ISSN:0975-9646

Scrutinize Evidences for Android Phones

Pooja Khandelwal, Divya Rishi Sahu , Deepak Singh Toma

CSE Department, MANIT,
Bhopal (M. P.), India

Abstract— **Android is one of the most popular open source mobile operating systems for smart phones. It facilitates a large number of applications for daily routine tasks ranging from entertainment to effective communication. On the other hand due to its open source architecture it is also vulnerable to malware attacks. These attacks leave traces in android operating system. Hence there is a need to develop a systematic approach to gather evidences from the android operating system. In this paper vulnerability of android software stack has been identified. Further, to improve the investigation process a systematic approach has been explored. This work aims to explore security risks associated with the android smartphone and major problems in android forensics.**

Keywords— ***Malware, Vulnerability, Android Attacks, Android Forensic, Evidences;***

I. INTRODUCTION

Smartphone is a mobile device having advanced features like web browsing, wireless connectivity, 3rd-party apps, GPS navigation etc.

All manufacturers installs different mobile OS in Smartphone such as Microsoft installs Windows, Nokia install Symbian, Google installs Android and Apple install iOS etc. In today's age, Smartphone have become the ubiquitous devices. According to an article in Times of India by Sonal Nerurkar [15], 51-million users in urban India use Smartphone. Android Smartphone is the most popular among its other rival operating systems, with a rapidly increasing market share.

Android is an open platform which is built on top of Linux and contains five layers i.e. Linux, Library, Run Time, Application Frame Work and Application Layer. Fig. 1 depicts the android layer architecture.

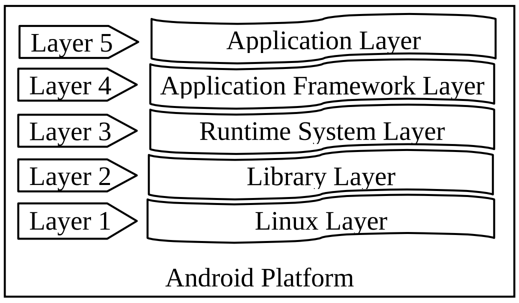


Figure 1: Android Layer Architecture

First layer is Linux layer which works as an interface between hardware and software. It contains drivers like camera driver, binder drive.

Second layer is Library layer that provides SGL, SSL, SQLite libraries to handle different types of data like structured data and web data. These open source libraries have their own vulnerability that causes android applications to be vulnerable, if not handled properly.

Third layer is Run Time layer that contains Dalvik virtual machine which enables android applications to run in its own process. In general android applications cannot communicate with one another. However, android provides communication mechanism to share data between applications. This inter process communication sometimes leads to privilege escalation attack. It also contains a set of core libraries that can be used by applications.

Above the runtime layer, the application framework layer exists that manages primary function of phone such as Resource management, Telephone management and Location Management. This layer contains Application Programming Interface (API) that can be used by applications to interact with rest of the system.

The upper most layer is application layer that contains all the applications either user installed or pre-installed applications. These pre-installed applications may have some flaws like improper handling of content providers, holding more permissions than the actual need etc.

User installed applications request resources of phone in the form of permission. It is up to the user to grant permissions to application or just deny the installation. Thus, as the user has the power to give permissions to an application, the device information is vulnerable because in general, user has no idea how these permissions are going to be used by applications in order to access user data. Due to these vulnerabilities application can misuse user data for their own purpose. Other than this it is also possible to create android botnets, to perform denial of service attacks and repackaging of applications for malicious purpose are some of the measure concern in front of android developers.

In spite of all these flaws android has become the most popular OS in the market. Users install applications that generate huge amount of digital data that is stored in mobile flash memory or SD card. These installed applications can be malicious in nature and to analyze their behavior and impact of applications it is required to forensically examine android phones.

To forensically examine digital data, it is necessary to extract it from mobile phones. Extraction and analysis of digital data is the first & the foremost part of digital forensics. Digital Forensic is defined as [10]:

"The use of scientifically derived methods which preserve, collect, validate, identify, analyze, interpret, document and present digital evidences derived from digital sources in order to facilitate the reconstruction of events found, to be criminal or unauthorized."

As Smartphone are digital device that store data in digital format; so to find out impact of attacks it is required to forensically examine android phones which requires android forensic.

Android forensic is a part of digital forensic in which digital data is collected from Smartphone and investigated to present evidences in the court of law. The most important point is to ensure that the integrity of digital data should not be compromised.

Section II gives an overview of the related works on android attacks and android forensics. In section III vulnerabilities in each android layer have been discussed. Section IV concentrates on problems that can occur in front of forensic expert. Finally in section V a model has been proposed that can be used to forensically examine android Smartphone.

II. Literature Review

Android offers security in the form of permissions, sandboxing of applications, component encapsulation and file access. In spite of all these security measures it is continuously being attacked by attacker.

Tongbo luo et al. [3] describes web view component of android that is used by applications to display web pages. Three types of interactions between android application and web pages are discussed i.e. event monitoring, invoke java from java script, invoke JavaScript from java. Attacks using malicious web pages and attacks by malicious application are discussed.

Alessandro Armando et al. [4] described vulnerability in android process creation i.e. in Zygote process which is responsible for creating all the processes in android operating system. Any process or application can call the zygote process and hence create huge number of processes that will exhaust all the device resources and make the system unresponsive.

Collin milliner [5] presents data leakage in mobile phone based web access. HTTP header is identified that are sent to a particular web server. Intended web site developer can see the contents of header that are directed to that sites web server when accessed by Phone Browser. A tool is developed that analyze the headers directed to web server and retrieve information like IMEI Number, Mobile Number that causes privacy violation.

Heloise Pieterse et al. [6] describe characteristics and trends of android botnet. Main characteristics of android botnets are repackaging an application, receiving commands, messaging, stealing information, downloading additional content and modifying the Android Manifest file. Three phases of android botnet development model i.e. Infection, Propagation and Execution are also described.

Lucas Davi et al. [8] describe Privilege escalation attacks that shows the weakness of android permission model that does not deal with the transitive privilege usages; hence application bypass restrictions imposed by sandboxes.

To prevent attacks and to understand the impact of attacks, at present there are various studies going on in android. Malicious applications can even corrupt or delete user data; so to retrieve deleted data forensic methods like live SD acquisition, kernel based behavior analysis have been proposed.

Takamasa et al. [9] presents the concept of kernel based analysis for the detection of malicious applications. Log and System call of application is collected then signature based classification is performed on the applications. At Linux layer a log collector is implemented that collects all system call made by an application. Then a log analyzer is used that matches activities with signatures described by regular expressions to detect a malicious activity.

Sheng-Wen Chen et al. described the concept of live CD/DVD/USB in computer forensics which needs not access hard disks and forensics friendly [2]. Live SD acquisition technique is used to effective recovery of the deleted data. It uses recovery mode of android to perform data acquisition. This method does not recover damaged data and instant data acquisition.

Thomas Bl¨asing et al. [12] propose an Android Application Sandbox (AASandbox) which performs both static and dynamic analysis on Android programs to automatically detect suspicious applications. Static analysis scans the software for malicious patterns without installing it. While dynamic analysis install the application and analyze the logs at low-level interaction with the system.

Android Application Sandbox (AASandbox) is placed in kernel space and assists to capture the log of the application behavior at the system level. The resulting log file is then summarized and reduced to a mathematical vector for better analysis. The kernel module logs each occurrence of a system call, including those which are performed by processes other than the currently analyzed application. This makes sure that a complete system state is recorded.

III. Android Vulnerabilities

In this section android vulnerabilities are identified and categorized on the basis of android layers. Attacker can perform the attack by exploiting these vulnerabilities which are represented in Table 1.

Table 1: Classification of android vulnerabilities

	Layers	Vulnerability
1.	**Application Layer**	**Applications Vulnerabilities**
2.	**Application Framework Layer**	**Content provider Vulnerability**
3.	**Library Layer**	**Vulnerability related to Web-Kit and SQLite Libraries**
4.	**Runtime System Layer**	**Privilege Escalation Attacks**
5.	**Linux Layer**	**Cross Layer Vulnerabilities**

Millions of third party applications are available on the internet which is developed by different organizations and developers. These applications may contain loop holes that penetrate the system.

Android also contains vulnerabilities that have a direct cause of android security breakdowns those results in tremendous amount of data and financial losses.

A. Application Layer Vulnerability

This layer provides the interface to the user for interacting with Smartphone's framework through applications. All applications residing on Smartphone uses API as an entry point into the rest of the android framework. Vulnerabilities in application layer occur due to the poor coding and flaws in applications. USSD attacks are the examples of the application vulnerability.

It takes advantage of the flaw in android dialer which resides in application layer [13]. Android dialer fails to identify the difference between phone number and USSD codes which allows USSD code execution through the phone. This vulnerability affects ICS, Jelly Bean and older version of android OS.

B. Application Framework Layer Vulnerability

Android OS restrict application to interfere with another application in normal circumstances. But it uses the concept of content provider by which applications can request and share data with another application that is not allowed in normal circumstances. This is a break in android sandboxing mechanism. Attacker may expose the sensitive data through calling the content provider using malicious application if sensitive data is mistakenly leaked in content provider.

C. Vulnerabilities in Library Layer

Android uses open source libraries like Webkit, SQLite, and SSL to provide various functionalities. Web kit library is open source web browser engine and having WebView as the core view class. Mostly free apps use 'addJavascriptInterface' which allows attacker to inject malicious code into web view through victim's web page and access preinstalled applications for malicious purpose. This functionality is vulnerable to remote code execution attack and allows JavaScript to call public methods of the injected java objects. Other than this attacker can also exploit SQLite engine for performing SQL injection attack.

D. Vulnerability in Run Time System Layer

Android applications need permission to grant privilege of accessing resources and to install into Smartphone. When these privileges are not handled properly by applications than it is possible for malicious application to take advantage of privilege application and perform illegal task. This type of attack is known as privilege escalation attack [7]. Dalvik virtual machine of android does not identify that a non-privileged application is accessing system resources with the help of privileged application.

E. Linux Layer Vulnerability

Zygote process resides in Linux layer with the 666 permission. Permission 666 means that Zygote process may read and write on OS and invoke to create the processes [5]. It is responsible for creating all the processes in android operating system.

So malicious application can take advantage of it and invoke huge number of processes that will use all the system resources and make the device unresponsive; causes denial of service attack.

IV. ISSUES IN ANDROID FORENSICS

Android forensic is a new era that is gaining attention of forensic experts. Smartphones are not just used to make a phone call, it provides advance features like mobile banking, video conferencing, web surfing. Data stored in these devices are continuously changing because of internal clock, background processes that change the contents of the memory making it hard for the investigators to acquire the exact image for forensic purpose. As a forensic point of view investigator face challenges in investigation process like connectivity issue, limited storage, background processes and fragmentation problem.

A. Connectivity Issue:

To forensically examine android phones, forensic expert connect the phone to the computer. There are basically two ways to connect it to the computer, first one is via Data cable and second one is via Bluetooth.

In the first case compatibility issue occurs due to the architectural difference in android smartphones manufactured by various vendors. Because android is open source OS so most manufacturers like Samsung, HTC, Micromax changes the architecture of android according to their requirement. Hence separate softwares are required to be installed into the computer to connect the phone of different manufacturers.

Therefore investigator faces a connectivity issue because different manufacturers manages files and folders differently which requires deep analysis of various Smartphone. For instance system partitions are available at different path for each manufacturer's android OS.

Table 2: Path of System Partition of Different Phones

Sr. No	Manufacturer	Path of System Partition
1.	HTC	/proc/mtd
2.	Samsung	/proc/emmc

It also creates a problem to find out exact path of data like system data, user data and applications data.

Second way to connect the phone with computer is Wi-Fi or Bluetooth. In this mode limited data may be accessed because an android device does not allow system data accessible over these connections.

B. Frequent Rotation in Log Entries:

Log entries are available for limited amount of time due to the limited buffer size and frequently rotation in log entries. Therefore if victim's phone log can be collected after some time of attack, quite information will be available for forensics purpose.

Smartphone supports limited storage capacity which limits the size of log file. Log files are stored in fixed size circular buffers i.e. /dev/log/main, /dev/log/radio, /dev/log/event, /dev/log/system which overwrite the previous log entries when size of the buffer is exceeded.

Log generated by Smartphone contains large amount of metadata information. By default the format of the log retrieved using logcat command is brief that contains Priority, Tag, Process ID and Message. To retrieve the log that has priority higher than or equal to information following command is used:

*Adb shell logcat *: I;*

It will return all logs having priority equal to or greater than information. Log generated in android has certain attributes like process ID, Thread ID, Time, Date, Priority, Tag, and Message. So the log can be represented as a set of variables-

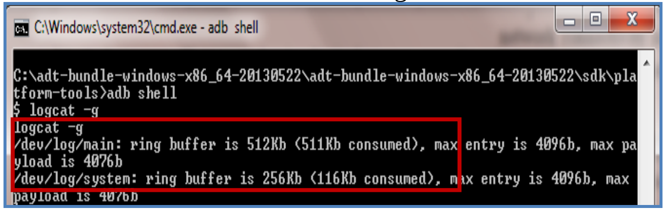
$$L = \{D, T, P, S_c, P_{ID}, T_{ID}, M\}; \qquad (1)$$

{ Where

D	=	Date of log created
T	=	Time of the log created in hh:mm:ss: sss format
P	=	Priority of log
Sc	=	It indicates the system component from which the message originates.
PID	=	Identification number of Process
TID	=	Identification number of Thread
M	=	Message }

To check the size of the log file there is a command provided by android SDK i.e. 'Logcat –g'.

For smartphone that has been tested, the main log size has come out to be 512kb and size of system log is 256kb. Preview for the same is showed in Fig. 2.



```
C:\adt-bundle-windows-x86_64-20130522\adt-bundle-windows-x86_64-20130522\sdk\pla
tform-tools>adb shell
$ logcat -g
logcat -g
/dev/log/main: ring buffer is 512Kb (511Kb consumed), max entry is 4096b, max pa
yload is 4076b
/dev/log/system: ring buffer is 256Kb (116Kb consumed), max entry is 4096b, max
payload is 4076b
```

Figure 2: Limited Log Size

Another factor which affects the log rotation is running processes. Each Smartphone have pre-installed applications such as email, browser, calendar and battery etc. Other than pre-installed applications user can also install applications from play store or internet according to the requirement. So the total installed applications (A_T) will be pre-installed applications (A_{pre}) and installed application (A_{user}).

$$A_T = A_{pre} + A_{user}$$

As each application in android has an associated process with it. So number of running processes (N_P) directly proportional to the number of installed applications in Smartphone.

$$N_P \; \alpha \; k_i * A_T$$

Where –

ki = proportional constant value

Process creates a log entry into log file every time when it runs in android phone. That is for all process there is some logs in the log file. For example if Bluetooth is turned on in the phone than this simple process would write few lines of log in log file. Therefore a relation between log entry and running process can be defined as:

$$R\,(P_i, L_{Pi}) \equiv \forall (P_i)\; \exists\, (L_{Pi})$$

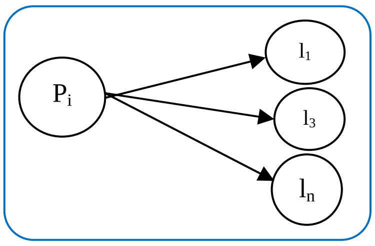
Where –

L_{Pi} = Log of a process P_i

As for every process some log entries are created in the log file. So the number of log entries (N_{LE}) created is directly proportional to the number of processes running in Smartphone.

$$N_{LE} \; \alpha \; k_i * N_P$$

As much as applications run by the user, as much as log created on the Smartphone and causes log rotation to occur frequently. Thus the relationship between processes and logs represents one-to-many relationship as depicts in figure 3.



$$\begin{tikzcd} & l_1 \\ P_i \arrow[ur] \arrow[r] \arrow[dr] & l_3 \\ & l_n \end{tikzcd}$$

Figure 3: One-to-Many Relationships between Process and Logs

As size of log files are limited in android phones causes previous events or processes logs to be missed out.

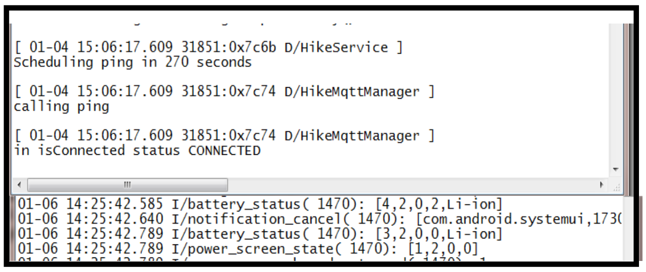
C. Fragmentation Problem:

Android phones have limited internal storage. So the page size is also small and contain limited amount of information. Large files are scattered all around the pages that causes problems in retrieving data for forensic purpose. Other than memory fragmentation android OS fragmentation also exists that causes experts to understand each and every android OS version for better retrieval of information.

D. Background Processes:

Smartphone is the device that always remains in active state or always running. So even when it sits in the idle state there are processes that run in the background like network connectivity etc. which may changes the content of the memory.

Applications like android location service i.e. com.android.location, radio service i.e. com.android.radio, whatsapp i.e. com.whatsapp, hike i.e. com.bsb.hike runs in the background. These continually checks network connectivity and creates a log entry.



```
[ 01-04 15:06:17.609 31851:0x7c6b D/HikeService ]
Scheduling ping in 270 seconds

[ 01-04 15:06:17.609 31851:0x7c74 D/HikeMqttManager ]
calling ping

[ 01-04 15:06:17.609 31851:0x7c74 D/HikeMqttManager ]
in isConnected status CONNECTED
```

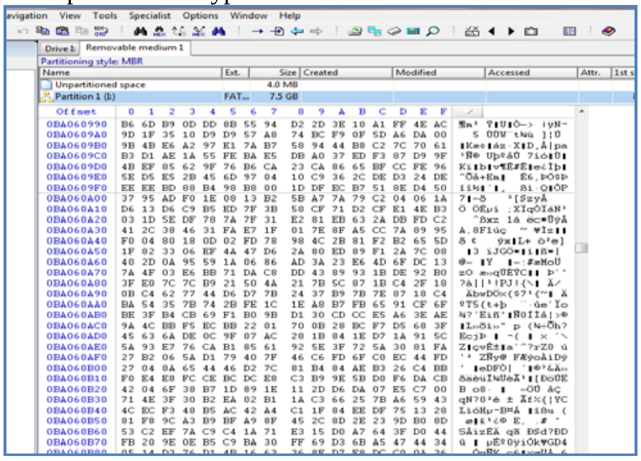
```
01-06 14:25:42.585 I/battery_status( 1470): [4,2,0,2,Li-ion]
01-06 14:25:42.640 I/notification_cancel( 1470): [com.android.systemui,173(
01-06 14:25:42.789 I/battery_status( 1470): [3,2,0,0,Li-ion]
01-06 14:25:42.789 I/power_screen_state( 1470): [1,2,0,0]
```

Figure 4: Log Written By Processes

So even when the end user does not accessing his phone, then even logs are continually written in log files.

E. Extraction:

There are basically two techniques exists to extract data from android devices. First is Physical method which performs data extraction at low level. This technique makes exact copy of the memory and extracts data in encrypted form which requires special tools for the analysis purpose. Figure 5 shows the contents of internal memory stored into Smartphone in encrypted format.

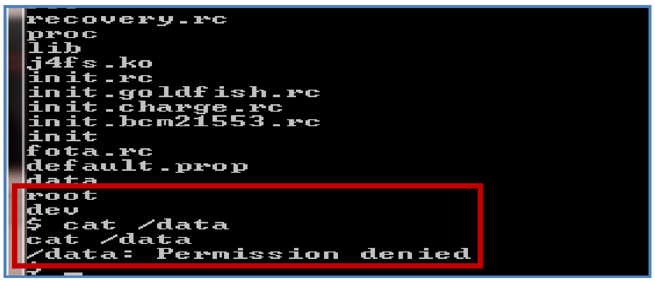

avigation View Tools Specialist Options Window Help
Drive I: Removable medium 1
Partitioning style: MBR

```
Name                   Ext.   Size    Created   Modified   Accessed   Attr.   1st s
Unpartitioned space           4.0 MB
Partition 1 (I:)       FAT...  7.5 GB

Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
0BA060990  B6 6D B9 0D DD 0B 55 94  D2 2D 3E 10 A1 FF 4E AC
0BA0609A0  9D 1F 35 10 D9 D9 57 A0  74 DC F9 0F 5D A6 DA 00
0BA0609B0  9B 4B E6 A2 97 E1 7A B7  58 94 44 B8 C2 7C 70 61
0BA0609C0  B3 D1 AE 1A 55 FE BA E5  DB A0 37 ED F3 87 D9 9F
0BA0609D0  4B EF 85 62 9F 76 B6 CA  23 CA 86 65 BF CC FE 96
0BA0609E0  5E D5 E5 2B 45 6D 97 04  10 C9 36 2C DE D3 24 DE
0BA0609F0  EE EE BD 88 B4 98 B8 00  1D DF EC B7 51 8E D4 50
0BA060A00  37 95 AD F0 1E 08 13 B2  5B A7 7A 79 C2 04 06 1A
0BA060A10  D6 13 D6 C9 B5 ED 7F 3B  58 CF 71 D2 CF E1 4E B3
0BA060A20  03 1D 5E DF 78 7A 7F 31  E2 81 EB 63 2A DB FD C2
0BA060A30  41 2C 38 46 31 FA E7 1F  01 7E 8F A5 CC 7A 89 95
0BA060A40  F0 04 80 18 0D 02 FD 78  98 4C 2B 81 F2 B2 65 5D
0BA060A50  1F 82 33 06 EF 4A 47 D6  2A 80 ED 89 F1 2A 7C 08
0BA060A60  40 2D 0A 95 59 1A 06 86  AD 3A 23 E6 4D 6F DC 13
0BA060A70  7A 4F 03 E6 BB 71 DA C8  DD 43 89 93 1B DE 92 B0
0BA060A80  3F E0 7C 7C B9 21 50 4A  21 7B 5C 87 1B C4 2F 18
0BA060A90  0B C4 62 77 44 D6 D7 7B  24 37 B9 7B 7E 87 18 C4
0BA060AA0  BA 54 35 7B 74 2B FE 1C  1E A8 B7 FB 65 91 CF 6F
0BA060AB0  BE 3F B4 CB 69 F1 B0 9B  D1 30 CD CC E5 A6 3E AE
0BA060AC0  9A 4C BB F5 EC BB 22 01  70 0B 28 BC F7 D5 68 3F
0BA060AD0  45 63 6A DE 0C 9F 07 AC  28 1B 84 1E D7 1A 91 5C
0BA060AE0  5A 93 E7 76 CA B1 85 61  92 5E 3F 72 5A 30 81 FA
0BA060AF0  27 B2 06 5A D1 79 40 7F  46 C6 FD 6F C0 EC 44 FD
0BA060B00  27 04 8A 65 44 46 D2 7C  81 B4 84 AE B3 26 C4 BB
0BA060B10  F0 E4 E8 FC CE BC DC E8  C3 B9 9E 5B D0 F6 DA CB
0BA060B20  42 04 6F 38 B7 1D 89 1E  11 2D D6 DA 07 E5 C7 00
0BA060B30  71 4E 3F 30 B2 EA 02 B1  1A C3 66 25 7B A6 59 43
0BA060B40  4C EC F3 48 B5 AC 42 A4  C1 1F 84 EE DF 75 13 28
0BA060B50  81 F8 9C A3 B9 BF A9 8F  45 2C 8D 2E 23 9D B0 8D
0BA060B60  53 C2 EF 7A C9 C4 1A 71  E3 15 D0 A7 64 3F D0 44
0BA060B70  FB 20 9E 0E B5 C9 BA 30  FF 69 D3 6B A5 47 44 34
```

Fig 5: Encrypted Data stored into internal memory

Second method is logical method. This method uses the Smartphone protocols to extract the evidences from device which are limited for forensic point of view. As android applications store data in /data/data/<package name> folder, this cannot be accessed without having root permissions. Smartphone manufacturers impose these limitations to protect device from mobile malware and viruses. When try to retrieve data from /data/data folder as depicted in Figure 6, throw the message 'permission denied'.



```
recovery.rc
proc
lib
j4fs.ko
init.rc
init.goldfish.rc
init.charge.rc
init.bcm21553.rc
init
fota.rc
default.prop
data
root
dev
$ cat /data
cat /data
/data: Permission denied
```

Figure 6: Logical Extraction

To retrieve internal file and to access the system level features, rooting of Smartphone that is super user permissions are required. Hence rooting a Smartphone breaches its security which may cause damage to the device.

V. MODEL FOR ANDROID FORENSIC

Android forensic can be considered as a seven step process that can be used to forensically examine android phones which are as follows.

A. Purpose

This phase identifies the purpose of investigation, whether attack is performed using the phone, like child pornography or it has been attacked by attacker to perform malicious activities like DoS attack or private data stealing attack, etc.

B. Extremity of Attack

In this phase impact of attack is examined. If the phone is used in criminal activities then identify amount of deleted data by the criminal. If attacks such as DoS, Botnet are conducted on phone then measure the extremity of attack and how much damage caused to the phone resources in order to perform attack.

Severity = No of Exploited Attack * Degree of Damage

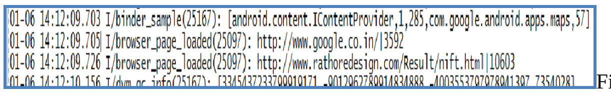
C. Evidence Identification

This phase is required to check the availability of evidences and identification of the appropriate location of evidences according to the attack and resources used. If attacker performed web based attack then evidences may be found from log file, history files, emails, recently visited pages and internal files of applications like data/data/com.google.android.gm and data/data/com.android.browser. For instance victim accesses legitimate application but this application redirects the control to abusing site then the evidences may retrieved from event log file which is available at /dev/log/event location in Samsung galaxy ace.

D. Evidence Collection

Relevant Evidence, such as process ID/application ID, process information, Smartphone log files and deleted data such as messages, images, and contact detail etc. are retrieved from Smartphone and duplicate copy of extracted evidences are created for analysis.

Android SDK provides ADB tool to retrieve information from android phones. Also, tools like via-forensics and AFLogical, can be used to retrieve information such as browser history, deleted messages and contacts. Event log of Samsung galaxy ace contains the information of sites open up by browser under browser_page_loaded tag. Snapshot of event log file is depicted in Figure 7. This could be useful information to check the sites opened by the user.



```
01-06 14:12:09.703 I/binder_sample(25167): [android.content.IContentProvider,1,285,com.google.android.apps.maps,57]
01-06 14:12:09.705 I/browser_page_loaded(25097): http://www.google.co.in/|3592
01-06 14:12:09.726 I/browser_page_loaded(25097): http://www.rathoredesign.com/Result/nift.html|10603
01-06 14:12:10.156 I/dvm_gc_info(25167): [3345437233799919171,-9012962789914834888,-4003553797978941397,7354028]
```

Figure 7: Event Log

E. Analysis

In analysis phase first of all collected data is preprocessed to find out relevant data regarding the attack. For example if dos attack has been performed than process ID, timestamp are relevant information but if web based attacks are performed than network connections,ports,accessed URL could be a relevant information.

F. Validity of Evidences

The primary task in forensics is to retrieve exact copy of the memory. But as mobile phones are always running and dynamic in nature, so the hash values (MD5) tend to change very frequently, the integrity of the copy will be in question [16]. The hash values will change each time when mobile is switched ON/OFF, message is received, signal strength changes, data connectivity state changes etc.

G. Reporting

Presenting relevant evidences extracted from Smartphone in order to generate the report according to the court of law to substantiate amid criminal and victim is very important. A report will be generated that describes the relation and correlation between the user activities and data generated in respect to those activities. Based on the evidences a report will be generated that identifies the malicious activities.

VI. CONCLUSION

Android phones are popular but possess vulnerabilities that are continuously being attacked by attackers. Vulnerabilities of android have been a direct cause of android security breakdowns that have incurred a tremendous amount of data and financial loss. In this research main flaws related to android have been presented. Hurdle in android forensics are also described. A model has been explored which may be useful for law enforcement agency to examine evidences on android phones.

REFERENCES

[1] David Irwin and Ray Hunt "Forensic information acquisition in mobile networks", University of Canterbury, Christchurch, New Zealand, 2009.

[2] Sheng-Wen Chen Chung-Huang Yang Chien-Tsung Liu "Design and implementation of live SD acquisition tool in android smart phone" Fifth International Conference on Genetic and Evolutionary Computing, 2011.

[3] Tongbo Luo, Hao, Wenliang Du, Yifei Wang, and Heng Yin,"Attacks on webview in the android system", ACSAC '11 Dec. 5-9, 2011, Orlando, Florida USA

[4] Alessandro Armando,Luca Verderame "Security issues in the android cross- layer architecture", arXiv:1209.0687v1 [cs.CR] 4 Sep 2012

[5] Collin Mulliner, "Privacy leaks in mobile phone internet access", 978-1-4244-7445-5/10 2010 IEEE

[6] Heloise Pieterse,Martin S Olivier, "Android botnets on the rise: Trends and Characteristics", 978-1-4673-2159-4/12/$31.00 ©2012 IEEE

[7] Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, Marcel Winandy, "Privilege escalation attacks on android"

[8] "Android Logging System", [online] available: http://elinux.org/Android_Logging_System, [Accessed: 22-Nov-2013]

[9] Han Bing "Analysis and research of system security based on android" Fifth International Conference on Intelligent Computation Technology and Automation, 2012.

[10] Francesco Di Cerbo1, Andrea Girardello2,Florian Michahelles2, and Svetlana Voronkova1 "Detection of Malicious Applications on Android OS" IWCF 2010, LNCS 6540, pp. 138–149, 2011.

[11] Paul Morris "Symantec: More than 1000 malicious android apps were published on play store in august ",online[Available],http://www.redmondpie.com/symantec-more-than-1000-malicious-android-apps-were-published-on-play-store-in-august/,[Accessed:22 Dec 2013]

[12] Alexios Mylonas1, Vasilis Meletiadis1, Bill Tsoumas1, Lilian Mitrou1,2, Dimitris Gritzalis " Smartphone Forensics: A Proactive Investigation Scheme for Evidence Acquisition" IFIP AICT 376,2012

[13] Lucian constantin , "USSD attack hit SIM cards and Samsung Android Devices ",Online[Available], http://www.computerworld.com/s/article/9231758/USSD_attack_hit_SIM_cards_and_Samsung_Android_devices, [Accessed:22 Dec 2013]

[14] Leonid Batyuk, Aubrey-Derrick Schmidt,Seyit Ahmet Camtepe, and Sahin Albayrak "An Android Application Sandbox System for Suspicious Software Detection", IEEE,2010.

[15] Sonal Nerurkar, "Teens drive Indian smartphone sales, study finds" online[Available] ,http://timesofindia.indiatimes.com/business/india-business/Teens-drive-Indian-smartphone-sales-study-finds/articleshow/22406572.cms [Accessed : 23 Nov 2013]

[16] Jungheum Park, Hyunji Chung, Sangjin Lee* "Forensic analysis techniques for fragmented flash memory pages in smartphones" J. Park et al. / Digital Investigation 9 (2012) 109–118

[17] Simon Hill "This freedom is free: liberate your android phone by making it a google edition", online[Available]: http://www.digitaltrends.com/mobile/how-to-make-android-phone-google-edition/,[Accessed:25 Nov 2013]

[18] Rohit "Android Master Key Vulnerability—PoC" Online[Available]:http://resources.infosecinstitute.com/android-master-key-vulnerability-poc/ , [Accessed:27 Nov 2013]

[19] Vijith Vijayan, "Android Forensic Capability and Evaluation of Extraction Toolsmore" ,online[Available],http://www.academia.edu/1632597/Android_Forensic_Capability_and_Evaluation_of_Extraction_Tools, [Accessed:22 Nov 2013]

[20] Kyle D. Lutes, Richard P. Mislan, "Challenges in mobile phone forensics", in Proceeding of the 5th International Conference on Cybernetics and Information Technologies, Systems and Applications(CITSA), 2008.

[21] Mohit Kumar, "Another Master Key Vulnerability Discovered in Android 4.3",Online [Available], http://thehackernews.com/2013/11/another-master-key-vulnerability.html,[Accesses:1 Dec 2012].