.



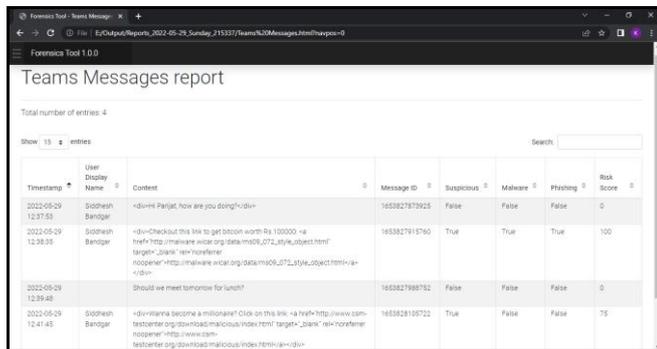
5. RESULTS

We can view the report generated in any web browser window. Following screenshot shows the index page of the report. Report is divided in different tabs based on artifacts and we can navigate to each artifact report by clicking on the artifact name.

1. Device details



10. Microsoft Teams App Messages



Timestamp	User Display Name	Content	Message ID	Suspicious	Malware	Phishing	Risk Score
2022-05-29 12:37:53	Siddhesh Bangde	<div>Hi Parag, how are you doing?</div>	1653827819825	False	False	False	0
2022-05-29 12:38:55	Siddhesh Bangde	<div>Checkout this link to get bitcoin worth Rs. 100000 -> href=http://malware.wor.org/data/m59_072_style_object.html?target='_blank'&rel='noopener noreferrer'>http://malware.wor.org/data/m59_072_style_object.html?target='_blank'</div>	1653827918760	True	True	True	100
2022-05-29 12:39:46	Siddhesh Bangde	<div>Should we meet tomorrow for lunch?</div>	1653827968732	False	False	False	0
2022-05-29 12:41:45	Siddhesh Bangde	<div> Wanna become a millionaire? Click on this link -> href=http://www.com-testcenter.org/download/malicious/index.html?target='_blank'&rel='noopener noreferrer'>http://www.com-testcenter.org/download/malicious/index.html?target='_blank'</div>	1653828105722	True	False	False	75

6. CONCLUSION

The paper initially addresses the complete architecture of the mobile phones, with Android OS architecture being the limelight. The vulnerabilities, threats owing to hardware and software architectures were looked upon. The Android security model was briefly discussed followed by mobile security w.r.t threats and attacks. The few prominent mobile attacks like improper platform usage, information gathering attack etc were also presented. Finally, a detailed discussion about the android forensics was provided.

The field of mobile forensics is bound to attain great value as the ever precious data on smartphones are becoming a target for cyber attacks. The study includes use of open source forensics and analysis tools, frameworks such as Android Debug Bridge, Andriller Tool, MobSF Tool, etc. Their results are documented as a guide for further studying and research purposes.

Finally, an Android forensic tool for the purpose of non destructive data acquisition software tool for rooted android devices was implemented and results were displayed. The tool was implemented with a user-friendly GUI, that takes input an android dump file and an artifact checklist as per the user preference. The extracted data is further checked for any malicious or potential attack data. The tool generates the result as interactive, well formatted HTML based reports with classification of data as safe or risky, malicious. The reports can be viewed on any web browser.

7. REFERENCES

1. Sathe, Sneha C., and Nilima M. Dongre. "Data acquisition techniques in mobile forensics." *2018 2nd international conference on inventive systems and control (icisc)*. IEEE, 2018.
2. Boueiz, Marie-Rose. "Importance of rooting in an Android data acquisition." *2020 8th international*

symposium on digital forensics and security (ISDFS). IEEE, 2020.

3. Almehmadi, Tahani, and Omar Batarfi. "Impact of android phone rooting on user data integrity in mobile forensics." *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2019.
4. Pan , Y., Ge, X., Fang, C., & Fan, Y. (2020) "A systematic literature review of android malware detection using static analysis." *IEEE Access*, 8, 116363-116379.
5. Dar, Muneer Ahmad. "A novel approach to enhance the security of android based smartphones." *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE, 2017.
6. Riadi, Imam. "Forensic Analysis of Android-based Instant Messaging Application." *2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*. IEEE, 2018.
7. Htun, Naing Linn, Mie Mie Su Thwin, and Cho Cho San. "Evidence data collection with ANDROSICS tool for android forensics." *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*. IEEE, 2018.
8. Hoog, Andrew. *Android forensics: investigation, analysis and mobile security for Google Android*. Elsevier, 2011.
9. Elenkov, Nikolay. *Android security internals: An in-depth guide to Android's security architecture*. No Starch Press, 2014.
10. Dwivedi, Himanshu. *Mobile application security*. Tata McGraw-Hill Education, 2010.

8. BIOGRAPHIES



Kush Dabade, B.Tech Student, Dept. Of Computer Engineering and IT, VJTI College, Mumbai, Maharashtra



Parijat Dhalkar, B.Tech Student,
Dept. Of Computer Engineering
and IT, VJTI College, Mumbai,
Maharashtra



Siddhesh Bandgar, B.Tech
Student, Dept. Of Computer
Engineering and IT, VJTI College,
Mumba, Maharashtra



Anshul Shende, B.Tech Student,
Dept. Of Computer Engineering
and IT, VJTI College, Mumbai,
Maharashtra