

Android Forensic Support Framework [AFSF]

MINOR PROJECT REPORT

submitted by

Shankar Raman R

AM.EN.P2CSN13021

in partial fulfillment for the award of the degree

of

MASTER OF TECHNOLOGY

IN

CYBER SECURITY SYSTEMS AND NETWORKS



AMRITA CENTER FOR CYBER SECURITY

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

Kollam - 690525

DECEMBER 2015

Android Forensic Support Framework [AFSF]

MINOR PROJECT REPORT

submitted by

Shankar Raman R

AM.EN.P2CSN13021

in partial fulfillment for the award of the degree

of

MASTER OF TECHNOLOGY

IN

CYBER SECURITY SYSTEMS AND NETWORKS

Under the guidance of

Prof. Prabhaker Mateti

Associate Professor

Computer Science and Engineering

Wright State University



AMRITA CENTER FOR CYBER SECURITY SYSTEMS AND NETWORKS

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

KOLLAM - 690525

DECEMBER 2015

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, KOLLAM -690 525



BONAFIDE CERTIFICATE

This is to certify that this minor project entitled “ **Android Forensic Support Framework**” submitted by **Shankar Raman R, Reg.No : AM.EN.P2.CSN13021** in partial fulfillment of the requirements for the award of the **Degree of Master of Technology** in **CYBER SECURITY SYSTEMS AND NETWORK** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering.

Prof. Prabhaker Mateti
(Supervisor)

Dr. Krishnashree Achuthan
(Professor and Head)

This minor project was evaluated by us on.....

INTERNAL EXAMINER

EXTERNAL EXAMINERS

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ENGINEERING, KOLLAM
AMRITA CENTER FOR CYBER SECURITY SYSTEMS AND
NETWORKS

DECLARATION

I, SHANKAR RAMAN R, (Reg.No: AM.EN.P2.CSN13021) hereby declare that this minor project entitled “**Android Forensic Support Framework**” is a record of the original work done by me under the guidance of **Prof. Prabhaker Mateti**, Associate Professor, Computer Science and Engineering, Wright State University, USA and this work has not formed the basis for the award of any degree / diploma / associateship / fellowship or a similar award, to any candidate in any University, to the best of my knowledge.

Place : Kollam

Date :

Signature of the Student

COUNTERSIGNED

Dr. Krishnashree Achuthan

Professor and Head, Centre for Cyber Security Systems and Networks

ACKNOWLEDGMENTS

At the very outset, I would like to give the first honors to **Amma, Mata Amritanandamayi Devi** who gave me the wisdom and knowledge to complete this minor project under her shelter in this esteemed institution.

I express my gratitude to my guide, **Prof. Prabhaker Mateti**, Associate Professor, Computer Science and Engineering, Wright State University, USA, for his valuable suggestion and timely feedback during the course of this minor project.

I would like to thank **Dr Krishnashree Achuthan**, Professor and Head of the Center for Cyber Security, for giving me useful suggestions and his constant encouragement and guidance throughout the progress of this minor project.

I would like to thank my co-guides **Mr. Vipin Pavithran**, Assistant Professor, Cyber Security, Amrita Vishwa Vidyapeetham for giving me useful suggestions and guidance throughout the progress of this minor project.

I express my special thanks to my colleagues who were with me from the starting of the minor project, for the interesting discussions and the ideas they shared. Thanks also go to my friends for sharing so many wonderful moments. They helped me not only enjoy my life, but also enjoy some of the hardness of this minor project.

hail plate?

Abstract

Smartphones have become part of our every day life. We tend to store our personal information such as our contacts, every day notes, bank details, account credentials, official documents etc. Also these devices shows to contain a large volume of other useful information such as Email, Browser history, Network connection and Chat logs.

Due to the potential information found in the Smartphones it has attracted criminals to commit crimes using them. They could use Smartphones for a number of activities such as committing fraud over email, harassment through text message, trafficking of child pornography, stealing the credentials.

Every action we perform on these devices leaves the traces behind. It is possible to reconstruct an action by identifying a relation among these evidences. Analysing and processing the evidences is a Forensic proces. We integrate forensic support to the existing Android ROM which collects all the possible evidence files and stealthily trasfers them to the cloud for analysis purpose.

Keywords: Mobile Forensics, Android Forensics

poor English
Have NOT even
proof read.

Contents

ABSTRACT	i
List of Figures	iii
List of Tables	iv
Abbreviations	v
1 INTRODUCTION	1
1.1 Importance of Forensic Support in Android	2
2 LITERATURE SURVEY	3
2.1 Papers Referred	3
2.1.1 Android Forensics: Simplifying Cellphone Examination	3
2.1.2 Android Antiforensics through a Local paradigm	3
2.1.3 Security analysis of Android Factory Resets	4
2.1.4 Security analysis of consumer-grade Anti-Theft Solutions Provided by Android Mobile Antivirus Apps	7
2.1.5 Analysis of WhatsApp Forensics in Android Smartphones	8
2.2 Background	8
2.2.1 Rooting the Emulator	8
2.2.2 Build Linux Kernel	9
2.2.3 Application Reversing	10
3 RELATED WORK	11
3.1 Android Forensic Support Framework	12
3.2 Symantec Smartphone Honey Stick Project	14
3.3 Automated Data Collection and Reporting from a mobile device	17
4 PROPOSED WORK	18
5 CONCLUSION AND FUTURE WORK	22
Bibliography	23

List of Figures

2.1	Rooting the emulator Screenshot 1	8
2.2	Rooting the emulator	9
3.1	AFSF: Architecture Diagram	12
4.1	Wireshark Screenshot	19
4.2	Recovering Deleted file using Inode Table	21

List of Tables

1.1	Global market share held by the leading smartphone operating systems	. 1
-----	--	-----

Abbreviations

AFSF **A**ndroid **F**orensic **S**upport **F**ramework

many more
eg.

Chapter 1

INTRODUCTION

The introduction of smartphones drastically changed the communication landscape. Right from chatting with Whatsapp to online banking transactions can be done using smartphones. Even from a student to working professional smartphones are now becoming an indispensable object in their life. In 2014 about 85 percent of the smartphone users are having Android. Android dominates in the number of application in the Play Store compared to other App Stores. In 2014 about 1.3 million apps are active on Google Play, this dominance is because of the high end features Android is providing.

Nowadays mobile phones are not just for calling and sending messages. They are as powerful as a desktop computer; a high end smart phone can do almost all computing which is done by a normal desktop computer and much more advanced features like NFC, Global Positioning etc. This increased availability of computing power for Smartphone catches the attention of criminals. They easily target mobile users as they are less aware and bother less about the risks associated with the mobile and mobile applications. It is quite obvious that the widely used platform is likely to be targeted more, as in the case of

Year	Android	iOS	Windows	Blackberry
2011	36.1	18.3	1.2	13.6
2012	69.3	16.6	3.1	4.9
2013	79.6	13	3.4	2.8
2014	84.7	11.7	2.5	0.5

cite?

TABLE 1.1: Global market share held by the leading smartphone operating systems

Microsoft Windows Operating System. A hacker wants to target mass and for doing that he has to target the most commonly used platform. Android is one such commonly used platform. As Android is capturing market, it is becoming favourite target platform of hackers. There are many Forensics available, these are primarily targeted on applications database files and to extract the deleted data are difficult. Usually data which are used by messaging and mail application stores on encrypted form and to extract these data are hard with the usual forensic tools. There are limitations on collecting deleted data from the device.

1.1 Importance of Forensic Support in Android

With these increased processing ability of smart phones, there is also ~~a potential increase~~ ^{are} for criminals to ~~use~~ ^{use} this technology as well. Smart phones can be used for some illegal purposes like forwarding fraud e-mails, communications related to terrorism, etc. Smart phones stored data are extremely useful to analysis during the course of investigation. Analysing basic mobile device investigators can collect call history, contact, and text message data; smart phones contain even more useful information, such as e-mail, browser history, and network connections, etc. These data are harder to acquire once the user delete ^s those information. Knowledgeable criminals can erase the forensic data from the mobile device. There are Mobile applications like Uninstall-It which can delete all the application data from device.

learn proper use of
singular/plural.

Background

make it
a later
chapter

Chapter 2

LITERATURE SURVEY

2.1 Papers Referred

2.1.1 Android Forensics: Simplifying Cellphone Examination

The paper was authored by Jeff Lessard (Champlain College) and Gary Kessler (Edith Cowan Univ) in September 2010. The paper was published in Small Scale Digital Evidence Forensics Journal and it has around 54 citations till date. The paper describes the challenges faced during the acquisition of evidence files from mobile devices and their limitations. They have demonstrated their work by using various tools such as *dd*, *FTK*, *adb* and *CelliBrite* tools.

2.1.2 Android Antiforensics through a Local paradigm

The paper briefly explains the forensics techniques that is applied to mobile devices and the tools (commercial/general) used to image the storage mediums. Their automated work is then explained elaborately and their experimental results are shown at the end along with their future work.

Anti-Forensics techniques aims at impeding the acquisition and analyzing the evidence. The common anti-forensics techniques include Destroying the evidence, Hiding evidence, Eliminating evidence sources, and Counterfeiting evidence.

Their work takes advantage of an Android feature called Private folder, a folder in which every an application installed in the device can store files and information. There are can be more than one private folder. The feature of the private folder is,

- Applications may store potential information in these folders.
- These folders has protection from accessing the data by any other application or default application managers in the device.
- The informations stored in these private folders are not easily visible to users, device apps and to some tools used for recovery (Scalpel).
- The contents of these folders will be logically erased when the corresponding application is uninstalled from the device.

They leverage the private folders features and they have used an application called AFDroid to carry out the anti forensics techniques. The application creates private folders to store and to remove evidence files. For storing the evidence files, the AFDroid relies on the Evidence Export Process (EEP), Evidence Import Process (EIP) to reconstruct data from the exported content and Evidence Destruction Process (EDP) to destroy the evidence files using the private folder (Anti-forensics).

Their experiments were carried out on a Samsung Galaxy i7500 mobile. They used Nandroid, MIAT and unyaffs to image and analyze the output image files. Unfortunately the commercial tool Parben was not compatible with their device. The EEP process was launched and the results were stored in AFDroids private folder. Casual examination failed to populate evidence from from SMS db, call logs, and multimedia gallery. The EDP process was also executed and they were not able to extract evidence files stored in the private folder using Scalpel, MIAT, and Nandroid Backup.

2.1.3 Security analysis of Android Factory Resets

The paper provides the comprehensive study and reports on these issues,

- quantifies the amount of data left behind the flawed implementation.
- detailed analysis on the flawed devices.

- reveals the drivers which are flaws.
- practical solutions to mitigate these issues.

What did they gain after performing the flawed factory reset on a device,

- Access to Google account credentials
- Associated data (contacts) with the Google services
- Wi-fi credentials.
- Whatsapp conversations
- Gmail app emails

Their study unveils the critical failures on android devices,

- Lack of android support for proper deletion of files in data partition in v 2.3x devices.
- Vendors making users to download incomplete upgrades to their flawed device.
- Lack of driver support for proper deletion (drivers are shipped by vendors)
- Lack of android support for proper deletion of internal and external SD card in all OS versions.
- Easily mitigating the full disk encryption up to KitKat.

Secure Deletion Levels

After deleting a file an OS typically deletes the it's names from the table, rather than deleting it's content. There exist various techniques for sanitising the data. The highest level of sanitisation is *analog sanitisation*, where we will apply a write pass of a fixed pattern across the media surface. The pattern can be either a zero or pseudorandom numbers. This level also includes Cryptographic erase method. The next level is *Digital sanitisation* and the final level is *logical sanitisation* which is performing the factory reset settings.

Linux Kernel Deletion APIs

This section gives an overview of the various parameters passed to the *ioctl()* system call

to erase flash blocks. On a raw flash, when `ioctl()` system call is made with *MEMERASE* option it provides digital sanitisation. On an *eMMC* there are two possible options *BLKDISCARD* which will not purge the blocks internally and *BLKSECDISCARD* provides secure deletion. The options available for logical sanitisations are *ERASE, SECURE TRIM, SECURE ERASE, SANITIZE*.

Analysis of Android Factory Resets

Priliminary results of the paper shows that the external storage is not sanitised unless the *External Storage* option is not selected in the Factory Settings. The OS performs the sanitisation of the external storage and reboots into Recovery/Bootloader mode where the data and cache paritions are sanitised.

90% of the Gingerbread devices failed to sanitize the *data partition* as they were using *BLKDISCARD* option with `ioctl()`. Secure command `ioctl(BLKSECDISCARD)` for logical sanitisation was introduced when Ice cream sandwich was released. But the vendors omitted the device drivers that supports the `ioctl(BLKSECDISCARD)`. The secure command was not supported by `ioctl()` and returned `EOPNOTSUPP`. Problems still existed in android versions such as ICS and JB due to the lack of device drivers support for the secure deletion command. Due to device drivers alone 15% of the devices resulted in improper sanitisation of the data partition.

No froyo or Gingerbread versions logically sanitize the *primary SD card*. AOSP just provides some sanitisation by calling `Fat::format()`, which can only sanitise few dozens of MB. Following android version JB did introduce `ioctl(BLKDISCARD)` for the primary SD card but since *BLKDISCARD* is not effective there were improper sanitisation.

Alternative Sanitization Methods There are third party apps which can be used to write random bytes on the partition of interest which can write the unallocated space. Providing Full Disk encryption helps normal users when they enable it before they start using their mobile. Overwriting the partition bit-by-bit provided logical sanitisation for all devices and all partitions and not digital sanitisation.

2.1.4 Security analysis of consumer-grade Anti-Theft Solutions Provided by Android Mobile Antivirus Apps

The paper is about the study made to prevent unauthorized access to stolen android devices. The experiments are conducted with the top 10 Mobile Antivirus (MAV) apps. A partial solution to this problem is provided by Mobile Antivirus Apps with their remote lock and wipe features. The paper provides solutions and discusses various flaws which are present in those MAVs.

Their analysis yields 7 issues.

1. Incorrect notification to the user after a remote lock/wipe has been performed
2. Questionable MAV authentication practices
 - MAV accepts short passwords and some do not accept special character.
 - Few MAVs such as Norton have failed to limit online password guessing.
3. Limitations and restrictions imposed by Android APIs and architecture
4. Misuse of Android security API by MAVs
5. Inconsistency of android APIs in every release
6. Incorrect Android API documentation
7. MAV reliance on carrier network in security.

Findings

- Because of API difference in Android versions, 9 MAVs can be uninstalled before a Gingerbread phone is remotely locked.
- Because of API misuse 4 MAV locks can be bypassed.
- Because of vendor customization all MAVs can be circumvented.

All shallow summaries

2.1.5 Analysis of WhatsApp Forensics in Android Smartphones

The paper focuses on the analysis conducted on the WhatsApp messenger. WhatsApp messages are stored in the Internal memory of the mobile phones. The analysis has shown that the messages were not stored encrypted. In the recent versions of the messengers uses encryption with custom AES algorithm. But still there are techniques which can allow the retrieval of these keys which can be used to decrypt the encrypted messages stored in msgstore.db.crypt

2.2 Background

2.2.1 Rooting the Emulator

1. Install busy box or push the busy box application using *adb*
2. Find out the block device for /system partition using *mount*
3. Remount the /system partition in rw mode
\$ adb shell mount -o rw,remount -t yfs2 /dev/block/mtdblock0 /system
4. Change the *su* binary permission to 06755

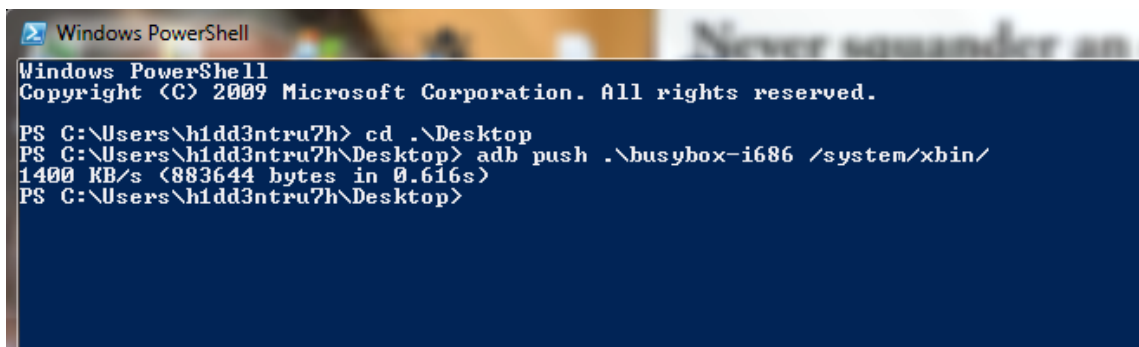


FIGURE 2.1: Pushing busybox binary into the emulator

```

root@generic_x86:/ # mount -o rw,remount -t ya /dev/block/mtdblock0 /system
root@generic_x86:/ # shell chmod 06775 /system/xbin/su
/system/bin/sh: shell: not found
root@generic_x86:/ # chmod 06775 /system/xbin/su
root@generic_x86:/ # uname -a
Linux localhost 3.4.67+ #38 PREEMPT Mon Jul 13 11:06:05 PDT 2015 i686
root@generic_x86:/ # chmod 555 /system/
app/      etc/      lib/      priv-app/ vendor/
bin/      fonts/   lost+found/ tts/      xbin/
build.prop framework/ media/    usr/
chmod 555 /system/xbin/busybox-i686
root@generic_x86:/ # chmod 555 /system/xbin/busybox-i686
root@generic_x86:/ # mv /system/xbin/
add-property-tag      kexeload      perfprofd      sqlite3
busybox-i686          ksminfo       procmon        strace
check-lost+found      latencytop    procrank       su
cpustats              librank       puncture_fs    taskstats
cpustats              micro_bench   rawbu          tcpdump
daemonize             micro_bench_static sane_schedstat timeinfo
dexdump              nc            showmap
directiotest          netperf       showslab
iperf3                netserver     simpleperf
root@generic_x86:/ # mv /system/xbin/busybox-i686 /sys
sys/      system/
v /system/xbin/busybox-i686 /system/xbin/busybox
root@generic_x86:/ # mv /system/xbin/busybox-i686 /system/xbin/busybox

```

FIGURE 2.2: Rooting the emulator

2.2.2 Build Linux Kernel

Building the Cross Binutils:

```
$export PATH="$HOME/opt/cross/bin:$PATH"
```

Configuration

```
$ ./binutils-2.23.1/configure --prefix=$HOME/opt/cross target=i586-elf --disable-nls
```

```
$ sudo make
```

```
$ sudo make install
```

Building the Cross Compiler:

* Create a folder gcc-build just outside the gcc-4.7.2

```
$ mkdir gcc-build
```

```
$ cd gcc-build
```

```
$ ./gcc-4.7.2/configure --prefix=$HOME/opt/cross --target=i586-elf --enable-languages=c,c++
--disable-nls --without-headers
```

```
$ sudo make all-gcc
```

```
$ sudo make install-gcc
```

```
$ sudo make all-libgcc (Check the next point if u get any error)
```

```
sudo make install-libgcc (Check the next point if u get any error)
```

* If you get the error: *** No rule to make target...

```
$ sudo make all-gcc all-target-libgcc
```

```
$ sudo make all-gcc all-libc++
```

```
$sudo make all-gcc all-libdecnumber
$sudo make all-gcc all-libelf
$sudo make all-gcc all-libiberty
$sudo make all-gcc all-libiconv
$sudo make all-gcc all-libtermcap
$sudo make all-gcc
$sudo make install-gcc
```

Testing can be done using Qemu

```
$ qemu-system-i386 -cdrom myos.iso
```

2.2.3 Application Reversing

To reverse a apk file, we can use *dex2jar* and *jd-gui* utility. APK files are ZIP archives which can be extracted to get the Dalvik dex file.

```
$ unzip flickrj-android-sample-android.apk
```

Using the *dex2jar* utility we can convert the dex file to a jar file

```
$ dex2jar classes.dex
```

Finally to get the Java code we will open the jar file in JD-GUI utility.

```
$ jd-gui classes_dex2jar.jar
```



Chapter 3

RELATED WORK

We propose a tracking system that stores and analyses all the user activities and device details such as SMS, Contacts, Call logs, Browser history, GPS locations, etc. We are implementing this module in the Framework Layer so that we can get the application databases and can make use of Android APIs. There are many events which are happening on the mobile. Our aim is to distinguish forensic relevant data and collect those data.

Application Databases and Application Sandboxing

Android applications use SQLite databases as the main data storage. Applications like Contacts, Messages, Instant message services takes use of SQLite to store its data. Both Dalvik and native applications run within the same security environment, contained within the Application Sandbox. Applications get a dedicated part of the filesystem in which they can write private data, including databases and raw files.

All the user activities get logged and stored. We collect user specific data such as Email communications, Call logs, SMS communications, Network specific data such as Browser History, Search keywords, WLAN credentials, Smart phone specific data such as GPS locations, installed Application details, etc. We collect these data on the device and once the data is connected to the Internet data will be uploaded to server in stealthy mode so that the user cannot find out the upload data traffic.

3.1 Android Forensic Support Framework

Android Forensic Support Framework [AFSF] has two modules Data Collection and Data Storage. Until the device gets connected to the Internet, data will be stored on the device.

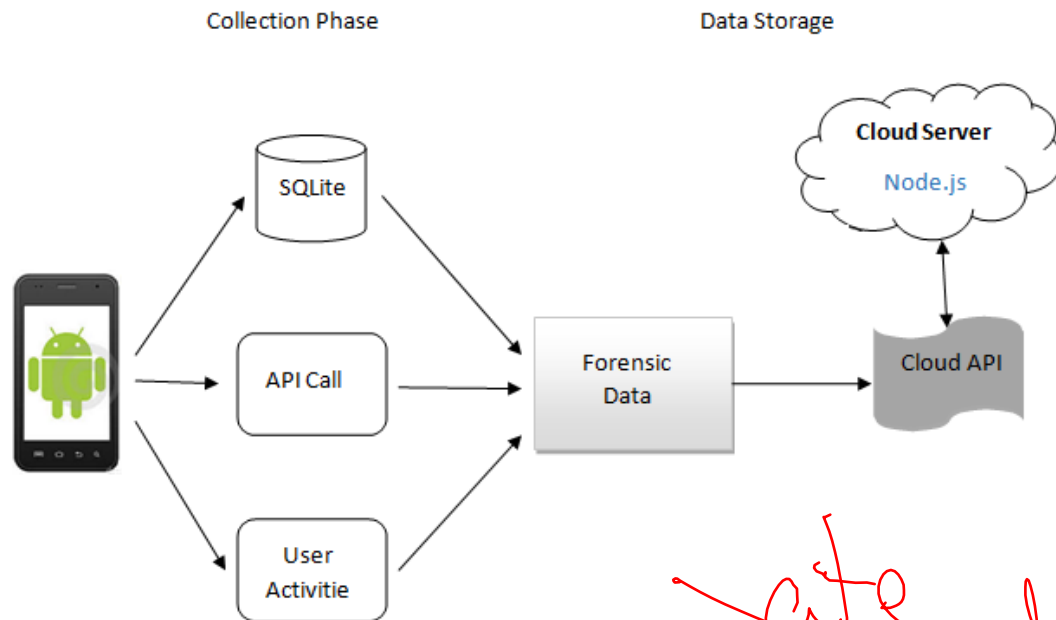


FIGURE 3.1: AFSF: Architecture Diagram

Observing Files

FileObserver : Its an Android API for monitoring file events. It is not recursive. Meaning that it will not track the files or subfolders changes inside a subdirectory.

notify : The command uses the inotify system call to monitor the file/directory changes. Similar to FileObserver it is not recursive.

inotify : It is also a tool to monitor the file changes.

Relevant data:

Critical Data : Phone contacts, Message, Call logs, Network Logs, Chat logs, GPS logs. These type of evidences are potential for investigations and there are hight chances that due to these data the disk space might get exhausted soon. So we categorize these type of data as Critical and we upload to our cloud.

Less critical data: System updates, APK files

According to priority these data are stored inside a xml file and are uploaded to the server. Once when the upload is complete, the data will be deleted

Uploading the evidences: Only when these conditions are met the files in the forensic partition will be uploaded,

- Wifi session is On
- Device is fully charged / While charging
- When the device is idle.

On the other end these data will be identified using the IMEI number to sort relevant data. The collected data gets cleared when they are uploaded to the server. Clearing is done based on priority. If the device does not connect to the internet, the data with less priority gets deleted.

Examination

- From a linux machine we connect to a device using adb
- We image the partition or entire disk using dd
- Extractor tool will image and mount the partition on the Linux machine. This extractor tool uses certain code from the Android Free Forensic Toolkit.

Stealth Challenges

- How to hide the suspicion from the user
- Device Storage
- Internet usage
- Usage of resources

We can hide a process using *hidepid* command. Which is a technique a rootkit generally prefers. *hidepid* accepts three different values:

- hidepid=0 (default): This is the default setting and gives you the default behaviour.
- hidepid=1: With this option an normal user would not see other processes but their own about ps, top etc, but he is still able to see process IDs in /proc
- hidepid=2: Users are only able too see their own processes (like with hidepid=1), but also the other process IDs are hidden for them in /proc!

Conclusion

1. The collected data will be stored encrypted in a separate file volume /forensics and the data will remain until the data is uploaded to the server. This partition is root owned and it cannot be accessed by other applications.
2. The custom inotify and inotifywait(ported, compiled using NDK) monitors the user activities and copies those files into the forensic partition
3. A XML file keeps track of the path names for the copied files.
4. The custom inotify checks if the WiFi session is enabled if yes it uploads the data from the forensic partition.
5. Also a shell script that copies an exact bit by bit copy of the disk for offline analysis of the evidence

3.2 Symantec Smartphone Honey Stick Project

Introduction

In todays world both consumers and corporations need to be concerned about the data stored in their smartphones. It helps individuals to choose appropriate safeguard in terms of policies, procedures, training and technology for employees using mobile devices.

The project was started to help businesses and individuals in understanding the threats to smarphones and their associated information that arise when the smartphone is lost.

For example, opening a corporate email app might immediately give the holder of the phone access to confidential corporate information such as intellectual property, financial plans, bid pricing or personal information about employees. This type of confidentiality breach could cost the employer significantly in lost revenue opportunities or even legal actions.

50 devices were intentionally lost and finders were expected to pickup them up for accessing data on those devices. These events will be logged centrally for performing risk assessment. More over there are no security applications or features enabled in those devices.

1. Each smartphone was loaded with simple apps
2. GPS tracking mechanism was also used to log each phones position occasionally which helps in identifying the whereabouts of the holder.
3. To track unethical access attempt, several apps had a simulated login page with prefilled credentials.
4. The log data from each phone was collected and stored in a database.
5. If a holder tries to contact the owner, this information will also be stored

Key Findings

1. 96 percent of lost smartphones were accessed by the finders of the devices
2. 89 percent of devices were accessed for personal related apps and information
3. 83 percent of devices were accessed for corporate related apps and information
4. 70 percent of devices were accessed for both business and personal related apps and information
5. 50 percent of smartphone finders contacted the owner and provided contact Information

Recommendation

Don't cut-n-paste

Recommendations were given to both consumers and corporations.

For Corporations: Organizations should develop and enforce strong security policies for employees using mobile devices for work; this includes requiring password-enabled screen locks. Mobile device management and mobile security software can aid in this area.

1. Companies should focus on protecting information as opposed to focusing solely on devices, securing information so it is safe no matter where it ends up. Organizations should educate employees about the risks both online and physical associated with mobile devices, such as the impact of a lost or stolen device.
2. Companies should take inventory of the mobile devices connecting to their networks; they cant protect and manage what they dont know about.
3. Businesses should have a formal process in place so everyone knows what to do if a device is lost or stolen. Mobile device management software can help automate such a process.
4. Companies should integrate mobile device security and management into the overall security and management framework and administer it the same way. In essence, treat mobile devices as the true endpoints they are.

For consumers

Smartphone users should use the screen lock feature and make sure it is secured with a strong password or draw to unlock pattern. This is the most basic security precaution and requires minimal effort on the part of the user, yet can provide a critical barrier between personal information and a stranger.

1. Consumers should use security software specifically designed for smartphones. Such tools can stop hackers and prevent cybercriminals from stealing information or spying on users when using public networks. In addition it can often help locate a lost or stolen device and even remotely lock or wipe it.
2. When out and about, users should make sure mobile devices remain nearby and never be left unattended, being mindful of where they put devices at all times. It is also a good idea to make sure they can differentiate their device from others that

might be sitting in the immediate vicinity; adding distinguishing features such as a sticker or case may help.

3.3 Automated Data Collection and Reporting from a mobile device

The work focuses on the application (Droidwatch) which automates and gathers probative information from smartphones and uploads them to an internal server for investigation purpose . A user consent banner will be displayed during the startup or upon launching the Android App. Only upon acceptance of terms and conditions, this particular service will function. When a user rejects the user consent terms the service will not record user activities or send it remotely to a webserver.

The DroidWatch app works by utilizing Android App Components such as *Content observers*, *Broadcast receivers* and *Alarms* to perform its data monitoring and collection. First, data sets (contacts, calendar, should be analyzed to determine if they generate system broadcasts. If they do, broadcast receivers should be considered as the collection component. If broadcasts are not available, consider content observers for implementation. Alarms should be used if broadcasts and content observers are unavailable or ineffective for the targeted data collection.

All logged events were transferred from the phone to the enterprise server were automatically ingested into Splunk, a well-known log indexing system with a Web interface for searching. Results were displayed as individual events, formatted as key-value pairs. All data was associated with a device ID and could be searched, filtered, and put into timelines for identification and trend analysis. Note that while Splunk was used in this research because of its cost (free for up to 500 megabytes per day), other products can perform similar functions.

Chapter 4

PROPOSED WORK

chapter

This section provides an overview of the proposed work

1. Transmitting user activity to the cloud using ICMP (stealthy protocols) protocol

- (a) Encrypting the evidence files
- (b) Compressing the encrypted evidence files

2. Tracking User activities

- (a) Tracking location
- (b) Ei Parameter
- (c) Recovering files using inode table

1. Transmitting user activity

The idea is to transfer the evidence files by injecting them into an echo request packet and sent to a remote cloud i.e the encrypted evidence files/data will be sent over the internet in the ICMP payload section. It is not easy to detect this type of traffic without properly inspecting the packets transmitted through the network. By this way we will achieve stealthy transfer of the evidence files. On the other end we will sniff the ICMP packets and will identify the payload and use them to reconstruct the evidence files.

(a) Compression Technique

Adding the data compression technique helps in reducing the consumption of the disk space and also to transfer files over the internet in a faster manner. A file compression

(b) Encryption

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: icmp Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
313	56.652437	10.30.56.123	10.30.56.101	ICMP	60	Echo (ping) request id=0x6221, seq=4352/17, ttl=64 (reply in 314)
314	56.655084	10.30.56.101	10.30.56.123	ICMP	60	Echo (ping) reply id=0x6221, seq=4352/17, ttl=64 (request in 313)
321	61.279039	10.30.56.123	10.30.56.101	ICMP	622	Echo (ping) request id=0x6221, seq=0, ttl=64 (reply in 322)
322	61.280243	10.30.56.101	10.30.56.123	ICMP	624	Echo (ping) reply id=0x6221, seq=0, ttl=64 (request in 321)
324	63.326038	10.30.56.123	10.30.56.101	ICMP	646	Echo (ping) request id=0x6a21, seq=0, ttl=64 (reply in 325)
325	63.327552	10.30.56.101	10.30.56.123	ICMP	646	Echo (ping) reply id=0x6a21, seq=0, ttl=64 (request in 324)
327	65.375013	10.30.56.123	10.30.56.101	ICMP	642	Echo (ping) request id=0xab21, seq=0, ttl=64 (reply in 328)
328	65.376476	10.30.56.101	10.30.56.123	ICMP	642	Echo (ping) reply id=0xab21, seq=0, ttl=64 (request in 327)
330	67.439530	10.30.56.123	10.30.56.101	ICMP	630	Echo (ping) request id=0xae21, seq=0, ttl=64 (reply in 331)
331	67.440933	10.30.56.101	10.30.56.123	ICMP	630	Echo (ping) reply id=0xae21, seq=0, ttl=64 (request in 330)
365	78.213693	10.30.56.123	10.30.56.101	ICMP	42	Echo (ping) request id=0xb121, seq=0, ttl=64 (reply in 366)
366	78.215206	10.30.56.101	10.30.56.123	ICMP	60	Echo (ping) reply id=0xb121, seq=0, ttl=64 (request in 365)

Frame 322: 622 bytes on wire (4976 bits), 622 bytes captured (4976 bits) on Ethernet II, Src: HewlettPc_42:80:72 (88:51:fb:42:80:72), Dst: HewlettPc_31:2f:a5 (6c:3b:e5:31:2f:a5)

Internet Protocol Version 4, Src: 10.30.56.101 (10.30.56.101), Dst: 10.30.56.123 (10.30.56.123)

Internet Control Message Protocol

```

0000  6c 3b e5 31 2f a5 88 51  fb 42 80 72 08 00 45 00  1:1..Q..B.r..E.
0010  02 60 5c 1c 00 40 01 97  65 09 18 05 00 04 8e  8f...e...8e...
0020  38 7b 00 00 c8 a5 67 21  00 00 2f 39 64 2f 34  8f...q...:/91/4A
0030  41 51 53 60 54 4a 52 61  42 41 51 51 45 41 53 41  8QSKZrJg ABAGQASA
0040  41 41 41 46 41 33 50 45  42 44 46 41 33 50 45 45  8TAAQ/2u B0NAP3P
0050  59 38 4d 63 42 47 51 55  54 61 56 56 46 66 65 4d  Y8N1BQUZ zAvVBfem
0060  69 65 47 35 75 65 63 55  57 76 75 54 48 49 2f 2f  TCGe3ueP wvuzHT
0070  2f 2f 2f 2f 2f 2f 2f 2f  2f 2f 2f 2f 2f 2f 2f 2f  2f 2f 2f 2f 2f 2f 2f
0080  2f 2f 2f 2f 2f 2f 2f 2f  2f 2f 2f 2f 2f 2f 2f 2f  2f 2f 2f 2f 2f 2f 2f
0090  2f 2f 2f 2f 2f 2f 2f 2f  2f 2f 2f 2f 2f 2f 2f 2f  2f 2f 2f 2f 2f 2f 2f
0100  2f 2f 41 4c 43 41 41 66  46 46 51 42 41 52 41 52  WAALCA ANAFQBAR
0110  45 41 2f 38 51 41 47 41  41 42 41 51 45 42 41 51  EA/8QAGA ABAGBAEQ
0120  41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 41  AAAAAAAAAA AAAAAAAA
0130  49 42 41 77 2f 78 78 41  41 6f 41 41 41 41 67 41  EBWNT/AA A0EAACAG
0140  49 43 41 51 40 43 42 77  41 41 41 41 41 41 41 41  ICACQ/CBw AAAAAAAA
0150  41 42 41 67 41 52 41 7a  45 33 49 52 4d 45 49 6c  ABAGAQCB ESTIME1T
0160  45 79 51 54 46 68 63 58  48 52 6f 64 48 2f 32 6f  EY0TNRKX KROGH/2g
0170  41 49 41 51 45 41 41 44  38 41 38 55 52 45 52 45  AIAQEAAD SABURERE
  
```

File: C:\Users\h1d430rhu\Desktop\icmp.pcap Packets: 1192 - Displayed: 64 (5.4%) - Load time: 0.00090

Profile: Default

FIGURE 4.1: Transferring files using ICMP protocol

2. Tracking User activities

(a) Tracking location

Technologies such as Global Positioning System (GPS) based data can also aid investigators by tracking the whereabouts of a suspect. GPS co-ordinates can be obtained by using the available APIs. If the GPS module fails to locate the current location we will try to at least find the recent location of the user by using two techniques. First one would be to get the location details from a Weather App(Weather & Clock Widget Android) database which would store the GPS location.

If the 1st solution fails to locate the current location we will identify at least the country location by extracting information from a http response from a web server. For example, a user sends a request to google.com from India. He will be receiving a response from google.co.in domain. If the same user accesses the google.com from Italy the response would be from google.it. That is, we will identify country location based on domain names. For instance, a daemon will periodically send http request to google.com and if we are in a different country we will get a different domain.

(b)EI parameter

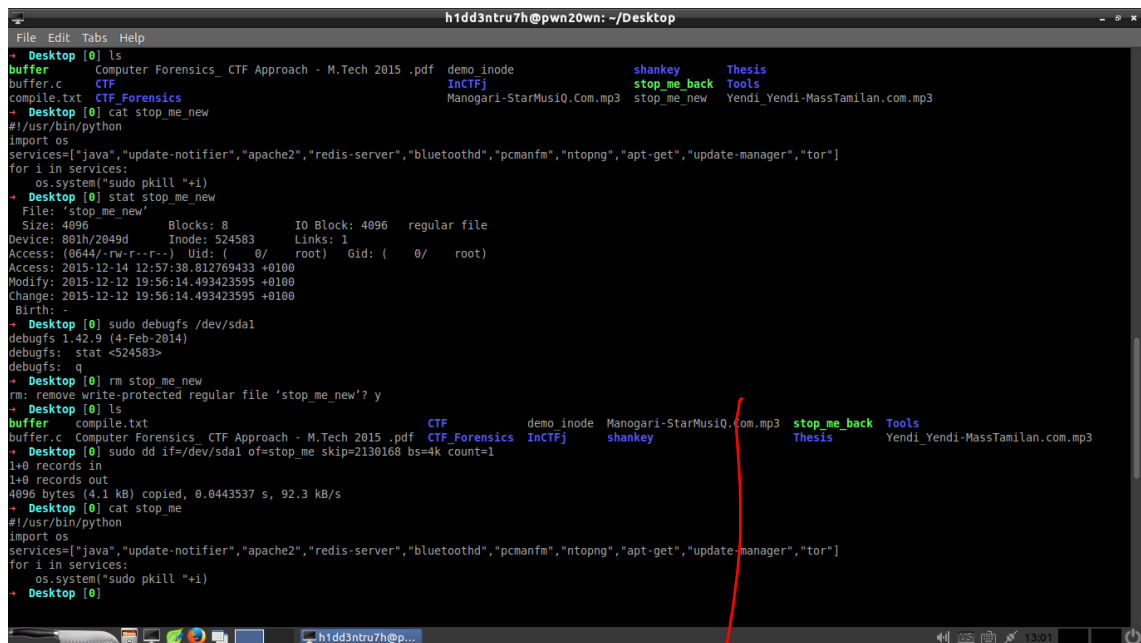
By parsing the Google Search URLs stored in the browsers SQLite database file we will use the ei parameter that contains an encoded Unix timestamp. By decoding the value contained in the ei parameter we would be able to find the time at which a Google search was made.

(c)Recovering files using Inode Table Using inode tables we will try to record the location for all the files in a particular directory of interest, likely in a place where the number files are less. For example, a file was intentionally deleted by the user from /Pictures directory. Upon deleting this file it is possible to recover them immediately using the information we recorded from the inode table. We can use inotify to monitor the user activity (here "delete"). To demonstrate,

1. We can find the inode number of a file using *stat* or *ls -li*
`$ stat hello.txt`
2. After finding the inode number now will determine the location of the file in our disk using debugfs utility. We will assume the inode number for hello.txt is 524464.
`$ sudo debugfs /dev/sda1`
`(debugfs) stat 524464;`
`EXTENTS: (0):2130005`
3. The location of the hello.txt is found, now will go ahead and delete the hellot.txt file and will recover them in the next step.
`$ rm hello.txt`
4. Now it is possible to recover the deleted file using the dd utility,
`$ sudo dd if=/dev/sda1 of=hello_Recovered.txt skip=2130005 bs=4k count=1`

Very general
 Concept

Limitation : *Recovery is not possible when this location is overwritten by a new data.*



```
h1dd3ntru7h@pwn20wn: ~/Desktop
+ Desktop [0] ls
+ Desktop [0] cat stop_me_new
+ Desktop [0] stat stop_me_new
+ Desktop [0] sudo debugfs /dev/sda1
+ Desktop [0] rm stop_me_new
+ Desktop [0] ls
+ Desktop [0] sudo dd if=/dev/sda1 of=stop_me skip=2130168 bs=4k count=1
+ Desktop [0] cat stop_me
+ Desktop [0]
+ Desktop [0] sudo pkill "+i)
```

The screenshot shows a terminal window with a dark background and light-colored text. The user is performing various file operations and using the 'stat' command to view file metadata. A red line is drawn from the 'stop_me' file in the terminal output to the caption below.

FIGURE 4.2: Recovering Deleted files using inode table: Screenshot

replace with a
typescript
capture

Chapter 5

CONCLUSION AND FUTURE WORK

Forensic analysis of Smartphone is ~~extremely~~ useful during the course of many investigations. Mobile devices have a huge collection of personal data that can help the investigators to track the activities of the suspect. These data include the call history, location history, messages, browser history, search URLs etc. The data in the device can be used to fingerprint the individual.

In addition to the previous work, we proposed techniques for stealthy transfer of the evidence files and also few techniques to collect user activities such as tracking location and finding the google search query time by parsing the URLs obtained from the browser SQLite files. We have also provided a solution to quickly recover the deleted files using the inode table.

As part of the future work the proposed techniques will be implemented and incorporated into our framework.

Bibliography

- [1] Jeff Lessard, Gary C. Kessler *Android Forensics: Simplifying Cell Phone Examination*. SMALL SCALE DIGITAL DEVICE FORENSICS JOURNAL, 2010
- [2] Alessandro Distefano, Gianluigi Me, Francesco Pace *Android anti-forensics through a local paradigm* Digital Investigation 7, 2010, S83-94
- [3] Laurent Simon, Ross Anderson *Security Analysis of Android Factory Resets* IEEE Security, 2015
- [4] Laurent Simon, Ross Anderson *Security analysis of consumer-grade Anti-Theft Solutions Provided by Android Mobile Antivirus Apps* IEEE Security, 2015
- [5] Mahajan, Aditya and Dahiya, MS and Sanghvi, HP *Forensic Analysis of Instant Messenger Applications on Android Devices* arXiv preprint, arXiv:1304.4915 2013
- [6] Wright, Scott *The Symantec smartphone honey stick project 2012*
- [7] Grover, Justin *Android forensics: Automated data collection and reporting from a mobile device* Digital Investigation, 2013

pretty done
why not?

example

example

Aditya Mahajan?