# A Forensic Approach to perform Android Device Analysis

**Masanam. Sai Prasanna Lakshmi, Pasupuleti Rajesh**

***Abstract— Android smartphones are providing a lot of interesting shreds of evidence to perform forensic investigation. Every installed application has log files which provide some valuable information. Android device can provide potential shreds of evidence which include internal and external storage data, shared preferences, internet artifacts, user data, application data and hidden directories etc [1]. To perform a complete forensic investigation to an android device, the tools available for mobile forensics are highly cost effective. And there are some open source tools which are having limitations i.e., we can read the data in the mobile but we can't extract the data and to proceed for forensic investigation. The forensic investigators will rely on commercial tools which will analyze the entire device and generate the report which is used for further forensic analysis. In order to perform complete analysis of an android device, a forensic approach is proposed which completely based on a command line tool provided by android developers apart from existing commercial forensic tools in the market. This paper presents a forensic analysis using ADB (Android Debug Bridge) tool, which analyses both volatile, non-volatile and network data of an android device. In general, android stores the data in.sqlite files format. In this paper, a tool DB Browser is used for analysing the.sqlite files of an android device and for capturing the network packets to and from a device, the network tools TcpDump and Wireshark is used. The analysis results also present the logs of WhatsApp and facebook applications, which are potential evidences to identify the root cause of the crime.***

***Keywords— mobile, android, forensics, ADB, application, non-volatile, volatile, network, analysis.***

## I. INTRODUCTION

The Android operating system is an Open source, Linux-based and Fully-open mobile platform [3]. It is designed for devices like smartphones, tablets etc. Android was initially unveiled in 2007 and the first commercial Android device was launched on September 2008 by GOOGLE [3]. Later on, Google developed Android TV in support of televisions, Android Auto designed for cars and Wear OS on behalf of wrist watches. Each of them is having their own user interface. As the Android OS is an open source, the code developers are allowed to play with the code as per their needs. Presently Android has the largest community of Application Developers writing and developing a number of applications to aid the functionality of the device. Google introduced different versions of Android and named with desserts such as Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Noughat, Oreo, and the latest Android version is Pie. [4]

The Android platform is mostly composed by using SDK (Software Development Kit). The SDK is a set of tools provided by Google that provides advanced background for creating Android compatible applications. Android applications are written in Java programming language using Application Programming Interface (API). To make an android application a Java source code is developed, compiled and formatted into a bytecode. The renewed code is executed in a virtual machine called Dalvik-VM [5]. Dalvik was designed by Dan Bornstein, which is integrated in the software stack that builds up the Android platform. Dalvik VM is an open source that executes the files in.dex format. The bytecode is translated to Dalvik bytecode and stored in.dex (Dalvik executable) or.odex (optimized Dalvik executable) files with respective bytecode conversions. The dxtool will convert the multiple class files into a.dex format [6]. The Dalvik executables may be customized again when installing onto a mobile device. Dalvik was designed in a way that it permits a device to run multiple instances of the VM efficiently.

Dalvik VM was no longer at runtime in newer Android versions because every aspect of Android OS has been changed moreover Dalvik virtually remains same since day one and considered to be slower when compared to renovated versions of Android. So Dalvik was replaced by ART (Android Run Time). ART translates the application's bytecode into native instructions that are later executed by the device's runtime environment. Because of native execution, it uses less CPU usage which results in less battery drain. ART is the fastest runtime than Dalvik VM because ART does ahead-of-compilation which converts Android apk to.odex to improve Application performance [7].

*ANDROID SOFTWARE STACK:*

The Android software stack comprising applications, an operating system, run-time environment, middleware, services, and libraries. Each layer and the corresponding elements within each layer are integrated to provide the optimal application development and execution environment for mobile devices.

**Revised Manuscript Received on March 10, 2019.**

**Masanam. Sai Prasanna Lakshmi,** M.Tech Student, Department of Computer Science and Engineering, KoneruLakshmaiah Educational Foundation, Vaddeswaram, Guntur District, AP, India (saiprasanna1895@gmail.com)

**Dr.Pasupuleti Rajesh,** Professor, Department of Computer Science and Engineering, KoneruLakshmaiah Educational Foundation, Vaddeswaram, Guntur District, AP, India (rajesh.pleti@kluniversity.in)

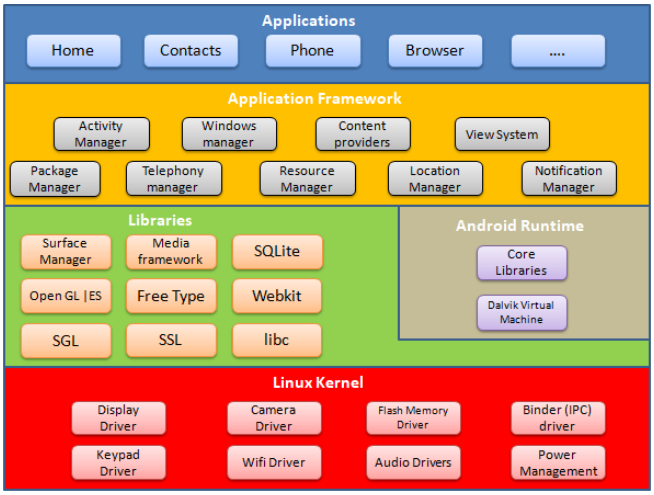



**Figure-1: The components of Android Operating System [8]**

**LINUX KERNAL:-** Linux is an open source platform which provides core features like security, process management, memory management, network stack and driver model. So Linux is used to create Android Operating System. Linux kernel exists at the root of the android architecture.

**LIBRARIES:-** The Android framework was developed various C/C++ core libraries with many open source tools, which is running on the top of the kernel.

The surface manager was responsible for rendering windows, surfaces of various apps on the screen.

The media framework is responsible for media codecs i.e., audio and video.

The Sqlite is used to store database which leaves memory footprints and task execution.

The Webkit Library is responsible for web browsing support.

The OGL (open graphics library) and SGL (scalable graphics library) are responsible for rendering the graphic libraries for 2D & 3D respectively.

SSL is responsible for internet security and Freetype library is used for rendering fonts.

**ANDROID RUNTIME:-** Designed to run the android applications in a guarded environment

**APPLICATION FRAMEWORK:-** The application framework is a set of services that communally form the environment in which android applications run and managed.

**Key services :-**

**Activity manager:** manages the life cycle and activity stack of applications.

**Windows manager:** manages windows, drawing surfaces and the abstraction of surface manager library.

**Content providers:** provides mechanism of exchange (share) data among apps.

**View system:** used to build application user interface.

**Package manager:** manages the information related to application packages of currently installed applications.

**Telephony manager:** manages the application of telephony services.

**Resource manager:** provides access to non-code resources.

**Location manager:** manages location application capabilities.

**Notification manager:** manages the application that displays alerts and notifications.

**APPLICATIONS:-** Located on the top of the Android software stack and comprised with both native applications and third party applications installed from google playstore.

## II. LITERATURE SURVEY

Mobile forensics is a branch of digital forensics that deals with the recovery of digital evidence or data from a mobile device or any device with both internal memory and communication ability such as PDA's, smartphones and Tablets etc., [9]

### *A. Investigative Models in Mobile Forensics:*

**Investigative Models in Mobile Forensics:**

In Mobile Device Forensics, there are three models which are used by the forensic investigators for Data Acquisition and Analysis.

The three models are:

1) CDR Analysis
2) Data Recovery and
3) Third party Application Analysis

In CDR analysis the call data records of a suspected device are analyzed. It's a basic approach to perform mobile device analysis to investigate the crime.

In Data recovery model, we can recover the deleted data along with the data that is present in the mobile to perform the investigation related to particular crime.

In Third party Application Analysis the data regarding any particular app is analyzed i.e., user created data to investigate the crime.

To perform any of this approach on a mobile device the forensic investigators need to depend on commercial tools which generate an automated analysis and report.

Mobile Forensic Process:

The mobile forensic process aims to collect the digital evidence or data from a mobile device and will preserve in a way that avoids damage or tampering of the evidence. To collect the digital evidences from the android devices the android device must be rooted.

Rooting is a process that allows you to attain privileged control i.e., root access of the device. Android uses Linux kernel and rooting the android device gives superuser permissions same as Linux. Rooting is often performed to overcome the limitations of hardware manufactures on the devices. To access the android device with all permissions rooting of the mobile device is must be performed. After rooting the device forensic investigators collect evidences from the device using forensic tools. [10]

## III. DATA ACQUISITION PROCESS

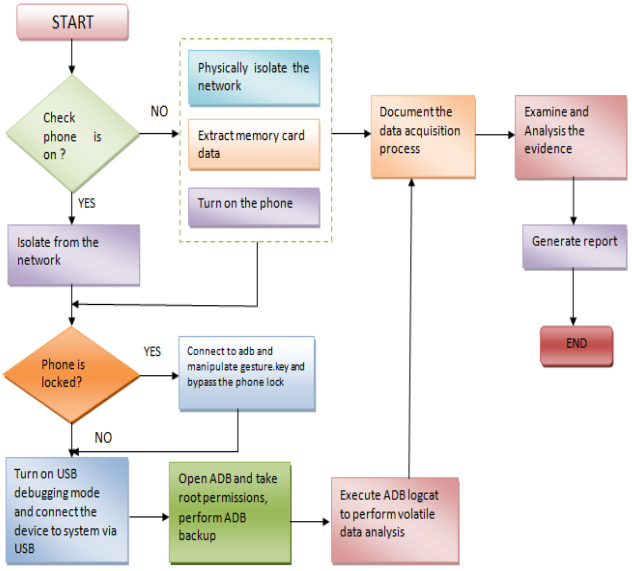



**Figure-2 Work flow for acquiring non-volatile and volatile data from an android device**

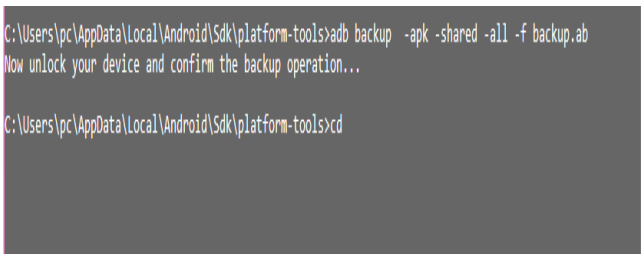
## IV. THE PROPOSED METHODOLOGY

The proposed method of the android device was experimented using android emulator which is in-built virtual device in the software called Android studio provided by Android Developers. To analyse the digital evidences from an android device, the proposed method is divided into three parts.

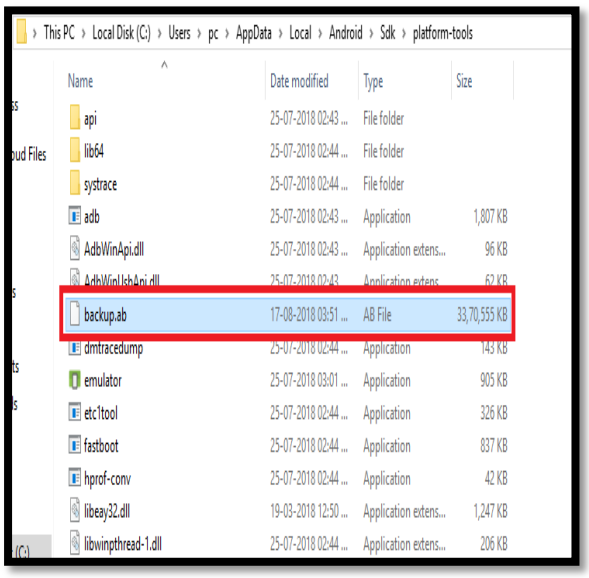
1) Non – volatile data analysis
2) Volatile data analysis
3) Network artifacts analysis

**NON – VOLATILE DATA RESULT ANALYSIS:**

- The Initial step to perform the data acquisition is enabling the USB debugging mode in the android device.
- Now connect the mobile to the system via USB cable.
- Now open ADB and execute the command **adb devices** to list the devices attached.
- To perform backup execute the command ***adb backup –apk –shared –all –f backup.ab***. On the device its will ask the permission to take the backup, click on Backup my full data.
- Backup of the device is successfully created.



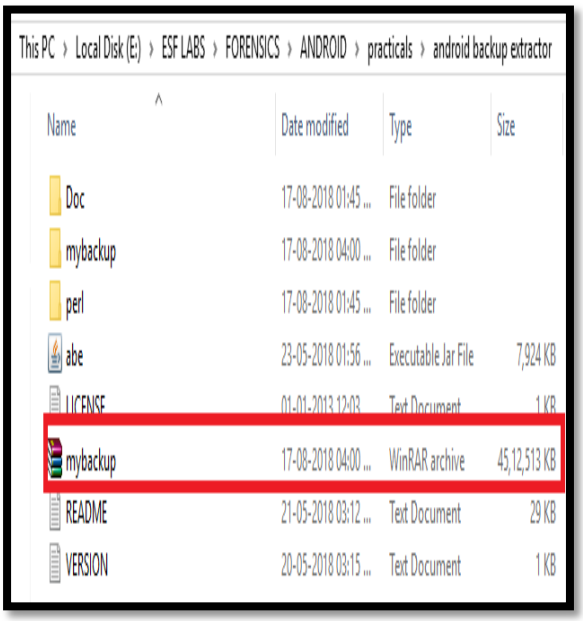

**Screenshot -1 Executing ADB command to perform backup**




**Screenshot -2 The backup of the device completed successfully**

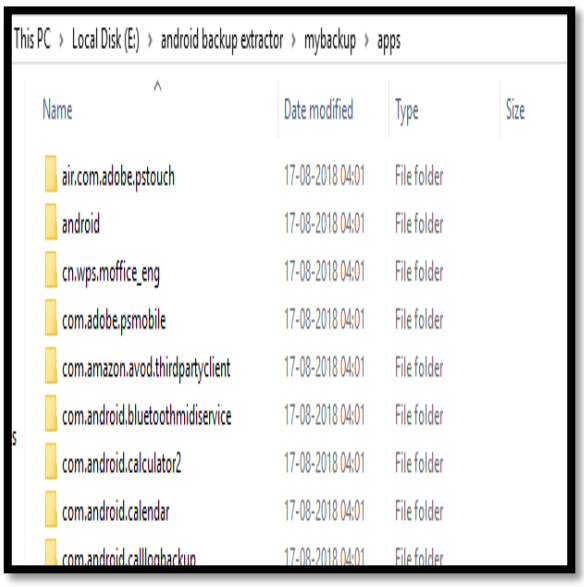
- Now extract the data from.ab file using Android Backup Extractor.
- Open command prompt and run the command ***java -jar abe.jar unpack backup.ab mybackup 1234***.

A file is created with the name mybackup. Rename it with.tar extension and extract the files. The data regarding installed apps in the device is retrieved and can be used for further analysis.



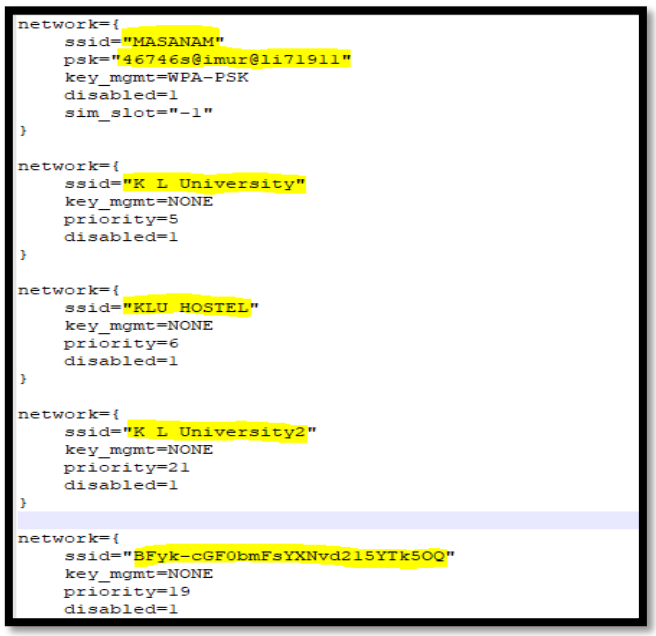

**Screenshot -3 Backup is successfully converted**


Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication




**Screenshot -4 Converted backup successfully extracted**

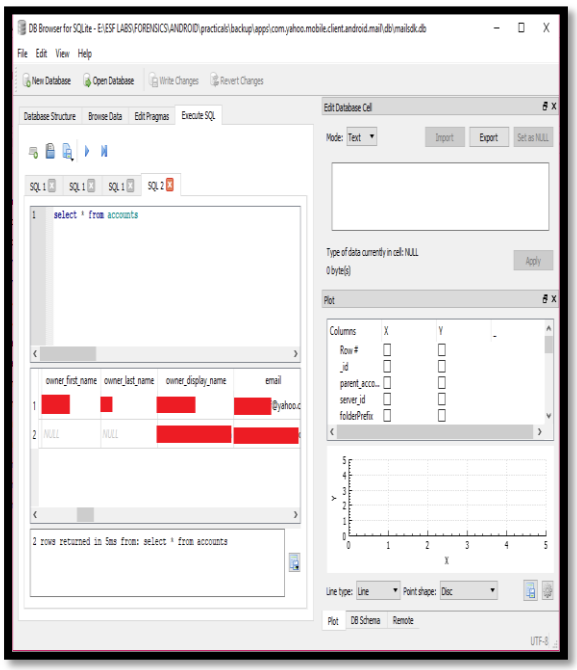
- The extracted appdata contains database files which will store the information of apps in the form of SQLite files. These SQLite files are analysed using a tool called DB Browser. It can be used in a particular third party application analysis also.
- Now go to **com.android.providers.settings\f**. open the Flattened file. You can see the all the Wi-fi connections that device was connected.



```
network={
    ssid="MASANAM"
    psk="46746s@imur@li71911"
    key_mgmt=WPA-PSK
    disabled=1
    sim_slot="-1"
}

network={
    ssid="K L University"
    key_mgmt=NONE
    priority=5
    disabled=1
}

network={
    ssid="KLU HOSTEL"
    key_mgmt=NONE
    priority=6
    disabled=1
}

network={
    ssid="K L University2"
    key_mgmt=NONE
    priority=21
    disabled=1
}

network={
    ssid="BFyk-cGF0bmFsYXNvd2l5YTk5OQ"
    key_mgmt=NONE
    priority=19
    disabled=1
```

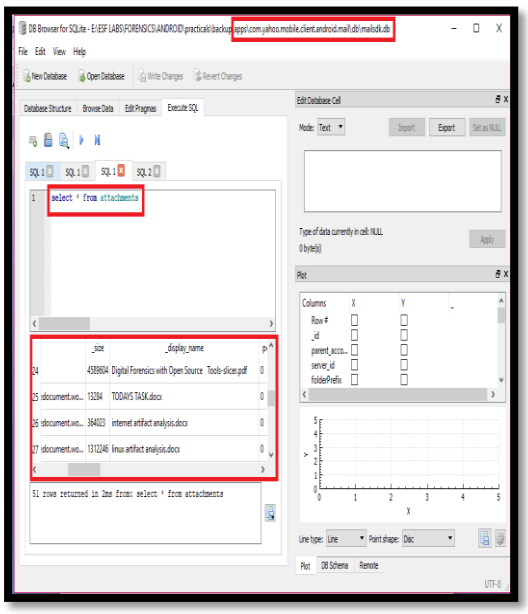
**Screenshot – 5 connected Wi-Fi username and passwords**

- Go to **com.yahoo.mobile.client.android.mail\db** and open mailsdk.db file. And execute a query **select * from accounts** that is listed in Tables. You can see the mail ID thst is logged in.



**Screenshot – 6 mail IDs are identified**

- In the same database file you can see attachments listed in the Table. Execute a query select * from attachments. You can see the attachments send or receive in that mail.



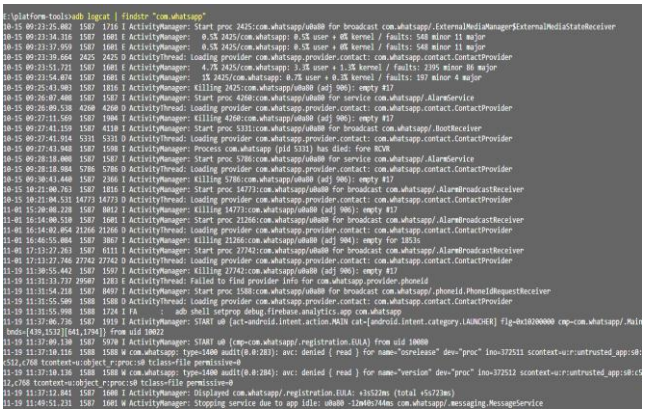
**Screenshots -7 attachments through mail are identified**

**VOLATILE DATA RESULT ANALYSIS:**

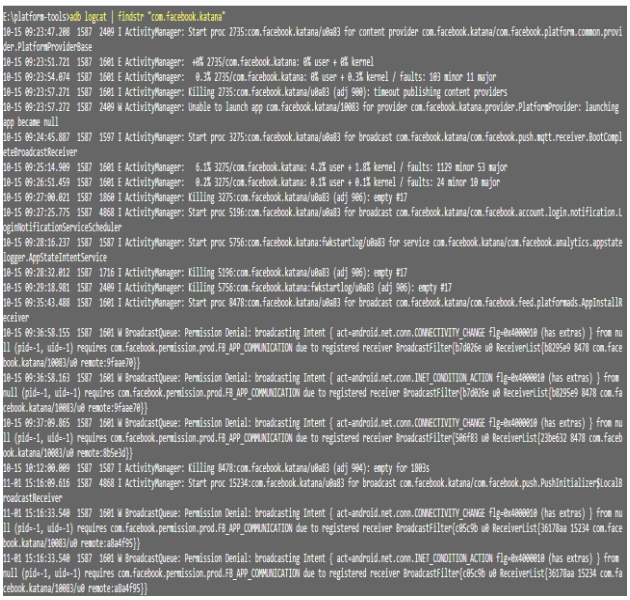
- Open ADB and execute the command ***adb logcat*** to list all the logs related to the running processes on the device.
- To list a particular application use the command adb logcat | findstr "com.example.app". In the below image the logs related to whatsapp & facebook.



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication*



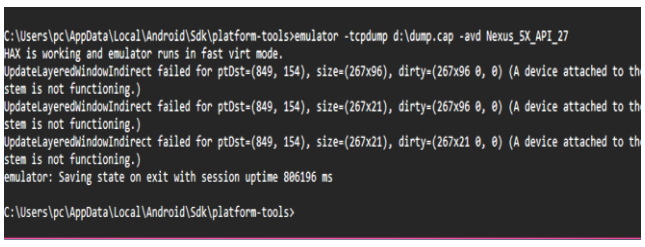
**Screenshot -8 whatsapp logs are successfully listed.**



**Screenshot -9 Facebook logs are successfully listed.**

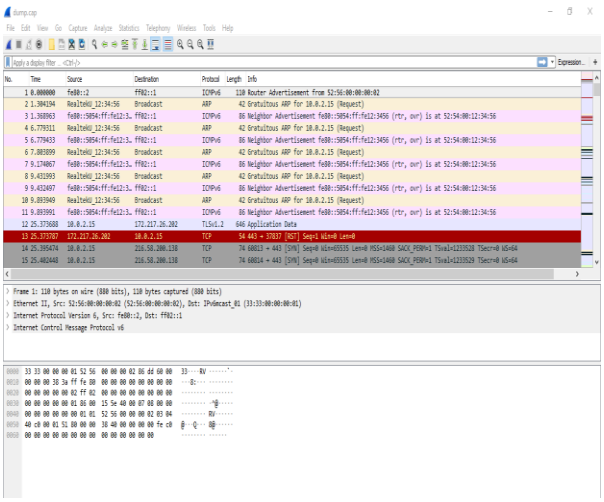
## NETWORK DATA RESULT ANALYSIS:

- Perform tcpdump by executing the command ***emulator –tcpdump d:\dump.cap –avd NEXUS_5X_API_27.*** As an output a pcap file is created.



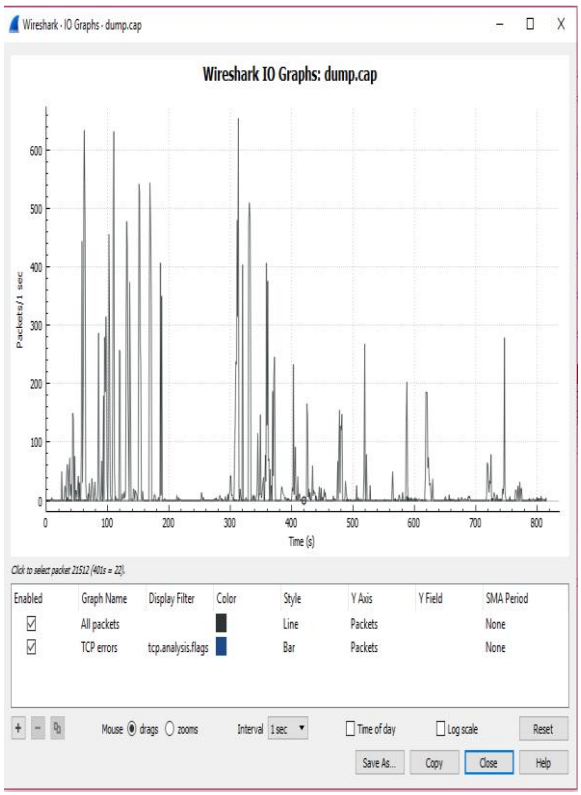
**Screenshot – 10 creating a pcap file using tcpdump**

- Open the pcap file in the Wireshark to analyze the network packets. Different protocols traffic is captured.



**Screenshot -11 Different protocols traffic data was captured**

- The Input and Output graph of the network packets captured.

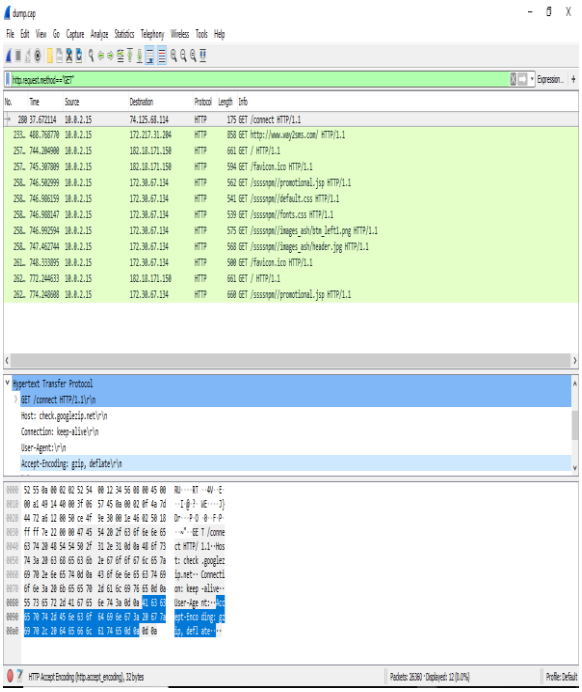



**Screenshot -12 The I/O graphs generated by wireshark**

- The HTTP traffic using display filter http.request.method= =”GET”

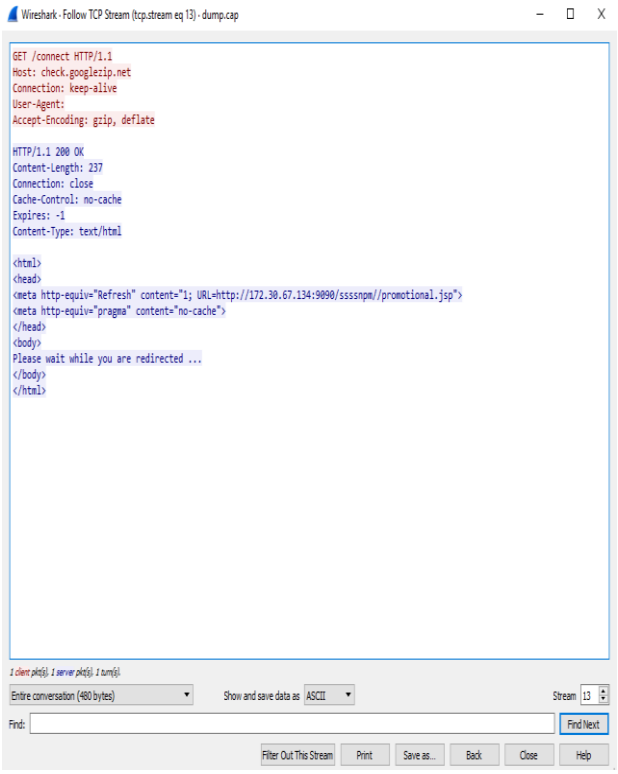


*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication*



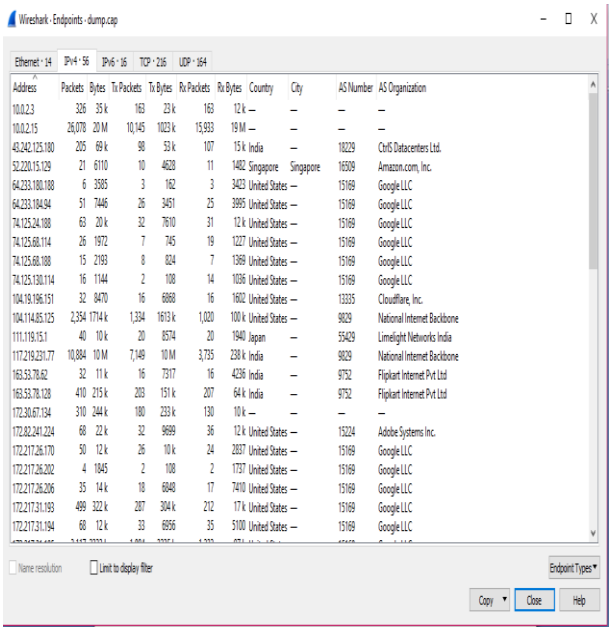
**Screenshot -13 The HTTP traffic captured**

- The tcpstream with the display filter tcp.stream.eq 13. The URL can be identified loaded I the TCP stream.




**Screenshot -14 TCP Stream captured.**

- You can also see the Endpoints in the pcap file. The EndPoint is the packets send and received by a specific address. The Geo location off the server of the particular application is listed.



**Screenshot -15 Geo location of Endpoints is successfully identified.**


## ACKNOWLEDGMENT

This work is supported by my project guide from the Department of Science and Technology and eSF labs. To show my gratitude to my project guide and the individuals helped from the company eSF Labs, who helped me in implementing the project successfully at each and every level of execution.


## CONCLUSION

The smartphones have grown, well-liked with user friendly interface as well as easy to handle. The existing approaches available for the android device forensic investigation are the commercial approaches. The tools used for android forensic analysis are commercial tools which generate a complete report of the device. There are some open source tools which reads only existed data on the phone. We can't extract or analyze that data using them. This paper proposes an open source approach to perform android device forensic analysis which is based on the backend modules of the android operating system. The open source command line approach proposed in this paper defines the data acquisition of an android device using the backup technique and ADB commands. This method can be validated on any device, but if it comes to a challenge like bypassing the phone lock may be difficult because as the new technologies are rising the security features also updating. The bypassing locks of android devices may change as the android versions are updating and uprading. The future work to this approch will be the database files acquired from the apps may be encrypted, which can be able to decrypt by using cryptographic algorithms and also working with android new versions and new updates.

## REFERENCES

1. Hans Hoefken Fachhochschule Aachen "Forensic Analysis of Geodata in Android Smartphones" Researchgate.
2. V. Venkateswara Rao, Dr. A.S.N Chakravarth "Forensic Analysis of Android Mobile Devices" IEEE.
3. Nihar Ranjan Roy, Anshul Kanchan Khanna, Leesha Aneja "Android Phone Forensic: Tools and Techniques" IEEE.
4. G. B. Satrya, P. T. Daely, and S. Y. Shin "Android Forensics Analysis: Private Chat on Social Messenger" IEEE.
5. Normaziah A. Aziz, Fakhrulrazi Mokhti, M. Nadhar M. Nozri "Mobile Device Forensics: Extracting and Analysing Data from an Android-based Smartphone" IEEE.
6. Jianye Liu, Jiankun Yu "Research on Development of Android Applications" IEEE
7. "Android ART: Google finally moves to replace Dalvik, to boost performance and battery life" ExtremeTech
8. "Android architecture – android medical apps" WordPress
9. "Mobile Device Forensics" Wikipedia
10. "Rooting (Android)" Wikipedia
11. Fan Zhou, Yitao Yang, Zhaokun Ding, Guozi Sun "Dump and Analysis of Android Volatile Memory on Wechat" IEEE.
12. João Paulo Claudino de Sousa, João José Costa Gondim "Extraction and analysis of volatile memory in Android systems: an approach focused on trajectory reconstruction based on NMEA 0183 standard" IEEE.
13. Xin Su, Dafang Zhang, Wenjia Li, Wenwei Li "Android App Recommendation Approach Based on Network Traffic Measurement and Analysis" IEEE
14. Abdelkader Lahmadi, Frederic Beck, Eric Finickel, Olivier Festor "A Platform for the Analysis and Visualization of Network Flow Data of Android Environments" IEEE.
15. Kecheng Liu, Wenlong Shen, Yu Cheng, Lin X. Cai, Qing Li, Sheng Zhou, Zhisheng Niu "Security Analysis of Mobile Device-to-Device Network Applications"IEEE.