

ANDROID MALWARE TRACEABILITY MATRIX FOR DIGITAL FORENSIC
INVESTIGATION



A' AISYAH MARDHIYYAH BINTI MOHAMMAD SHAHINI

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

BORANG PENGESAHAN STATUS

JUDUL: ANDROID MALWARE TRACEABILITY MATRIX FOR DIGITAL FORENSIC INVESTIGATION

SESI PENGAJIAN: 2016/2017

Saya A' AISYAH MARDHIYYAH BINTI MOHAMMAD SHAHINI mengaku membenarkan tesis Projek Sarjana Muda ini disimpan di Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dengan syarat-syarat kegunaan seperti berikut:

Tesis dan projek adalah hakmilik Universiti Teknikal Malaysia Melaka.

Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan untuk tujuan pengajian sahaja.

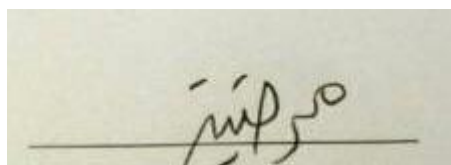
Perpustakaan Fakulti Teknologi Maklumat dan Komunikasi dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.

** Sila tandakan (/)

_____ SULIT (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

_____ TERHAD (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

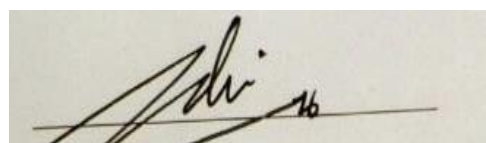
_____ TIDAK TERHAD



(TANDATANGAN PENULIS)

Alamat tetap 265A, KM 10,
Jalan Simpang Empat,
06650 Alor Setar, Kedah

Tarikh: 21/8/2017



(TANDATANGAN PENYELIA)

ENCIK MOHD ZAKI MAS'UD
NAMA PENYELIA

Tarikh: 21/8/2017

ANDROID MALWARE TRACEABILITY MATRIX FOR DIGITAL FORENSIC INVESTIGATION

A' AISYAH MARDHIYYAH BINTI MOHAMMAD SHAHINI



This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science (Computer Security)

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

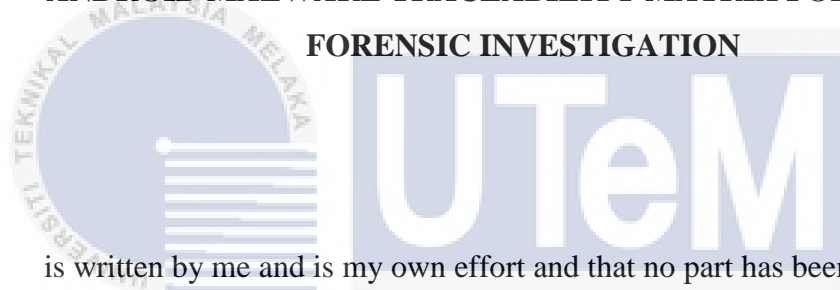
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2017

DECLARATION

I hereby declare that this project report entitled

ANDROID MALWARE TRACEABILITY MATRIX FOR DIGITAL FORENSIC INVESTIGATION



is written by me and is my own effort and that no part has been plagiarized
without citations.

اونيورسي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

STUDENT:

A' AISYAH MARDHIYYAH BINTI MOHAMMAD SHAHINI

Date: 21/8/2017

SUPERVISOR:

ENCIK MOHD. ZAKI BIN MAS'UD

Date: 21/8/2017

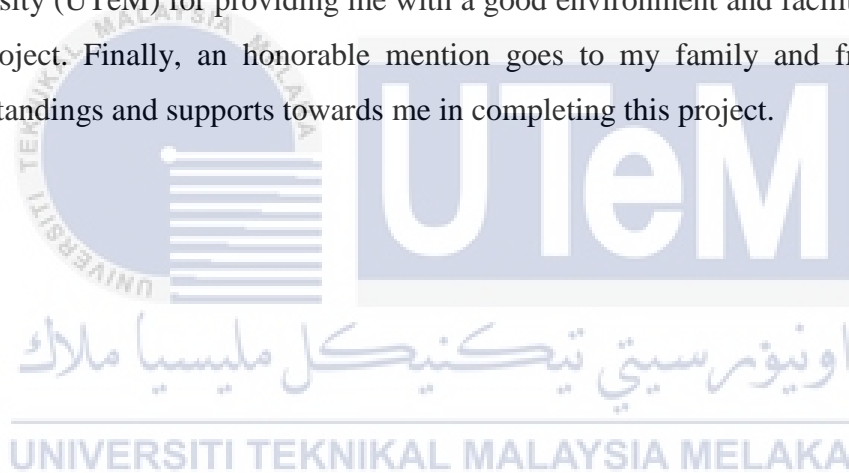
DEDICATION

To my beloved parents, Ir. Mohammad Shahini Puteh and Puan Bushra Abd Rahman,
who are my greatest inspiration in education.



ACKNOWLEDGEMENTS

Firstly, I would like to thank to my super supportive supervisor of this project, Encik Mohd. Zaki Bin Mas'ud for the valuable guidance and advice. He inspired me greatly to work in this project. His willingness to motivate us contributed tremendously to my project. I also would like to thank him for showing me some example that related to the topic of this project. Besides, I would like to thank the authority of Technical Malaysia University (UTeM) for providing me with a good environment and facilities to complete this project. Finally, an honorable mention goes to my family and friends for their understandings and supports towards me in completing this project.



ABSTRACT

The objective of digital forensic investigation process in a cybercrime is to preserve any evidence in its most original form while performing a structured investigation by collecting, identifying and validating the digital information for the purpose of reconstructing past events while maintaining the chain of custody. With the current scenario, mobile technology has also exposes to cybercrime, this has make the investigation process more complex. The traceability process has become a crucial part of the digital investigation process because it is capable to map the events of an incident from different sources in collecting evidence of an incident to be used for other additional investigation aspects. The need of finding and mapping evidence in Android platform has also becoming more important. Thus, this project proposes the adaptability of the traceability matrix to represent the relationship in the digital forensic investigation process by assimilating the traceability features in the mobile technology environment especially on Android. The objective of this project is to identify, analyze and construct Android malware traces for forensic investigation and show the link between the evidence, the entities and the sources related in the process. Besides, the proposed project is expected to assist the forensic investigator in gaining accurate and complete evidence that can be further used in a court of law. To make it real, there are four phases that had been conducted. The first phase was a literature review where a detailed study of the traceability issues that involved to mobile forensic. The second phase was analyzing data set while the third phase was a construction of the traceability matrix and the last phase was completing the documentation.

ABSTRAK

Objektif proses penyiasatan forensik digital dalam jenayah siber adalah untuk memelihara mana-mana bukti dalam bentuk yang paling asal semasa menjalankan penyiasatan yang tersusun dengan mengumpul, mengenal pasti dan mengesahkan maklumat yang digital untuk tujuan membina semula kejadian yang lepas di samping mengekalkan rantaian jagaan. Dengan senario semasa, teknologi mudah alih juga telah memberi pendedahan kepada jenayah siber, hal ini menjadikan proses siasatan lebih kompleks. Proses pengesanan telah menjadi sebahagian yang penting dalam proses siasatan digital kerana ia mampu untuk memetakan peristiwa kejadian daripada pelbagai sumber dalam mengumpul bukti kejadian yang akan digunakan untuk lain aspek siasatan tambahan. Keperluan untuk mencari dan memeta bukti dalam platform Android juga telah menjadi lebih penting. Oleh itu, projek ini mencadangkan penyesuaian matriks pengesanan untuk mewakili hubungan dalam proses siasatan forensik digital dengan menerapkan ciri-ciri pengesanan dalam persekitaran teknologi mudah alih terutamanya pada Android. Objektif projek ini adalah untuk mengenal pasti, menganalisis dan membina jejak malware Android untuk siasatan forensik dan menunjukkan hubungan antara bukti, entiti dan sumber yang berkaitan dalam proses. Selain itu, projek yang dicadangkan itu dijangka membantu penyiasat forensik dalam mendapatkan bukti yang tepat dan lengkap yang boleh terus digunakan dalam mahkamah undang-undang. Untuk membuatnya berhasil, terdapat empat fasa yang telah dijalankan. Fasa pertama adalah kajian yang lebih dalam di mana satu kajian terperinci mengenai isu-isu kebolehsasaran yang terlibat untuk forensik mudah alih. Fasa kedua telah menganalisis data yang ditetapkan manakala fasa ketiga adalah pembinaan matriks pengesanan dan fasa terakhir melengkapkan dokumentasi.

TABLE OF CONTENTS

CHAPTER	SUBJECT	PAGE
	DECLARATION	i
	DEDICATION	ii
	ACKNOWLEDGEMENT	iii
	ABSTRACT	iv
	ABSTRAK	v
	TABLE OF CONTENTS	vi
	LIST OF TABLES	ix
	LIST OF FIGURES	X
CHAPTER 1	INTRODUCTION	
	1.1 Introduction	1
	1.2 Problem Statement	3
	1.3 Project Question	4
	1.4 Project Objective	4
	1.5 Project Scope	6
	1.6 Project Contribution	6
	1.7 Thesis Organization	7
	1.8 Conclusion	8
CHAPTER II	LITERATURE REVIEW	
	2.1 Introduction	9
	2.2 Related Work	11
	2.3 Critical Review	22
	2.4 Proposed Solution	25
	2.5 Conclusion	25

CHAPTER III	METHODOLOGY	
	3.1 Introduction	26
	3.2 Methodology	27
	3.3 Project Milestone	29
	3.4 Conclusion	32
CHAPTER IV	DESIGN	
	4.1 Introduction	33
	4.2 Requirements	34
	4.3 Analysis Approach	36
	4.4 Network Design	38
	4.5 Traceability Matrix	39
	Design	
	4.6 Conclusion	40
CHAPTER V	IMPLEMENTATION AND ANALYSIS	
	5.1 Introduction	41
	5.2 Environment Set Up	42
	5.3 Process of Collecting Traces	43
	5.4 Conclusion	62
CHAPTER VI	TESTING	
	6.1 Introduction	63
	6.2 Test Organization	64
	6.3 Test Design	64
	6.4 Test Result	65
	6.5 Conclusion	68
CHAPTER VII	PROJECT CONCLUSION	
	7.1 Introduction	69
	7.2 Project Summarization	69

7.3 Project Contribution	70
7.4 Project Limitation	71
7.5 Future Works	71
7.6 Conclusion	71
REFERENCES	72
APPENDICES	73



LIST OF TABLES

TABLE	TITLE	PAGE
1.1	Summary of Research Problem	3
1.2	Summary of Research Question	4
1.3	Summary of Project Objective	5
1.4	Summary of Research Contribution	6
2.1	List of Existing Research (Traceability Matrix)	11
2.2	List of “Traceability” Definition	14
2.3	List of Android Behaviour	15
2.4	Android Malware and Their Definition and Behaviour	17
2.5	List of Type of Malware	18
2.6	List of network traffic features	21
2.7	List of System Call	21
2.8	List of Existing Research (Traceability Matrix)	22
3.1	Project Milestones	29
4.1	Details of Hardware Requirement	36
4.2	Home Page	40
5.1	List of Traces in Network Traffic	48
5.2	List of Parameter in System Call	59
5.3	Digit and Type of File Permission	53
5.4	List of Traces in Logcat	59
6.1	Result from Network Traffic	66
6.2	Result from System Call	67
6.3	Result from Logcat	68

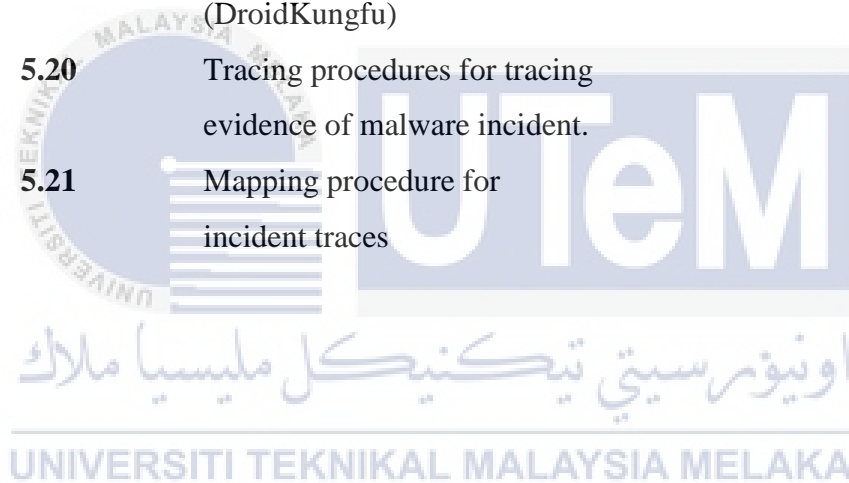
LIST OF FIGURES

FIGURE	TITLE	PAGE
1.1(a)	Worldwide Smartphone OS Market Share (Share in Unit Shipments)	2
1.1(b)	Worldwide Smartphone OS Market Share (Share in Unit Shipments) (IDC, 2016)	2
1.2	The growth of android mobile malware variants. (Symantec Corp, 2016)	2
2.1	Outline of Android Malware Traceability Matrix for Digital Forensic Investigation.	10
3.1	Project Methodology	27
3.2	Literature Review	27
3.3	Analysis and Design	28
4.1	Specification of The Workstation	34
4.2	Genymotion Interface (i)	35
4.3	Genymotion Interface (ii)	36
4.4	Analysis Design	37
4.5	Physical Design	38
4.6	Logical Design	39

5.1	Steps on Environment Set Up	42
5.2	Steps of Collecting Data from Network Traffic	44
5.3	Request Method	44
5.4	Follow TCP Stream	44
5.5	Domain Name	45
5.6 (a)	Domain Name (AnserverBot)	46
5.6 (b)	Domain Name (DroidKungfu)	46
5.6 (c)	Domain Name (DroidDream)	47
5.7	Steps of Collecting Data from System Call	49
5.8 (a)	Gain Access to a File (AnserverBot)	49
5.8 (b)	Gain Access to a File (DroidKungfu)	50
5.8 (c)	Gain Access to a File (DroidDream)	50
5.9 (a)	Open a File (AnserverBot)	50
5.9 (b)	Open a File (DroidKungfu)	51
5.9 (c)	Open a File (DroidDream)	51
5.10(a)	Get Information of a File (AnserverBot)	51
5.10(b)	Get Information of a File (Droid Kungfu)	52
5.10(c)	Get Information of a File (DroidDream)	52

5.11(a)	Change Mode of the System (AnserverBot)	53
5.11(b)	Change Mode of the System (Droid Kungfu)	53
5.11(c)	Change Mode of the System(DroidDream)	54
5.12(a)	Write to the File (AnserverBot)	54
5.12(b)	Write to the File (DroidKungfu)	55
5.12(c)	Write to the File(DroidDream)	55
5.13(a)	Rename the file (AnserverBot)	55
5.13(b)	Rename the file (DroidDream)	56
5.14(a)	Read the file (AnserverBot)	56
5.14(b)	Figure 5.14(b) Read the file (DroidDream)	56
5.15(a)	Receive a message from a socket (AnserverBot)	57
5.15(b)	Receive a message from a socket (DroidKungfu)	57
5.15(c)	Receive a message from a socket (DroidDream)	57
5.16	Execute the File	57
5.17(a)	Delete the File (AnserverBot)	58

5.17(b)	Delete the File (DroidDream)	58
5.17(c)	Delete the File (DroidKungfu)	58
5.18	Steps of Collecting Data from Logcat	60
5.19(a)	Priority, Message and PID (DroidDream)	60
5.19(b)	Priority, Message and PID (AnserverBot)	60
5.19(c)	Priority, Message and PID (DroidKungfu)	61
5.20	Tracing procedures for tracing evidence of malware incident.	61
5.21	Mapping procedure for incident traces	62



CHAPTER I

INTRODUCTION



اونيورسيتي تيكنيكل مليسيا ملاك

1.1 Introduction

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Previously, malware has infected PCs for many years. Due to the fast growth of smartphone industry, it has made mobile platforms a prime target for malware developers to perform attacks. According to a study made by International Data Corporation in third quarter of 2016, Android dominated the smartphone market with a share of 86.8% as shown in **Figure 1.1(a)** and **Figure 1.1(b)**. Besides, according to an internet security threat report made by Symantec Corporation, from 2013 until 2015, the number of android mobile malware variant increases as shown in **Figure 1.2**.

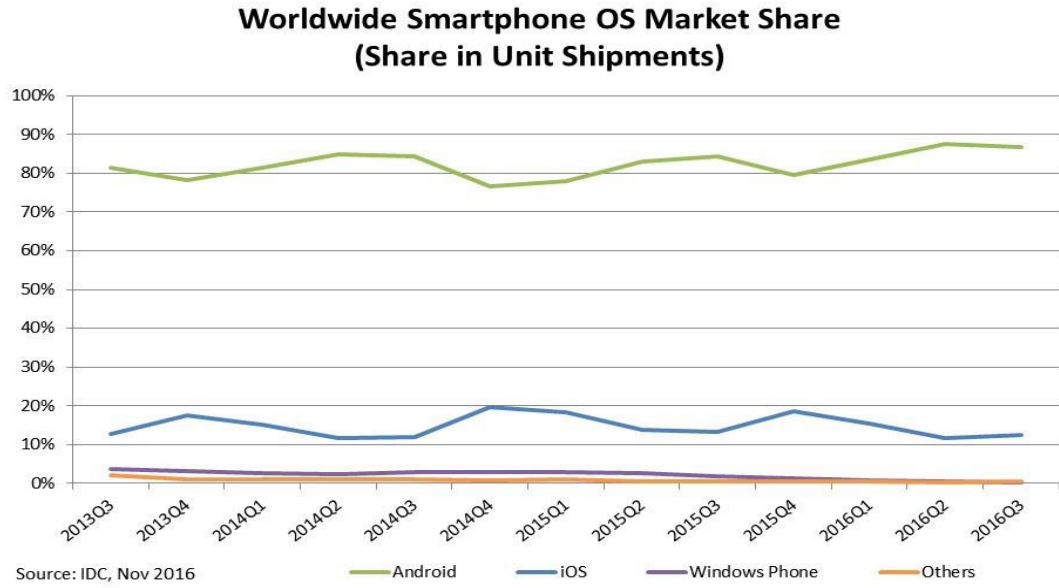


Figure 1.1 (a) Worldwide Smartphone OS Market Share (Share in Unit Shipments)

Period	Android	iOS	Windows Phone	Others
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

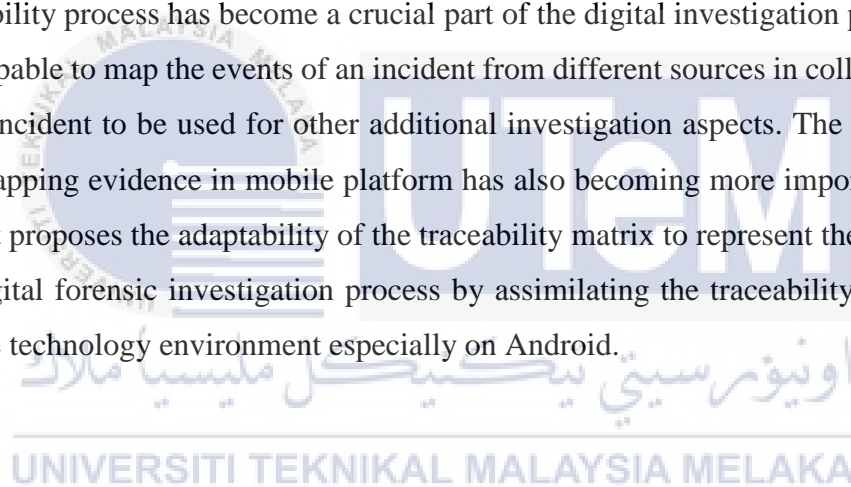
Source: IDC, Nov 2016

**Figure 1.1 (b) Worldwide Smartphone OS Market Share (Share in Unit Shipments)
(IDC, 2016)**



Figure 1.2: The growth of android mobile malware variants. (Symantec Corp, 2016)

Unfortunately, Android application developers can upload their applications without any verification or validation of their trustworthiness. Unauthorized repositories also exist, where developers can upload applications, including cracked applications. This has allowed attackers to upload malware to the market and also to spread malware through the unauthorized repositories. A number of applications have been modified and the malware have been bind, packed and spread through unauthorized repositories. Therefore, detection of Android malware efficiently has great significance and definitely Android malware analysis becomes more and more common task during mobile forensic investigations. Every month thousands of new malware types are created, so it becomes critical for any digital forensic investigator to have accurate and complete evidence. The traceability process has become a crucial part of the digital investigation process because it is capable to map the events of an incident from different sources in collecting evidence of an incident to be used for other additional investigation aspects. The need of finding and mapping evidence in mobile platform has also becoming more important. Thus, this project proposes the adaptability of the traceability matrix to represent the relationship in the digital forensic investigation process by assimilating the traceability features in the mobile technology environment especially on Android.



1.2 Problem Statement

The rapid growth of Android malware is posing challenges to malware detection technology. The Research Problem (RP) is summarized into **Table 1.1**.

Table 1.1 Summary of Research Problem

	Research Problem
RP ₁	Difficulty to investigate the android malware incidence.

1.3 Project Question

Therefore, two Research Questions (RQ) are constructed to identify the research problem as discussed in previous section is depicted in **Table 1.2**.

Table 1.2 Summary of Research Question

	RQ	Research Question
RP ₁	RQ ₁	What is the parameter used to trace android malware activity?
	RQ ₂	Where the traces of android malware activity can be found?

RQ1: What is the parameter used to trace android malware activity?

This research question is to find out the suitable parameter to be used to trace android malware activity. It is important to determine which parameter to be used as each type of malware infect on different parameter.

RQ2: Where the traces of android malware activity can be found?

This research question is to find out where the traces of android malware activity can be found and the steps used to find the malicious activities.

1.4 Project Objective

From the research problem and question, the project objective has been determined. The project objective is depicted in **Table 1.3**.

Table 1.3 Summary of Project Objective

RP	RQ	RO	Project Objective
RP ₁	RQ ₁	RO ₁	To investigate traces of android malware activity.
	RQ ₂	RO ₂	To generate the android malware traceability matrix for digital forensic investigation.

RO1: To investigate traces of android malware activity.

It is important to study on traces of android malware activity first before generating the android malware traceability matrix. To start an analysis, identifying parameter becomes a crucial part. As android malware might behave in different way due to its purpose, thus, a lot parameter might involve in the analysis.

RO2: To generate the android malware traceability matrix for digital forensic investigation.

After the parameter used to analyze the malware is determined, thus the next step is to collect data and analyze the data to generate the android malware traceability matrix.

1.5 Project Scope

The scopes of this project are as follow:

- a. This project only investigates on android malware.
- b. This project only focuses on investigating the log cat, network traffic and system call on data sources to trace android malware malicious activity.

1.6 Project Contribution

The research contributions of this project are summarized in **Table 1.4**.

Table 1.4 Summary of Research Contribution

RP	RQ	RO	RC	Project Contribution
RP ₁	RQ ₁	RO ₁	RC ₁	Proposed the parameter used to trace the android malware activity.
	RQ ₂	RO ₂	RC ₂	Proposed the android malware traceability matrix for digital forensic investigation.

By using the finding and result in this project, a forensic investigator will be able to gain accurate and complete evidence that can be further used in a court of law.

1.7 Thesis Organization

This report consists of six chapter namely Chapter I: Introduction, Chapter 2: Literature Review, Chapter 3: Methodology, Chapter 4: Design and Implementation, Chapter 5: Testing and Result Analysis and Chapter 6: Conclusion.

Chapter 1: Introduction.

This chapter will discuss about introduction, project background, research problem, research question, research objective, scope, project significant and report organization.

Chapter 2: Literature Review.

This chapter will explain related work of this project, such system parameter and traceability.

Chapter 3: Methodology.

This chapter will explain the method used to analyze the android malware and organize the sequence of project work in phase by phase.

Chapter 4: Design and Implementation.

This chapter will introduce the software and hardware use in this project, environment setup, implementation of malware as well as the data collected.

Chapter 5: Testing and Result Analysis.

This chapter will analyze the collected data.

Chapter 6: Conclusion.

This chapter will be summarized all chapters as a conclusion.

1.8 Conclusion

In this chapter, problem statement, project questions and project objective of the projects have been clearly identified as well as the plan to conduct the analysis. The next chapter, Chapter 2 will be discussed on the types of android, malware, techniques and parameter used to conduct the analysis of android malware activity.



CHAPTER II

LITERATURE REVIEW

2.1 Introduction

In this chapter, a literature review about Android malware will be discussed as shown in **Figure 2.1**.



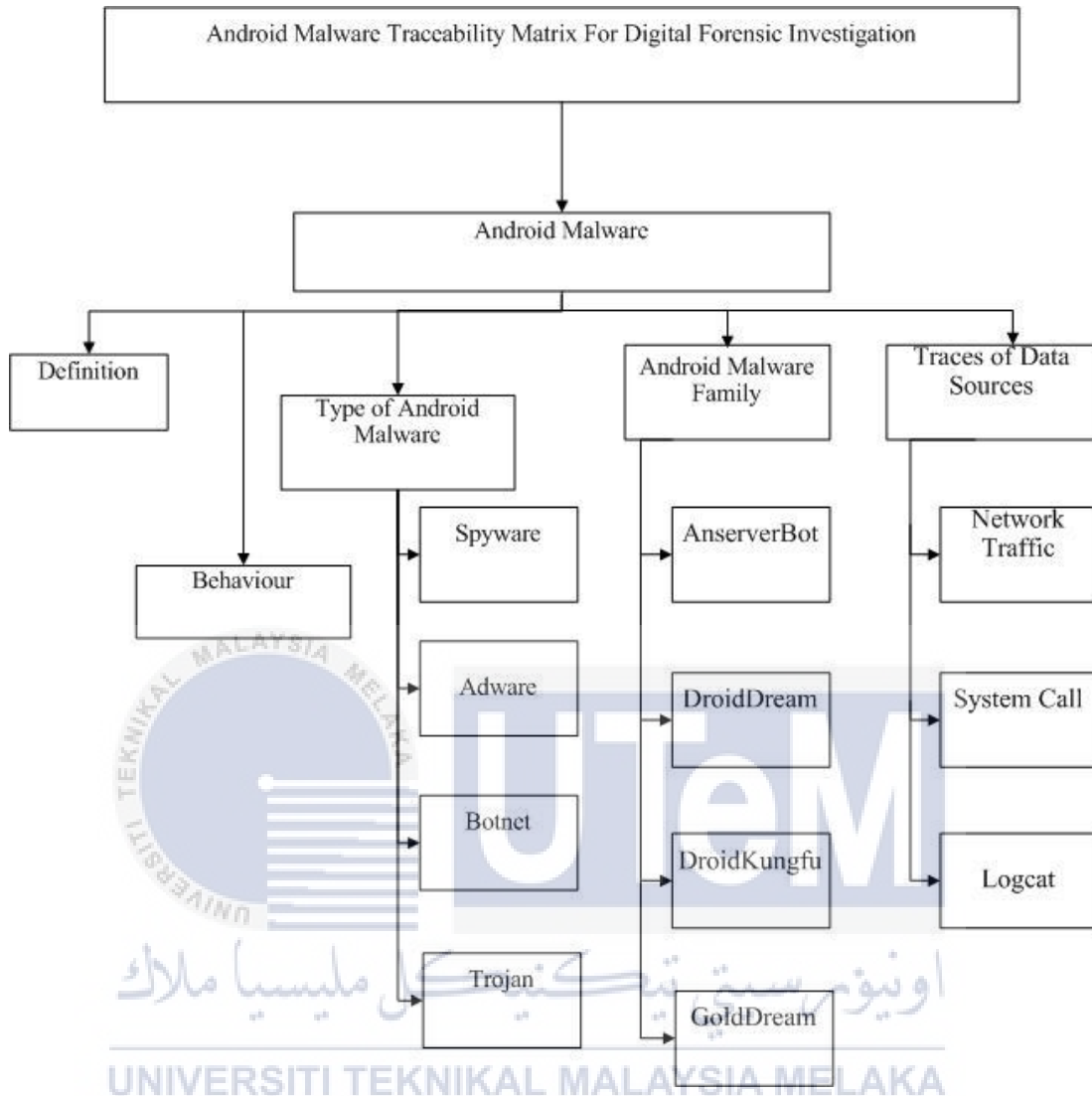


Figure 2.1 Outline of Android Malware Traceability Matrix for Digital Forensic Investigation

2.2 Related Work

Analysis and forensics of malware behavior is an essential technology that extracts the runtime behavior of malware and supplies signatures to detection systems and provides evidence for recovery, cleanup and forensics (Wu et al., 2016.). The objective of digital forensic investigation process in a cybercrime is to preserve any evidence in its most original form while performing a structured investigation by collecting, identifying and validating the digital information for the purpose of reconstructing past events while maintaining the chain of custody. With the current scenario, mobile technology has also exposes to cybercrime, this has make the investigation process more complex. The traceability process has become a crucial part of the digital investigation process because it is capable to map the events of an incident from different sources in collecting evidence of an incident to be used for other additional investigation aspects. The need of finding and mapping evidence in Android platform has also becoming more important.

In recent years, traceability matrix had been widely used in food industry, software testing, military hardware, aerospace and automotive industry as summarized in **Table 2.1**.

Table 2.1 List of Existing Research (Traceability Matrix)

No	Field	Author	Approach
1	Military hardware	(Department Of Defense Handbook System Requirements Document Guidance, 2011)	Defines system or subsystem level functional and performance requirements derived from warfighter capabilities documents, Concept of Operations (CONOPS), Concept of Employment (CoE), Analysis of Alternatives (AoA), system level performance metrics, mission threads or use cases, and usage environment, which are

				<p>captured in a program's System Requirement Document. These aggregate requirements are also now known as attributes.</p> <p>Attributes are further broken out and prioritized as Key Performance Parameters (KPPs) and Key System Attributes (KSAs). Traceability matrix traces System Requirement Development requirements up to warfighter ability documents and down to the lowest level hardware or software component that implements the requirement.</p>
2	Software testing		(Sachdeva,2013)	<p>Defines use case ID or requirement ID, use case or requirement description and one column for each test case.</p>
3	Food Ordering		(Austin, 2013)	<p>Defines system, subsystem level functional, component level and verification requirement derived from static and time-based structure. Static structure consists of component-based and people-based while time-based consists of activity-based and parameter-based structure.</p>

4	Animal Health			<p>Defines the system chosen for animal identification and traceability include outcomes of risk assessment, the animal and public health situation, animal population parameters (for example, species, numbers, and distribution), types of production, animal movement patterns, available technologies, trade in animals and animal products, cost/benefit analysis and other economic, geographical and environmental considerations, and cultural aspects.</p>
5	Aerospace		(Grogan, et al, 2005)	<p>These need to be balanced with other typical parameters for instrument accommodation such as data rate and volume requirements, pointing and stability requirements, mounting and structure requirements and thermal, power, mass and volume constraints.</p>

Table 2.1 shows several field used traceability matrix as a medium to track and map the origin. Since digital forensic investigation requires a lot of evidences to identify the offender or the origin of the crime, traceability matrix can ensure that all the evidence is fully covered.

Technically, traceability was defined and discovered as shown in **Table 2.2**:

Table 2.2 List of “Traceability” Definition

Author	Definition
(Oghazi, P., B. et al., 2007)	The means to identify and follow real or imaginary objects through a process chain.
ISO 8402:1995	The ability to trace the history, application or location of an entity, by means of recorded identifications.
(Golan, E., et al., 2004)	The definition of traceability can be broad, because in most of the time the processes are very complex.
(Clayton, R, 2005)	The ability to map events in cyberspace, particularly on the Internet, back to real-world instigators, often with a view to holding them accountable for their actions.
(Lázaro, P.G.-C., 2004)	How difficult it is to establish the source and destination of communications on computers and communication networks, such as the Internet.
(Selamat, Yusof, et al., 2011)	The ability to trace and map the events of an incident from difference sources in order to obtain evidence of an incident for further process of investigation.

2.2.1 Android Malware

a) Definition

In recent years, Android-based mobile devices are undoubtedly the most popular operating system for smartphones and tablets. Meantime, the dramatic increasing of the number of Android malware being more sophisticated as reported in F-secure Threat Report 2015(Rover, 2015). According to *Malware Fighting Malicious Code* (Skoudis, Ed Zeltser, 2004), malware is defined as a set of instruction that run on your computer and make your system do something that an attacker wants it to do. On the other hand, Android

malware basically means a set of instructions that run on your Android-based mobile devices and make your system do something that an attacker wants it to do.

b) Behaviour

According to a survey of Android Malware (Jiang and Zhou, 2013), Android malware was categorized into four behaviours as shown in **Table 2.3**:

Table 2.3 List of Android Behaviour

Android Behaviour	Attack Techniques
<p>Malware Installation</p>	<ol style="list-style-type: none"> 1. Repackaging <ul style="list-style-type: none"> - One of the most common techniques malware authors use to piggyback malicious payloads into popular apps. 2. Update Attack <ul style="list-style-type: none"> - It may still repackage popular apps but it only includes an update component that will fetch or download the malicious payloads at runtime. 3. Drive-by-download <ul style="list-style-type: none"> - Enticing users to download “interesting” of “feature-rich” apps. 4. Spyware <ul style="list-style-type: none"> - It intends to be installed to victim’s phones on purpose. 5. Fake apps <ul style="list-style-type: none"> - Send SMS messages to premium-rate numbers without user awareness.

	<p>6. Apps that also intentionally include malicious functionality.</p> <ul style="list-style-type: none"> - Provide the functionality they claimed but unknown to users, they also include certain malicious functionality. <p>7. Apps that rely on the root privilege to function well.</p> <ul style="list-style-type: none"> - Grant the root privilege to these apps without asking the user.
<p>Activation</p>	<p>The application activates itself for doing malicious activities. The authors found in their analysis that the malicious applications activate themselves during boot completion events, package removing or package installation events, SMS received events, call phone, during network connectivity events, during system events and during the launch of some popular host applications main activity.</p>
<p>Malicious Payloads</p>	<ol style="list-style-type: none"> 1. Privilege Escalation <ul style="list-style-type: none"> - Malicious application tries to take root access to exploit the root privilege. 2. Remote Control <ul style="list-style-type: none"> - Controlling it from a remote device. 3. Financial Charges <ul style="list-style-type: none"> - The malicious application uses victim's device for sending premium messages

	<p>without user's knowledge in order to harm him with financial loss.</p> <p>4. Personal Information</p> <ul style="list-style-type: none"> - Collecting user's information like SMS received, phone call log, and device information
Permission Usage	Android requires applications to declare all the required permissions, and Android will prompt declared permissions to users at the beginning of an installation process (Xu, Zhang, et al., n.d.).



اونيورسيتي تيكنيكل مليسيا ملاك

c) Type of Android Malware

Basically, there are four types of most popular Android malware as summarized in **Table 2.4**.

Table 2.4 Android Malware and Their Definition and Behaviour

Android Malware	Definition and Behaviour
Spyware	Software that reveals private information about the user or computer system to eavesdroppers (Hypponen, 2006). A malware that spies the user's activities (Abualola, Alhawai, et al., 2016).

Adware	Advertising a product or a website that are harmless but annoying (Feizollah, Anuar, et al., 2015).
Botnet	Collection of several bots connected with each other's with the help of networks. Mobile botnets provide botmaster with root permissions over the compromised mobile device, enabling botmaster to send mails or text messages, make phone calls, access contacts and photos (Joshi, Khanna, et al., n.d.).
Trojan	A software that appears to provide some functionalities but, instead, contains a malicious program (Polla, Martinelli, et al., 2013).

According to the current attack trends and analysis of the present literatures, (Raveendranath, Venkiteswaran, n.d.) described the types of malwares as shown in **Table 2.5** :

Table 2.5 List of Type of Malware

Type of Malware	Description
Information Extraction	Compromises the devices and steals personal information such as personal information, IMEI number, etc.
Automatic Calls and SMS	User's phone bill is increased by making calls and sending SMS to some premium numbers.

Root Exploits	The malware will gain system root privileges and takes control of the system and modifies the information.
Search Engine Optimizations	Artificially search for a term and simulates clicks on targeted websites in order to increase the revenue of a search engine or increase the traffic on a website.
Dynamically Downloaded code	An installed benign application downloads a malicious code and deploy it in the mobile devices.
Covert Channel	A vulnerability in the devices that facilitates the information leak between the processes that are not supposed to share the information.
Botnets	A network of compromised mobile devices with a BotMaster which is controlled by Command and Control servers (C&C). Carry out Spam delivery, DDos attacks on the host devices

d) Android Malware Family

According to (Zaki, Sahib, et al., 2013.), AnserverBot, DroidDream and DroidKungfu were defined as follow:

- **AnserverBot**

The AnserverBot have the potentiality to repackage itself by contacting the C&C server. The malicious application is embedded to a legitimate application then sending device information by connecting automatically to a C&C server and downloading a new malicious repackage application without the user consent. The repackage application contents malicious payloads, namely, anserverb.db and anservera.db (Trendmicro, Zhou and Jiang).

- **DroidDream**

DroidDream have the capability to steal device's information, connect to the internet, access device's SD card, modify system files, attempt to gain root/admin access, connect to C&C server and download another malicious package.

- **DroidKungfu**

The goals of DroidKungfu malware family are to gain a root access to the android OS, capturing device information and connecting itself to a C&C server for sending captured information and receiving command (Eset antivirus, F-secure and Microtrend report).



e) **Traces of Data Source**

- **Network Traffic**

Nowadays, smart phones, especially Android based, have attracted the users community for their feature rich apps to use with various applications like browsing, chatting, mailing, maps, GPS, mobile payment applications, image editing and video processing. Meanwhile the popularity of these devices attracted the malicious attackers as well. Thus, mobile network traffic analysis has been used to identify the malware by monitoring for any malicious communication activity through the network. Some of the features or parameter used in this project based upon the behavior of different malware are summarized in **Table 2.6** (Jeong et al., 2016).

Table 2.6 List of network traffic features

Parameter
IP number
Port number
Time
Packet Size
Protocol
Packet Count

- **System Call**

The system call tracing is one of the effective dynamic analysis technique for detecting malware as it can analyze the malware at the run time. Besides, this technique does not require the application code for malware detection. Therefore, this can detect that android malware also which are difficult to detect with static analysis of code (Malik and Khatter, 2016). Some of the parameter used in this project based upon the behavior of different malware are summarized in **Table 2.7** (Zovi et al., n.d.).

Table 2.7 List of System Call

Function
CREATEFILE
READFILE
WRITEFILE
CREATE FILE MAPPING
MAPVIEWOFFILE
RegOpenKeyEx
RegSetValueEx.
RegGetValue

- **Logcat**

The main objective of logcat in this project is to gather evidence that SMS or MMS messages are sent. For example, malware that sends SMS in the background and deletes the messages in the SMS application, can be spotted.

2.3 Critical Review

Currently, there is a lot of research was made due to the development of malware itself as shown in **Table 2.8**.

Table 2.8 List of Existing Research(Traces)

No	Authors/ Years	Analysis Approach	Parameter	Software/ Hardware
1	(Hsieh,Wan -Chen, et al., 2015)	Static, dynamic and hybrid analysis.	APP	XmanDroid : Privilege escalation detection tool
2	(Baskaran and Ralescu, 2016)	Static, dynamic and hybrid analysis.	API, network traffic, battery usage, CPU usage, permission call and number of running process	Strace : Monitoring tool. Logger : Extracting the sum of Internet traffic, percentage of battery used and battery temperature for every minute. Linux based features : Monitoring memory, CPU and network. Web crawler : Extracting tool

3	(Gascon, Yamaguchi, et al., n.d.)	Static analysis.	Structure of the underlying code.	Support Vector Machine (SVM), Androguard framework.
4	(Hsiao and Chen, 2016)	Dynamic analysis.	API	Virtual Machine, QEMU Machine Protocol, virtual Android device, Android Debug Bridge.
5	(Qin, Xu, et al., 2012)	Static analysis.	APK	Windows 7
6	(Huang, Zheng, et al., 2016)	Static and dynamic analysis.	APK	VirusTotal, Spark, Apache Cassandra, emulator.
7	(Zaki, Sahib, et al., 2013.)	Hybrid analysis.	API, network traffic, event handler	VMware
8	(Mas'ud, Sahib, et al., 2016)	Dynamic analysis.	System call	Strace : Capturing system call in a log files.
9	(Dash et al., 2016)	Dynamic analysis.	System call	Binder, sandbox.
10	(Peng, Zhao, et al., 2016)	Dynamic analysis.	API	Apktool, dex2jar.
11	(Data, 2016)	Dynamic analysis.	API and permissions	Monkey tool
12	(Shibahara, Yagi, et al., 2016)	Dynamic analysis.	Network behaviour	BotnetWatcher
13	(Saxena, Shrivastava, et al., 2016)	Dynamic and static analysis.	API, system call	Androguard, VirusTotal, Strace Monkey tool

14	(Zhenyu, Ming, Zhen, et al., 2016)	Dynamic analysis.	network, file system and registry.	Virtual machine, Virusshare
15	(S. Wu, Wang, et al., 2016)	Static analysis.	API	Dex2jar, Java static analysis tool: Soot or WALA.
16	(Suarez-tangil, Tapiador, et al., 2014)	Dynamic and static analysis.	Code structure.	Androguard
17	(Huda, Abawajy, and et al., 2016)	Hybrid analysis.	API	IDA Pro, Python, SQLite
18	(Shabtai, Mimran, et al., 2014)	Dynamic analysis	Network traffic	Android Software Development Kit
19	(Abdullah, Ibrahim, et al., 2015)	Static analysis.	System call	APIfy: web service that provides .apk archives of Android application.
20	(Shaikh Bushra and Madhumita, 2015)	Dynamic analysis.	System call	Java programming, Eclipse IDE, BlueStacks Apps Player, Android SQLite, Android Development Tools.

Based on **Table 2.8**, even most of the research is about malware detection, the parameter can be used for this project as well. Majority of the research have used system call as the source but it is important to have more than one source in order to get complete

and accurate evidences. For example, network traffic and logcat are also contributing to accuracy of the evidences.

2.4 Proposed Solution

In this project, the focus is only on four type of Android malware family which is AnserverBot, DroidKungfu, and DroidDream. The traceability matrix of malware behaviour is based on network traffic, system call and logcat. Malicious activity can be identified by monitoring the network traffic while malicious code also can be identified by tracing system call as well as logcat.

2.5 Conclusion

As a conclusion, this chapter had discussed in-depth on a literature review of this project. All related study about android, malware, definition, behaviour and traces of data sources have done. This chapter presented the differences on how they analyse behaviour of Android malware between the other research.

CHAPTER III

METHODOLOGY



3.1 Introduction

In this chapter, the scopes of this project were defined. Some suitable and reasonable methods were discussed in more details as to make sure this project is in progress and working well.

3.2 Methodology

In this section, a few phases that involved in this chapter were described as shown in **Figure 3.1**:

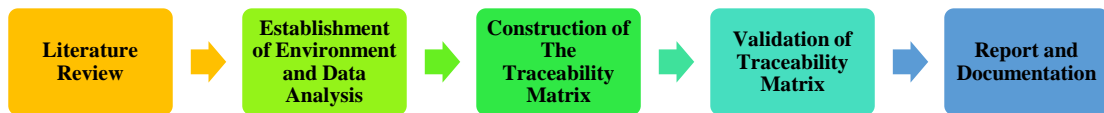


Figure 3.1 Project Methodology

Generally, there are five phases in this project. These phases are discussed in the following subsection.

3.2.1 Phase I: Literature Review

In this phase, a study of android malware specifically on malware behaviour was conducted as shown in **Figure 3.2**. After all, the information was used as reference for next phases.

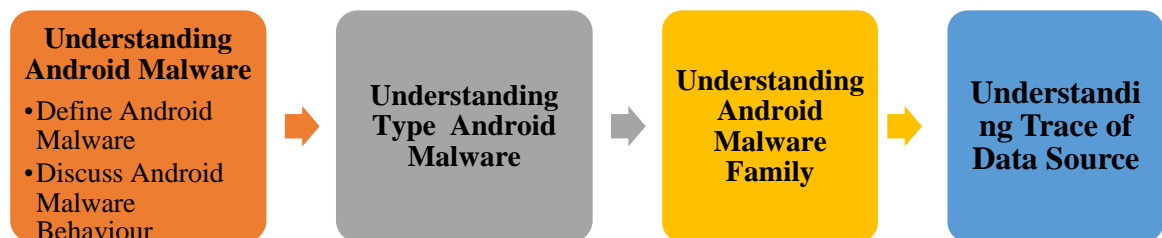


Figure 3.2 Literature Review

3.2.2 Phase II: Establishment of Environment and Data Analysis

This phase discussed on how data was collected as shown in **Figure 3.3**. An experimental environment will be established by activating the android malware. The environment is set up with hardware and software requirements of proposed project and will be used to simulate the incident of android malware propagation and behaviour. An isolated environment is needed in order to prevent the malware outbreak the network. Thus, this project used Genymotion as the virtualization platform. The data then collected and analysed.

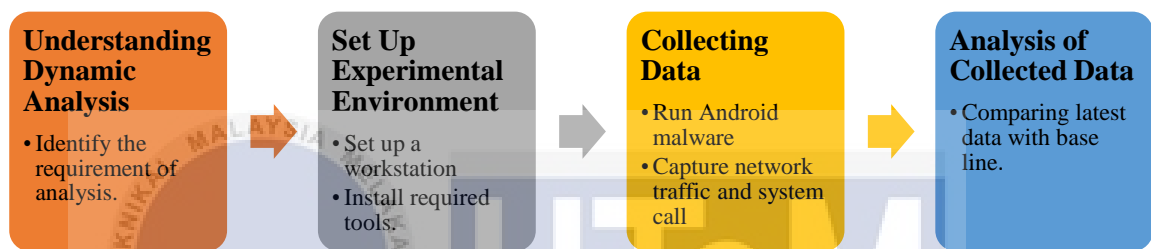


Figure 3.3 Analysis and Design

3.2.3 Phase III: Construction of The Traceability Matrix

In this phase, the attributes are determined by discovering and extracting the android malware incident events. Then, the correlations or relationships of the events will be identified based on the android malwares involve in this project. These attributes and their relationship will be used in constructing the traceability matrix. This matrix will facilitate the forensic investigator in tracing and linking the incident events in order to identify the origin of the incident.

3.2.4 Phase IV: Validation of Traceability Matrix

Hence, it is important to check the usability of the traceability matrix whether the data were accurate or not after all the data have been collected, gathered and tabulated. Several sets of android incident data will be used during implementation.

3.2.5 Phase V: Report and Documentation

Finally, the traceability matrix and the results from analysis will be documented.

3.3 Project Milestone

The milestone of this project was stated in Table 3.1.

Table 3.1: Project Milestones

Week	Activity	Note
1 13 – 17 February 2017	Proposal Submission	Topic research.
2 20 – 24 February 2017	Proposal Enhancement	Topic research.
3 27 Feb – 3 Mar 2017	Chapter 1	Topic research.
4 6 – 10 Mar 2017	Chapter 2	Topic research.
5 13 – 17 Mar 2017	Chapter 3	Network environment set up.
6 20 – 24 Mar 2017	Chapter 3	Installation of operating system, virtual machine and android emulator.
7 27 – 31 Mar 2017	Chapter 4	Collecting normal behaviour of the application.
8 1 – 9 April 2017	Mid Semester Break	Research
9 10 – 14 April 2017	Chapter 4	Collecting normal behaviour of the application

10 17 – 21 April 2017	Chapter 4	Collecting normal behaviour of the application
11 24 – 28 April 2017	Chapter 4	Collecting normal behaviour of the application
12 1 – 5 May 2017	Chapter 5	Collecting information of the application after the infection of the malware
13 8 – 12 May 2017	Chapter 5	Collecting information of the application after the infection of the malware
14 15 – 19 May 2017	Chapter 5	Collecting information of the application after the infection of the malware
15 22 – 26 May 2017	Chapter 5	Collecting information of the application after the infection of the malware
16 29 May – 2 June 2017	Chapter 5	Collecting information of the application after the infection of the malware
3 – 11 June 2017	Semester Break	Research
12 – 16 June 2017	Chapter 6	Finding the malware signature command and function in the application source code
19 – 23 June 2017	Chapter 6	Finding the malware signature command and function in the application source code

26 – 30 June 2017	Chapter 6	Finding the malware signature command and function in the application source code
3 – 7 July 2017	Chapter 7	Finding the malware signature command and function in the application source code
10 – 14 July 2017	Chapter 7	Finding the malware signature command and function in the application source code
17 – 21 July 2017	Chapter 7	Finding the malware signature command and function in the application source code
24 – 28 July 2017	Documenting Result	Documenting the project findings
31 July – 4 August 2017	Documenting Result	Documenting the project findings
7 – 11 August 2017	Documenting Result	Documenting the project findings
14 – 18 August 2017	Final Presentation	Presenting the project result to the supervisor and evaluator

3.4 Conclusion

Therefore, every project has different methodologies that is being used to make the project complete and working well. Generally, the methodology is divided into two parts, there are theoretical study and exploratory study. Next chapter, details explanation of project design will be discussed.



CHAPTER IV

DESIGN



اونيورسيتي تيكنيكل مليسيا ملاك

4.1 Introduction

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

This chapter is focused on the design and implementation of the proposed project. Generally, this chapter discussed about requirements of software and hardware for environment set up, experimental design including physical and logical design as well as how traceability matrix of Android malware is constructed. In addition, the details of every phases are described in this chapter.

4.2 Requirement

In this section, the requirements of hardware and software of the proposed project are discussed.

4.2.1 Software Requirement

i. Windows 7

- A Microsoft operating system that had been used as basic platform to install the other software. **Figure 4.1** shows the specification of the workstation.



Figure 4.1 Specification of The Workstation

ii. Genymotion Android Emulator

- An Android emulation platform as well as virtualization platform that can develop and test Android applications without using a physical device. Besides, this emulator provides some Android version which is from 4.1.1 until 7.1.0 and device model such as Google, HTC, Motorola, Samsung and Sony. **Figure 4.2** and **Figure 4.3** show the interface of Genymotion Android Emulator.



Figure 4.2 Genymotion Interface (i)

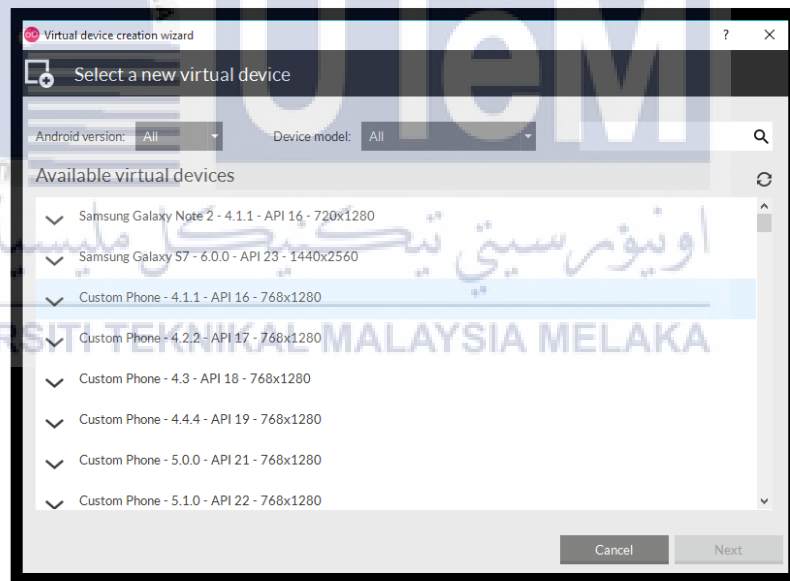


Figure 4.3 Genymotion Interface (ii)

iii. Terminal Emulator

- A terminal window allows the user access to a text terminal and all its applications such as command-line interfaces (CLI) and text user interface (TUI) applications.

iv. Wireshark

- An open source tool for profiling network traffic and analyzing packets in real time and display the packet data in detailed and human-readable format.

v. Strace

- A diagnostic, debugging and instructional user space. It is used to print a list of system calls made by the program.

4.2.2 Hardware Requirements

Table 4.1 shows the details of hardware requirement.

Table 4.1 Details of Hardware Requirement

Feature	Specification
Manufacturer	Dell Inc.
Model	Dell OptiPlex 7010
Processor	Intel Core i5 (3rd Gen) 3470 / 3.2 GHz
Memory (RAM)	2GB
Storage	250GB

4.3 Analysis Approach

In this section, both experimental and analysis design will be described as follows:

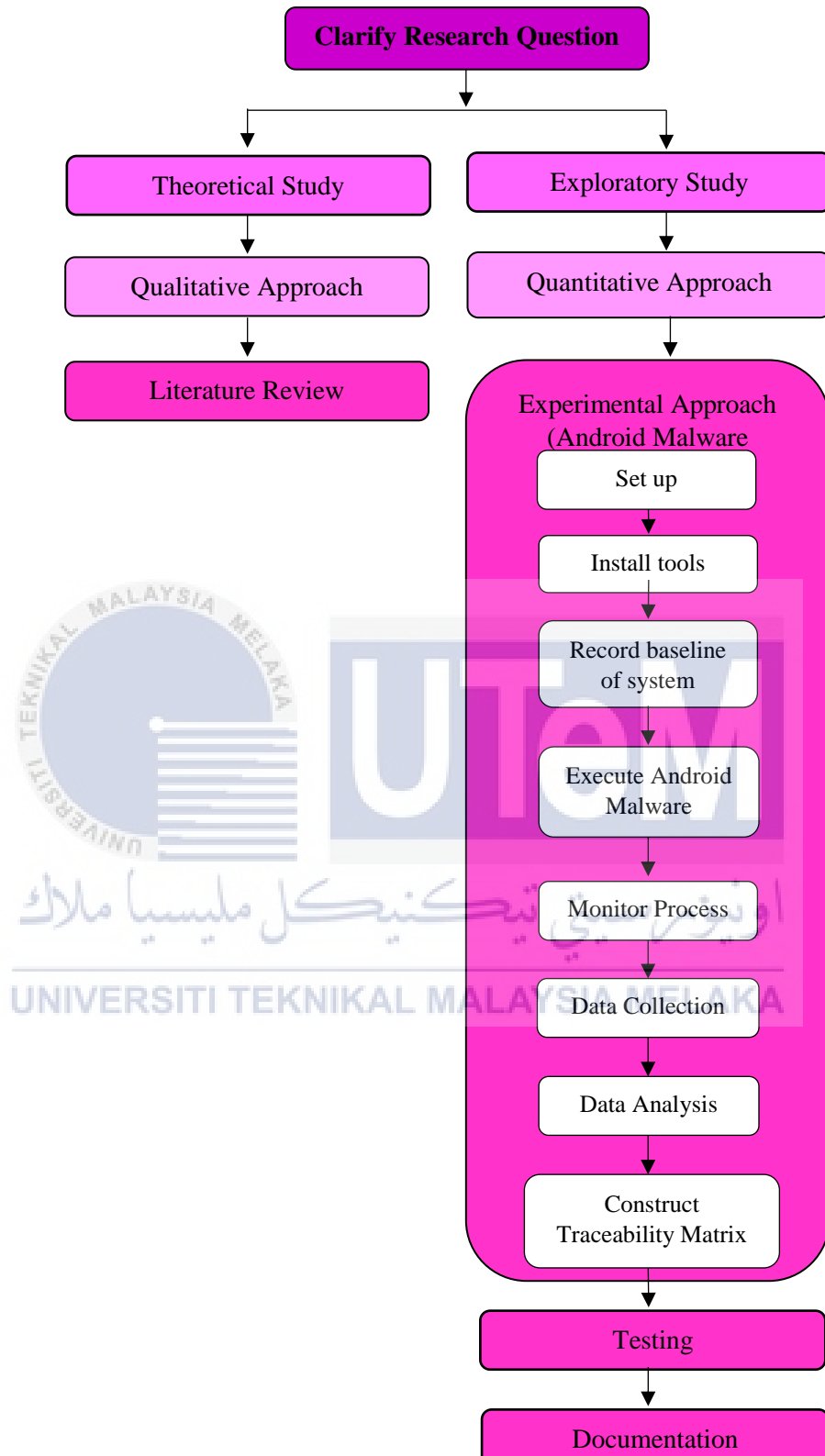


Figure 4.4 Analysis Approach

Figure 4.4 shows analysis approach to carry out the process of constructing traceability matrix. Generally, this project implemented two type of study which is theoretical and exploratory study. In theoretical study, a qualitative approach is applied which is literature review while in exploratory study, a quantitative approach is applied which is experimental approach. The details of the experimental approach will be explained in Chapter 5 and Chapter 6.

4.4 Network Design

The experimental environment is designed for collecting and analyzing data.

4.3.1 Physical Design

Figure 4.5 shows the physical design of this project. This design consists of a workstation and malware remote server as well as the internet is connected. Android malware will be activated in workstation while malware remote server is located in attacker site.



Figure 4.5 Physical Design

4.3.2 Logical Design

Figure 4.6 shows the logical design of this project. This design consists of a workstation, a virtual device (Android emulator) and malware remote server as well as the internet is connected. Then, the workstation will capture the network traffic and system

call on Android malware. The comparison between baseline of the system and the infected system is carried out as to construct traceability matrix of Android malware.

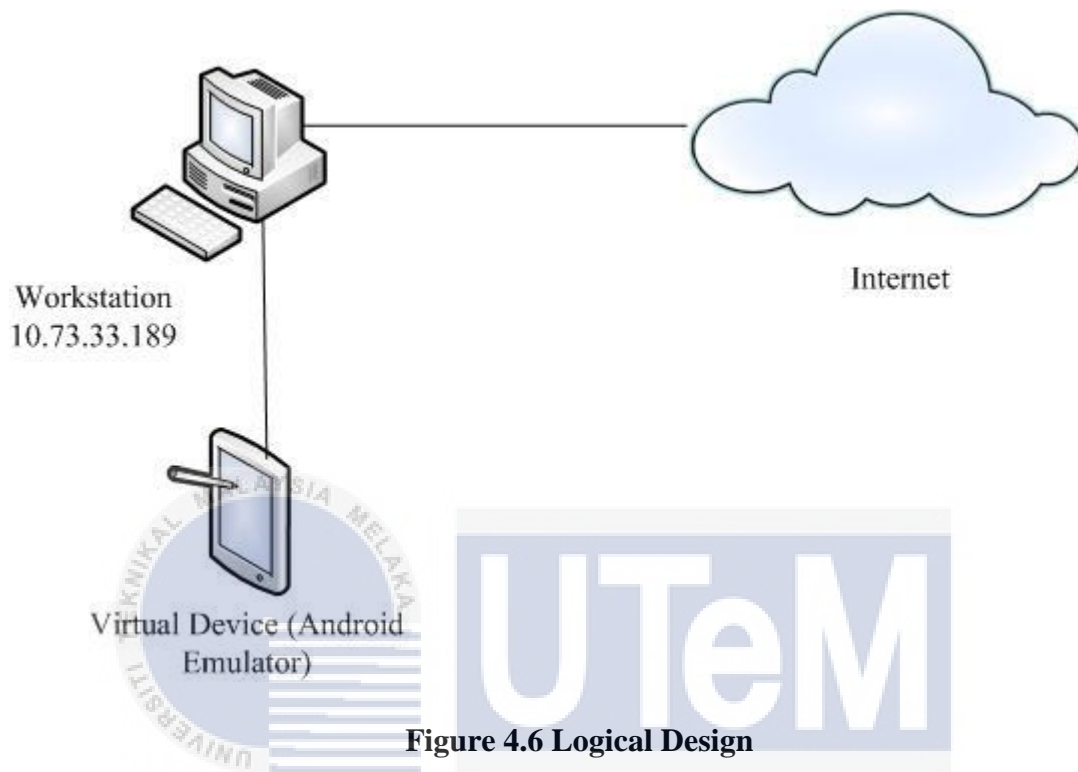


Figure 4.6 Logical Design

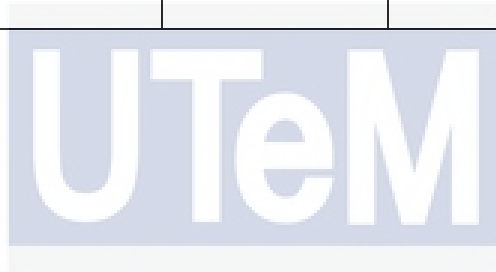
4.5 Traceability Matrix Design

Generally, the proposed traceability matrix is to identify the relationship between the components of forensic investigation and the incident event discovered. As to make it real, first of all, “artifact types” are identified. For this project, artifact types are defined as sources of evidences which are network traffic, system call and logcat. The next step is the relation between those artifacts is defined. For this project, every source of evidences is mapped between each other. For example, network traffic is mapped to system call and system call is mapped to logcat. Lastly, a traceability matrix for each link between artifact types is constructed. In addition, the information from multiple matrices is combined into

a single one in order to make important information more accessible as drafted in **Table 4.2**.

Table 4.2 Home Page

	DATA NAME	Source 1: Network Traffic	Source 2: System Call	Source 3: Logcat
TEST CASE ID		<u>N1</u>	<u>S1</u>	<u>L1</u>

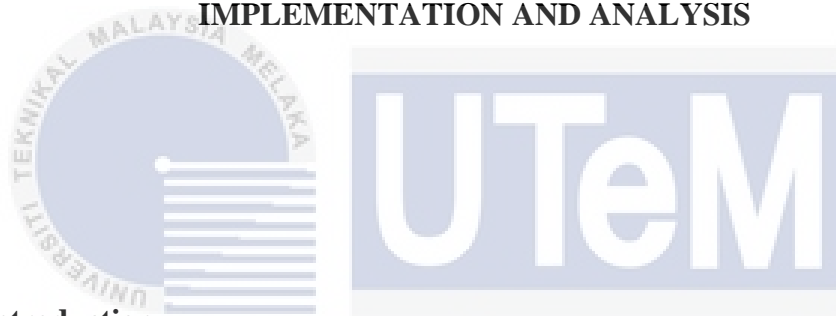


4.6 Conclusion

This chapter had covered experimental design of this project and how this project is carried out in details. In chapter 5, the implementation of traceability matrix will be discussed.

CHAPTER V

IMPLEMENTATION AND ANALYSIS



5.1. Introduction

This chapter has been discussed about the experimental setup and the process of collecting traces from three different sources of evidences. The implementation that had been done is based on the previous chapter.

5.2 Environment Set Up

Based on the design that had mentioned in Chapter 4, the actual environment will be set up as **Figure 5.1**. Firstly, Windows 7 is installed as the platform because most of the programs are able to run successfully on Windows 7. Secondly, since this project is using Android malware, Genymotion Android Emulator is the most suitable application as to activate Android malware on it instead of using real device. Lastly, Wireshark is installed as it has sorting and filtering option so the packets can be easily read as well as Busybox is installed as to read and capture the system call. The setting up of a controlled and sterile environment is undeniably essential for analyzing malware.

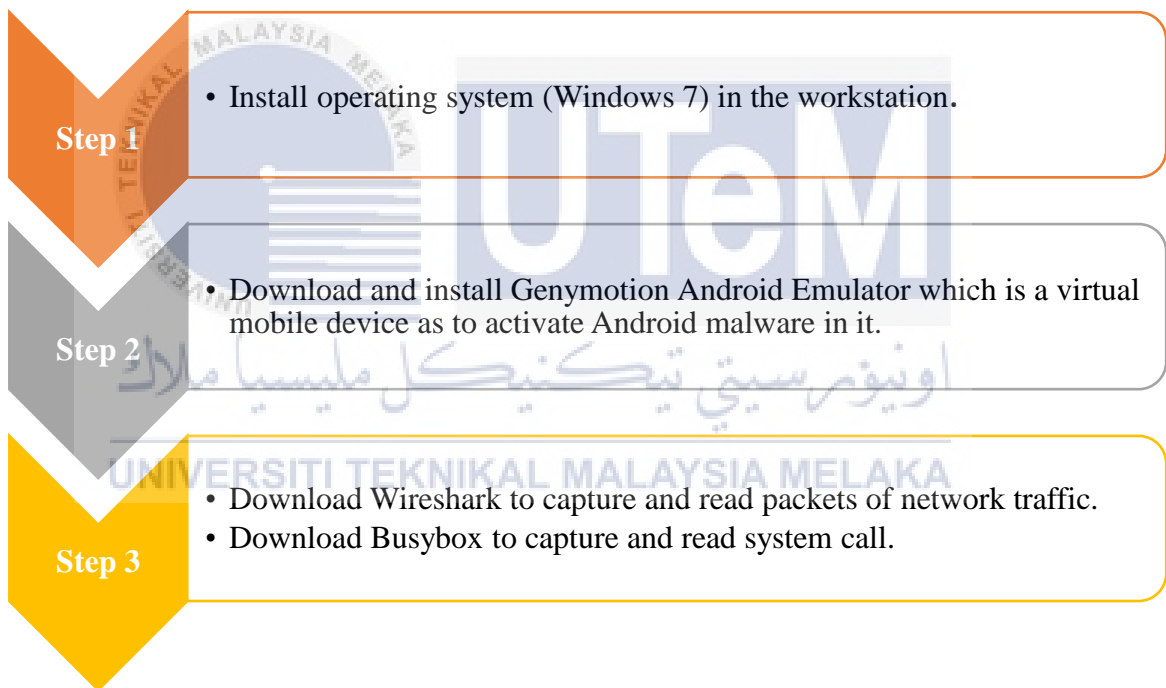


Figure 5.1 Steps on Environment Set Up

An isolated environment is needed as to prevent the malware outbreak the network connection when performing malware analysis. After that, this project will proceed with collecting traces in network traffic, system call and logcat.

5.3 Process of Collecting Traces

In this section, the process of collecting traces from network traffic, system call and logcat will be explained. This analysis is based on three malware which are AnserverBot, DroidKungfu and DroidDream. Based on these three malwares, common traces will be selected to construct the traceability matrix.

5.3.1 Process of Collecting Traces from Network Traffic

Firstly, baselining the environment. "Baselining" means taking a snapshot of the existing environment. The elements of the environment that have to be baselined is network traffic. Sniffing software is used for this project. Any sniffing software running in lavish mode is sufficient for this project. Nevertheless, it is recommended to use a protocol analyzer like Wireshark to make this project easier. After activating the malware, the network traffic is captured and the differences between the new snapshot and the baseline snapshot are determined. In this project, only malicious URL is considered as malicious packet. To get the details of the URL, simply right click and select TCP stream. An appropriate display filter and pop up a dialog box with all the data from the TCP stream laid out in order, as shown in **Figure 5.4**. If there is no any malicious packets data, repeat Step 1 to Step 4 until the malicious packets are found. The process of collecting network traffic data is summarized as shown in **Figure 5.2**:

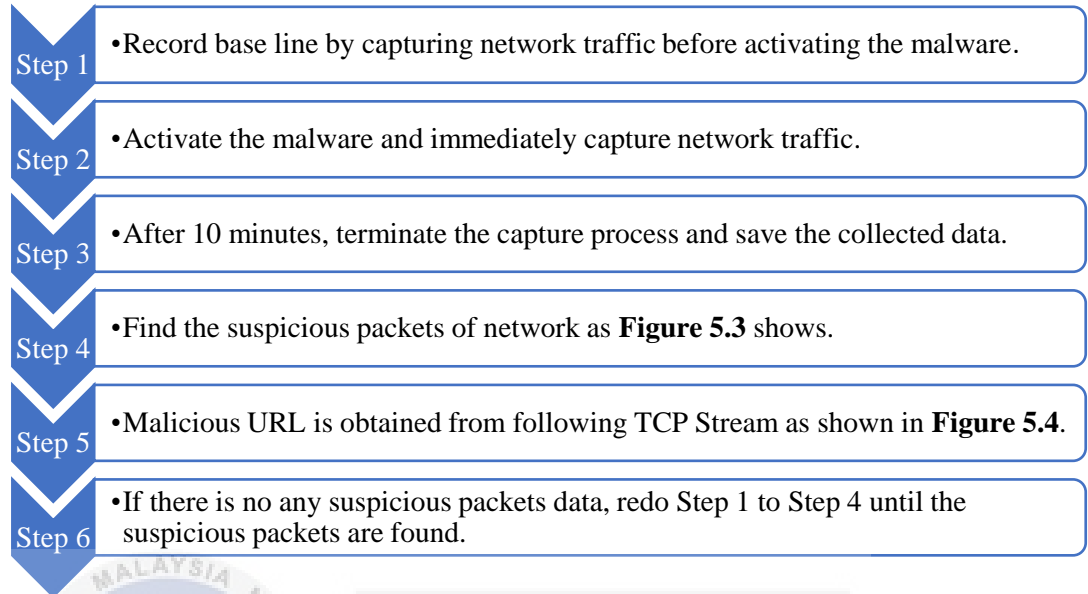


Figure 5.2 Steps of Collecting Data from Network Traffic

No.	Time	Source	Destination	Protocol	Length	Info	source port	desport
88	2014-09-19 16:57:05.217929	192.168.1.112	109.201.133.191	HTTP	482	POST /v2/svc/android/a2.do?md5=686473c9b14592...	54669	80
73	2014-09-19 16:56:58.225411	192.168.1.112	199.2.137.140	HTTP	656	POST /jk.action?a=502121505149379&a1=0qS2fqVs...	42286	8080

Figure 5.3 Request Method

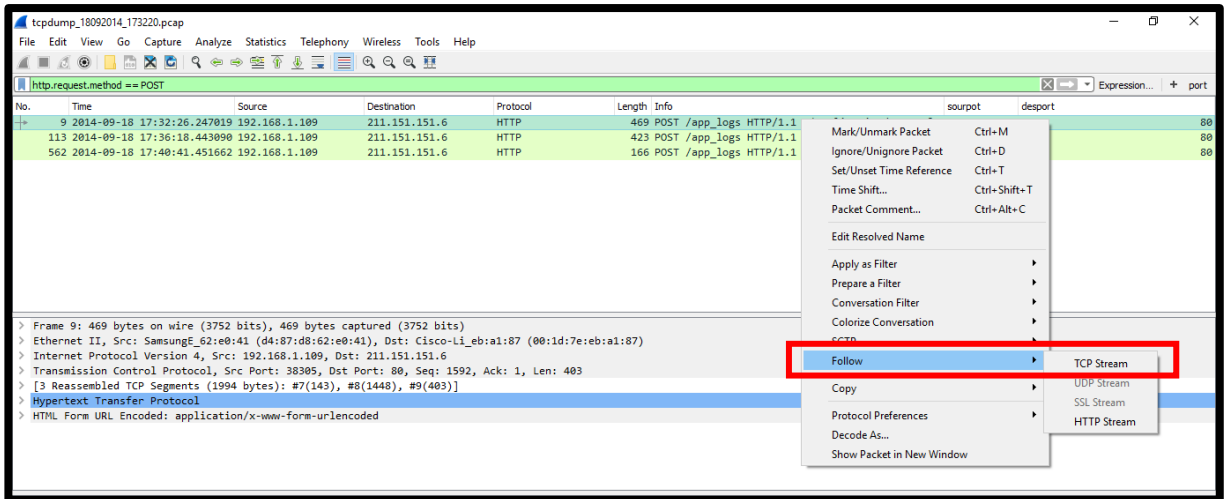


Figure 5.4 Follow TCP stream

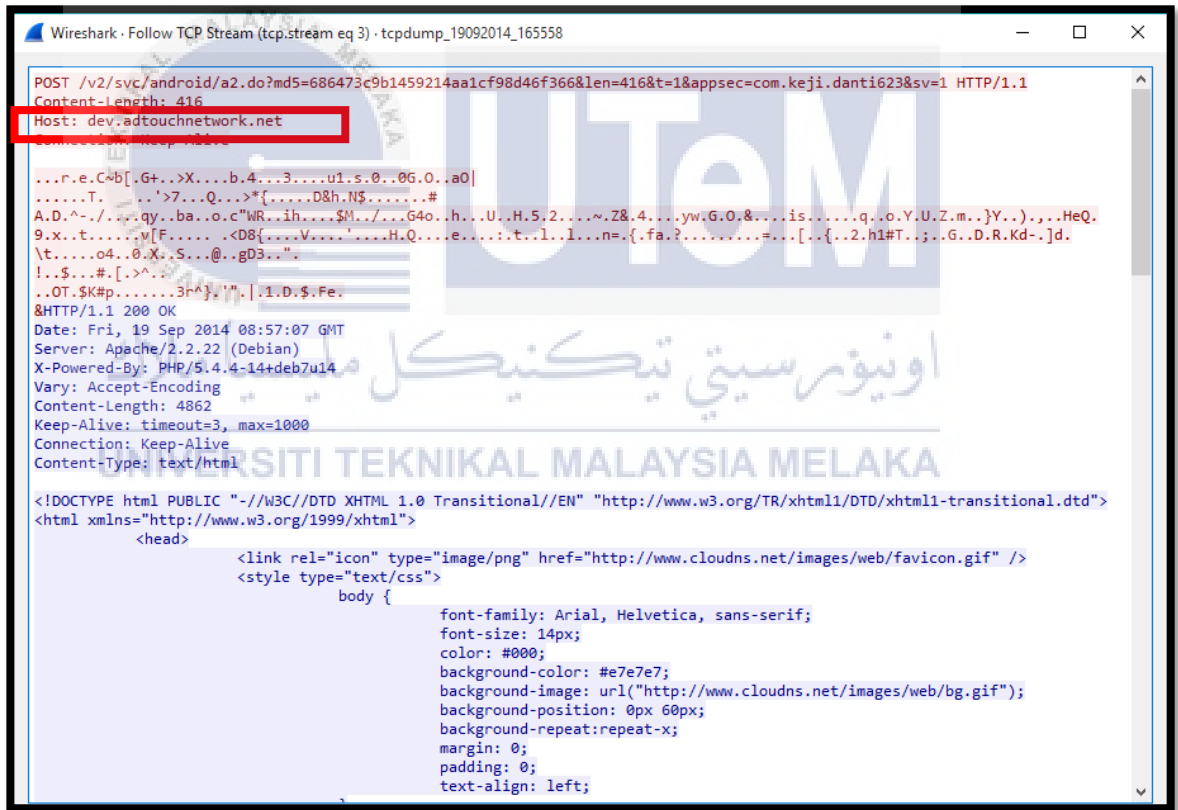


Figure 5.5 Domain Name

AnserverBot:

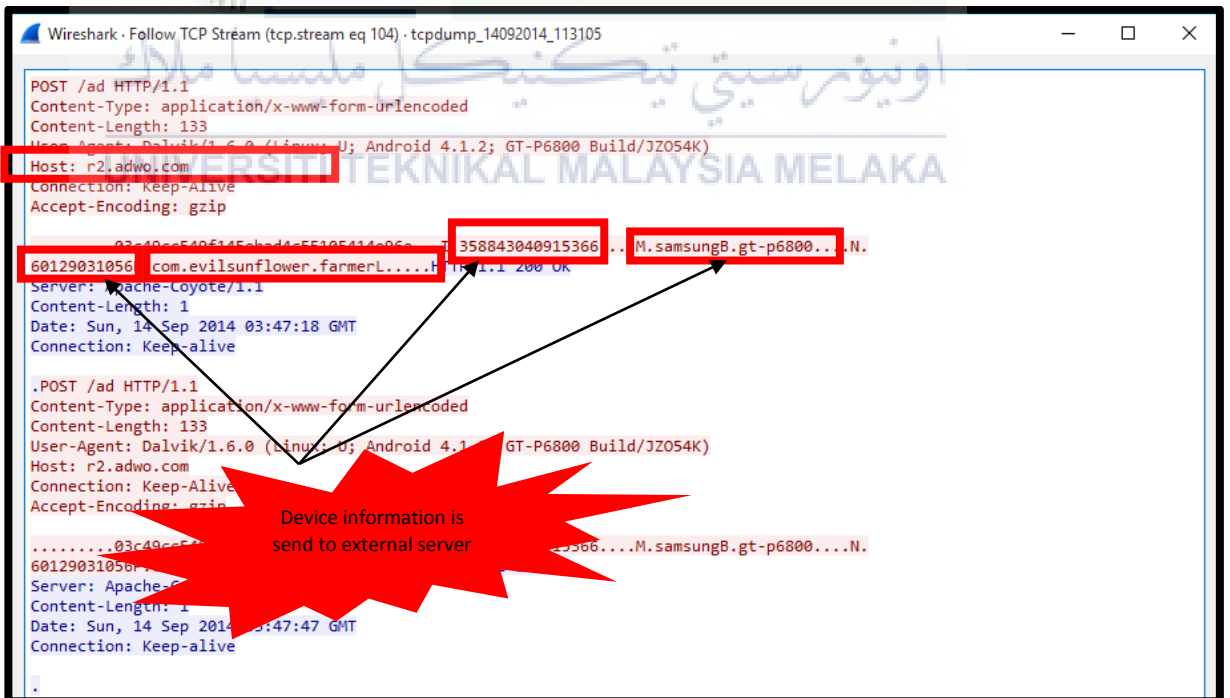


```
Wireshark · Follow TCP Stream (tcp.stream eq 2) · tcpdump_19092014_165558
POST /jk.action?a=502121505149379&a1=0qS2fqVs0qSLfqlBfNrB&c=5&c1=01_&d=4.1.2&d1=0vQs9wV_&e=samsung&e1=rCkgr430zY__&f=GT-P6800&f1=iLlgyJ7QfJS_&g=8&g1=PS_&h=351522052444237&h1=fHvs0vQ2fv2011nf4&i=16&i1=fq7_&j=1411117017296&j1=fqlsfqjs0NSs0NVB0n_&k=0&k1=fS_&l=0&l1=fS_&m=0&m1=fS_&n=0&n1=fS_&p=com.keji.danti623&_l=7CMg9IgrRHID0ztkOutDCfwf_&key=fqlsfqjs0NSs0Nrcf1_
HTTP/1.1
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.2; GT-P6800 Build/JZ054K)
Host: b4.cookier.co.cc:8080
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Figure 5.6(a) Domain Name (AnserverBot)

Based on AnserverBot, the malware attempts to request from *b4.cookier.co.cc:8080* as shown in **Figure 5.6(a)**. From this TCP stream, malicious file named *com.keji.danti623* as can be seen in **Figure 5.6(a)**. As to make sure what the malicious file is actually did, it can be found in system call and logcat too by finding the malicious file name.

DroidKungfu:



```
Wireshark · Follow TCP Stream (tcp.stream eq 104) · tcpdump_14092014_113105
POST /ad HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 133
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.2; GT-P6800 Build/JZ054K)
Host: r2.adwo.com
Connection: Keep-Alive
Accept-Encoding: gzip
.....03c49ccf5d.....358843040915366...M.samsungB.gt-p6800...N.
60129031056...com.evilsunflower.farmerL...H...L.1.200 OK
Server: Apache-Coyote/1.1
Content-Length: 1
Date: Sun, 14 Sep 2014 03:47:18 GMT
Connection: Keep-alive
.POST /ad HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 133
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.2; GT-P6800 Build/JZ054K)
Host: r2.adwo.com
Connection: Keep-Alive
Accept-Encoding: gzip
.....03c49ccf5d.....358843040915366...M.samsungB.gt-p6800...N.
60129031056...com.evilsunflower.farmerL...H...L.1.200 OK
Server: Apache-Coyote/1.1
Content-Length: 1
Date: Sun, 14 Sep 2014 03:47:47 GMT
Connection: Keep-alive
```

Device information is send to external server

Figure 5.6(b) Domain Name (DroidKungfu)

Based on DroidKungfu, the malware attempts to request from *r2.adwo.com* as shown in **Figure 5.6(b)**. From this TCP stream, malicious file named *com.evilsunflower.farmer* as can be seen in **Figure 5.6(b)**. The information send to the C&C server are the IMEI number of the device 358843040915366, the device product's model Samsung GT-P6800 and the device phone number 60129031056. As to make sure what the malicious file is actually did, it can be found in system call and logcat too by finding the malicious file name.

DroidDream:



Figure 5.6(c) Domain Name (Droid Dream)

Based on DroidKungfu, the malware attempts to request from *www.umeng.com* as shown in **Figure 5.6(c)**. The information send to the C&C server are the IMEI number of the device 358843040915325, the device product's model Samsung GT P6800, Android version 4.1.2 and the device phone carrier MY MAXIS. As to make sure what

the malicious file is actually did, it can be found in system call and logcat too by finding the domain name.

It can be concluded that malicious traffic is identified when the malware did some http request since most of the malware do http request when performing malicious activities. Thus, the common traces from network traffic are shown in **Table 5.1**.

Table 5.1 List of Traces in Network Traffic

Traces
IP address
Port number
Method
URL

5.3.2 Process of Collecting Traces from System Call

First, malware apk file is installed and launched in Genymotion Android Emulator. Some activities such as phone call, texting and browsing had made. Next, application is terminated and the data is collected. The data is opened with NotePad ++ as shown in **Figure 5.8(a)** and read line by line to identify the traces. The malicious activities such as access directory file, execute system file and change mode are identified. If there is no any malicious activity, repeat Step 1 to Step 4 until the malicious activity is found. The process of collecting system call data is simplified as shown in **Figure 5.7**:

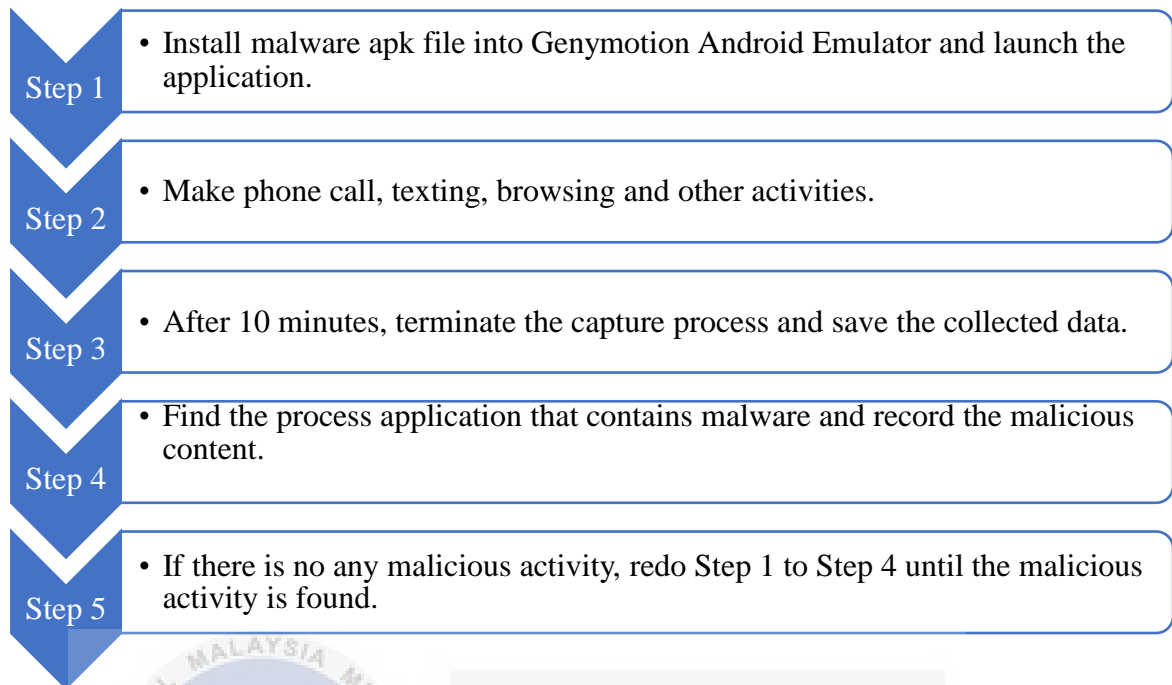


Figure 5.7 Steps of Collecting Data from System Call

i. Determine File Accessibility

The *access()* checks for the process whether would be allowed to read, write or test for existence of the file whose name is *pathname*. All the malicious file names was retrieved from network analysis as discussed in **5.3.1**.

AnserverBot:

```

1546 6667,1411117037.450045,"futex",1.3,"0x409a569c","0x81 /& Futex_222 */",1",...1.0.001481
1547 6667,1411117037.452241,"access","IO",0,2,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml"" ,"F_OK",,,,,"-1",0.000094
1548 6667,1411117037.452706,"open","IO",0,3,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml"" ,"O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE",0
1549 6667,1411117037.453104,"fstat64",,0,2,"74","{st_mode=S_IFREG|0600, st_size=0, ...}",,,,,0,0.000037
1550 6667,1411117037.453736,"write","IO",0,3,"74",""<?xml version='1.0' encoding='utf-8' standalone='yes' ?>\n<map>\n<int name=""book_firstVisibleItem_

```

Figure 5.8(a) Gain Access to a File (AnserverBot)

The *access()* checks for the process whether would be allowed to read, write or test for existence of the file named *com.keji.danti623* as shown in **Figure 5.8(a)**.

DroidKungfu:

```
315361 14252,1410665789.481248,"futex",,1,3,"0x4123a9d8","0x80 /* Futex ??? */",,2,,,,,0,0.001701
315362 14252,1410665789.483085,"access","IO",0,2,""/data/data/com.evilsunflower.farmer/shared_prefs/sdkPrefs.xml.bak"", "F_OK",,,,,,-1",0.000064
315363 14252,1410665789.483463,"access","IO",0,2,""/data/data/com.evilsunflower.farmer/shared_prefs/sdkPrefs.xml"", "F_OK",,,,,,-1",0.000057
315364 14252,1410665789.483828,"stat64","IO",0,2,""/data/data/com.evilsunflower.farmer/shared_prefs/sdkPrefs.xml"", "0x535ffbf8",,,,,,-1",0.000054
```

Figure 5.8(b) Gain Access to a File (DroidKungfu)

The *access()* checks for the process whether would be allowed to read, write or test for existence of the file named *com.evilsunflower.farmer* as shown in **Figure 5.8(b)**.

DroidDream:

```
36896 17872,1412306645.893636,"futex",,1,3,"0x409a269c","0x81 /* Futex ??? */",,1,,,,,1,0.002234
36897 17872,1412306645.896181,"writev",,0,3,"3",["4", 1], [{"UmAd SDK 2.1\0", 13}, [{"[11:24:05]Ad loaded.-10-1\0", 26}], "3",,40,0.000126
36898 17872,1412306645.896892,"access","IO",0,2,""/data/data/com.beauty.leg/files"", "F_OK",,,,,,0,0.000091
36899 17872,1412306645.897358,"open","IO",0,2,""/data/data/com.beauty.leg/files/686DA94B758249A59AB92CF1A855349"", "O_RDONLY|O_LARGEFILE",,,,,,59,0.000079
36900 17872,1412306645.898117,"fstat64",,0,2,"59",["st_mode=S_IFREG|0662, st_size=126, ...]",,,,,,0,0.000053
```

Figure 5.8(c) Gain Access to a File (DroidDream)

The *access()* checks for the process whether would be allowed to read, write or test for existence of the file named *com.beauty.leg* as shown in **Figure 5.8(c)**.

ii. Open the File

The *open()* function modifies pathname into a file descriptor. *O_CREAT* will create file if the file does not exist while *O_LARGEFILE* will support 32 bits when the files cannot be represented in 31 bits to be opened.

AnserverBot:

```
3342 |11119536.608245,"umask",,0,1,"0",,,,,,77,0.000021
3343 |11119536.608317,"umask",,0,1,"011",,,,,,0,0.000020
3344 |11119536.608390,"umask",,0,1,"00",,,,,,11,0.000020
3345 |1119536.608460,"open","IO",0,2,""/data/data/com.keji.danti623/databases/db.db-journal"", "O_RDONLY|O_LARGEFILE",,,,,,78,0.000030
3346 |11119536.608638,"umask",,0,1,"011",,,,,,0,0.000020
3347 |11119536.608710,"umask",,0,1,"077",,,,,,11,0.000020
```

Figure 5.9(a) Open a File (AnserverBot)

Based on **Figure 5.10 (a)**, the *fstat64()* function is trying to get information of the file named *com.keji.danti623* and the file size is 20480.

Droid Kungfu:

```

648069 4918,1410661179.728745,"stat64","IO",0,2,""/system/framework/twframework-res.apk""",{st_mode=S_IFREG|0644, st_size=3613044, ...},,,,,0,0.000096
648070 4918,1410661179.729466,"stat64","IO",0,2,""/system/framework/framework-res.apk""",{st_mode=S_IFREG|0644, st_size=25320246, ...},,,,,0,0.000070
648071 4918,1410661179.730311,"stat64","IO",0,2,""/data/app/com.glu.android.dinercn-1.apk""",{st_mode=S_IFREG|0644, st_size=35014, ...},,,,,0,0.000066
648072 4918,1410661179.731095,"getpid",,0,0,,,,,4918,0.000051

```

Figure 5.10(b) Get Information of a File (Droid Kungfu)

Based on **Figure 5.10 (b)**, the *stat64()* function is trying to get information of the file named *com.glu.android.dinercn* and the file size is 35014.

DroidDream:

```

163 5041,1411032760.330824,"close","IO",0,1,"42",,,,,,0,0.500054
164 5041,1411032760.331958,"stat64","IO",0,2,""/data/data/com.droiddream.sex/shared_prefs/mobclick_agent_state_com.droiddream.sex.xml""",{st_mode=S_IFREG|0660, ...},,,,,0,0.000066
165 5041,1411032760.331958,"stat64","IO",0,2,""/data/data/com.droiddream.sex/shared_prefs/mobclick_agent_state_com.droiddream.sex.xml""",{st_mode=S_IFREG|0660, ...},,,,,0,0.000066
166 5041,1411032760.331958,"stat64","IO",0,2,""/data/data/com.droiddream.sex/shared_prefs/mobclick_agent_state_com.droiddream.sex.xml.bak""",{st_mode=S_IFREG|0660, ...},,,,,0,0.000066

```

Figure 5.10(c) Get Information of a File (DroidDream)

Based on **Figure 5.10 (c)**, the *stat64()* function is trying to get information of the file named *com.droiddream.sex* and the file size is 437.

iv. Change Permission Mode

The *chmod()* is defined as the mode of the file given by path or referenced by file is modified. **Table 5.2** shows the digit and the type of permission.

Table 5.2 Digit and Type of File Permission

Digit	Permission
0	None
1	Execute Only
2	Write Only
3	Write and Execute
4	Read Only
5	Read and Execute
6	Read and Write
7	Full

AnserverBot:

```

2487 6667,1411117038.771406,"chmod","IO",0,2,""/data/data/com.keji.danti623/databases/db.db"","0660",,,,,0,0.000184
2488 6667,1411117038.772343,"gettimeofday",,0,2,{1411117038, 772406},"NULL",,,,,0,0.000063
2489 6667,1411117038.772927,"gettimeofday",,0,2,{1411117038, 773001},"NULL",,,,,0,0.000064
    
```

Figure 5.11(a) Change Mode of the System (AnserverBot)

Based on **Figure 5.11(a)**, 0660 is a permission of a file. First digit is 0 specifies none permission, the second digit is 6 indicates that the file is set read and write permission to owner, the third digit is 6 indicates that the file is set read and write permission only to group and the fourth digit is 0 indicates that file is set none permission to all other.

Droid Kungfu:

```

153 13908,1410665581.085237,"fstat64",,0,2,"61",{st_mode=S_IFREG|0600, st_size=118, ...}",,,,,0,0.000041
154 13908,1410665581.085751,"close","IO",0,1,"61",,,,,0,0.000083
155 13908,1410665581.086026,"chmod","IO",0,2,""/data/data/com.evilsunflower.farmer/shared_prefs/airpushTimePref.xml"","0664",,,,,0,0.000169
156 13908,1410665581.086540,"stat64","IO",0,2,""/data/data/com.evilsunflower.farmer/shared_prefs/airpushTimePref.xml""",{st_mode=S_IFREG|0664, st_size=11
157 13908,1410665581.087222,"lstat64",.0,2,""/data/data/com.evilsunflower.farmer/shared_prefs/airpushTimePref.xml.bak"","0xbea265c0",,,,,"-1".0.000079
    
```

Figure 5.11(b) Change Mode of the System (Droid Kungfu)

Based on **Figure 5.11(b)**, 0664 is a permission of a file. First digit is 0 specifies none permission, the second digit is 6 indicates that the file is set read and write permission to owner, the third digit is 6 indicates that the file is set read and write permission to group and the fourth digit is 4 indicates that file is set read only permission to all other.

DroidDream:

```

891 5041,1411032769.361630,"close","IO",0,1,"42",,,,,,0,0.000033
892 5041,1411032769.361736,"chmod","IO",0,2,""/data/data/com.droiddream.sex/shared_prefs/mobclick_agent_state_com.droiddream.sex.xml"", "0660",,,,,,0,0.000061
893 5041,1411032769.361958,"stat64","IO",0,2,""/data/data/com.droiddream.sex/shared_prefs/mobclick_agent_state_com.droiddream.sex.xml"", "{st_mode=S_IFREG(0660, st_size=478, ...)}",,,,
894 5041,1411032769.362231,"lstat64",,0,2,""/data/data/com.droiddream.sex/shared_prefs/mobclick_agent_state_com.droiddream.sex.xml.bak"", "{st_mode=S_IFREG(0660, st_size=478, ...)}",,,,
895 5041,1411032769.362489,"unlink","IO",0,1,""/data/data/com.droiddream.sex/shared_prefs/mobclick_agent_state_com.droiddream.sex.xml.bak"",,,,,,0,0.000137

```

Figure 5.11(c) Change Mode of the System(DroidDream)

Based on **Figure 5.11(c)**, 0660 is a permission of a file. First digit is 0 specifies none permission, the second digit is 6 indicates that the file is set read and write permission to owner, the third digit is 6 indicates that the file is set read and write permission only to group and the fourth digit is 0 indicates that file is set none permission to all other.

v. Write to the File

The write() writes up to *count bytes* from the buffer pointed *buf* to the file referred to by the file descriptor *fd*.

AnserverBot:

```

80267 111119693.573407,"mprotect",,0,3,"0x40d58000", "4096", "PROT_READ",,,,,,0,0.000042
80268 111119693.573597,"write","IO",0,3,"71",,"getaddrinfo b4.cookie.co.cc ^ 1024 0 1 0\0"", "42",,,,,,42,0.000063
80269 111119693.573972,"mprotect",,0,3,"0x5172e000", "3320", "PROT_READ|PROT_WRITE|PROT_EXEC",,,,,,0,0.000066
80270 111119693.574161,"mprotect",,0,3,"0x5172e000", "3320", "PROT_READ|PROT_WRITE|PROT_EXEC",,,,,,0,0.000066

```

Figure 5.12(a) Write to the File (AnserverBot)

Figure 5.12 (a) shows write() is trying to write up *getaddrinfo b4.cookie.co.cc^1024* to the file referenced by the file descriptor which is 71 and the size of content is 42.

Droid Kungfu:

```
649846 5642,1410661249.202853,"mprotect",,0,3,"0x40d2c000","4096","PROT_READ|PROT_WRITE",,,,0,0.000055
649847 5642,1410661249.203033,"mprotect",,0,3,"0x40d2c000","4096","PROT_READ",,,,0,0.000046
649848 5642,1410661249.203225,"write","IO",0,3,"65",,""getaddrinfo r2.adwo.com ^ 1024 0 1 0\0""",,"37",,,,37,0.000064
649849 5641,1410661249.200600,"gettimeofday",,1,2," {1410661249, 200637}",,"NULL",,,,0,0.003055
649850 5641,1410661249.203816,"mprotect",,0,3,"0x52290000","3500","PROT_READ|PROT_WRITE|PROT_EXEC",,,,0,0.000126
```

Figure 5.12(b) Write to the File (DroidKungfu)

Figure 5.12 (b) shows `write()` is trying to write up `getaddrinfo r2.adwo^1024` to the file referenced by the file descriptor which is 65 and the size of content is 37.

DroidDream:

```
30028 5071,1411032978.060485,"mprotect",,0,3,"0x40d03000","4096","PROT_READ|PROT_WRITE",,,,0,0.000029
30029 5071,1411032978.060574,"mprotect",,0,3,"0x40d03000","4096","PROT_READ",,,,0,0.000023
30030 5071,1411032978.060659,"write","IO",0,3,"45",,""getaddrinfo www.umeng.com ^ 1024 0 1 0\0""",,"39",,,,39,0.000190
30031 5048,1411032978.060762,"lseek",,1,3,"0x939360c","0x81 /> FILEX ???",,"1",,,,0,0.022268
30032 5048,1411032978.061138,"sigprocmask",,0,3,"SIG_BLOCK",,"11" "IOCTL_USB1_PIPE1",,,,0,0.000028
```

Figure 5.12(c) Write to the File(DroidDream)

Figure 5.12 (c) shows `write()` is trying to write up `getaddrinfo www.umeng.com ^1024` to the file referenced by the file descriptor which is 45 and the size of content is 39.

vi. Rename the File

The `rename()` changes the name or location of a file, renames a file, moving it between directories if required. This activity only occurred in AnserverBot and DroidDream as shown in Figure 5.13(a) and Figure 5.13(b).

AnserverBot

```
6667,1411117038.996456,"access","IO",0,2,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml""",,"F_OK",,,,0,0.000075
6667,1411117038.996763,"access","IO",0,2,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml.bak""",,"F_OK",,,,1,0.000031
6667,1411117038.997190,"rename","IO",0,2,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml""",,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml""",,,,0,0.000031
6667,1411117038.997907,"open","IO",0,3,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml""",,"O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE",,"0600",,,,43,0.000162
6667,1411117038.998325,"fstat64",,0,2,"43",,"[st_mode=S_IFREG|0600, st_size=0, ...]",,,,0,0.000030
```

Figure 5.13(a) Rename the file (AnserverBot)

AnserverBot

```

121 5151,1411118344.343183,"mprotect",,0,3,"0x51f54000","3512","PROT_READ|PROT_EXEC",,,,0,0.000034
122 5151,1411118344.343493,"recv",,0,4,"71",,""POST /v2/svc/android/a2.do?md5=b0f32328531f5529748bf6ccc60&len=416&st=1&aappsec=com.keji.danti623&sv=1 HTTP/1.1\r\nContent-Length: 416\r\nHost:
123 5151,1411118344.344203,"recv",,0,4,"71",,""226(323(371a.\364 1\3708H\305)\330\205\3700\270\212\2712\214e\373\361^\236\272\242\246\225h\1377^k5\n\354\334x\215)\207\275\327\241\F\0z\
124 5151,1411118344.345358,"recvfrom",,0,6,"71",,""HTTP/1.1 200 OK\r\nDate: Fri, 19 Sep 2014 09:19:06 GMT\r\nServer: Apache/2.2.22 (Debian)\r\nPowered-By: PHP/5.4.4-14+deb7u14\r\nVary: Ac
125 5151,1411118344.984193,"mprotect",,0,3,"0x51f43000","2996","PROT_READ|PROT_WRITE|PROT_EXEC",,,,0,0.000146

```

Figure 5.15(a) Receive a message from a socket (AnserverBot)

DroidKungfu

```

44244 5018,1410660812.950166,"gettimeofday",,0,2,"1410660812.9502101","NULL",,,,0,0.000047
44245 5018,1410660812.950412,"recv",,0,4,"39",,""POST /ad HTTP/1.1\r\nContent-Type: application/x-www-form-urlencoded\r\nContent-Length: 127\r\nUser-Agent: Dalvik/1.6.0 (Linux; U; Android 4
44246 4918,1410660812.952791,"clock_gettime",,0,2,"CLOCK_MONOTONIC","(519, 113977624)",,,,0,0.000049
44247 4918,1410660812.952994,"clock_gettime",,0,2,"CLOCK_MONOTONIC","(519, 114171374)",,,,0,0.000046
44248 4918,1410660812.953192,"recvfrom",,0,6,"40",,"0xbe9bec98",,"34",,"64",,"0",,"0",,"-1",,0.000052
44249 4918,1410660812.953389,"ioctl",,0,3,"10",,"0xc0186201",,"0xbe9bf478",,,,0,0.000234

```

Figure 5.15(b) Receive a message from a socket (DroidKungfu)

DroidDream

```

30511 5071,1411032978.441308,"mprotect",,0,3,"0x50986000","17504","PROT_READ|PROT_EXEC",,,,0,0.000057
30512 5071,1411032978.441927,"recv",,0,4,"45",,""POST /app logs HTTP/1.1\r\nContent-Length: 1797\r\nContent-Type: application/x-www-form-urlencoded\r\nHost
30513 5071,1411032978.442832,"recv",,0,4,"45",,""content=%7B%22body%22%3A%7B%22launch%22%3A%5B%7B%22date%22%3A%222014-09-18%22%2C%22time%22%3A%2217%3A36%3D
30514 5048,1411032978.438718,"futex",,1,3,"0x4098569c",,"0x80 / Futex ??? *",,"2",,,,,"-1",,0.010045
30515 5048,1411032978.448866,"mprotect",,0,3,"0x52bac000","208",,"PROT_READ|PROT_WRITE|PROT_EXEC",,,,0,0.000050

```

Figure 5.15(c) Receive a message from a socket (DroidDream)

ix. Execute the file

The `execve()` executes the program pointed to by `filename`. Figure 5.16 shows `execve()` is trying to execute a program.

DroidKungfu

```

52438 5023,1410660817.841858,"close","IO",0,1,"095",,,,,"-1",,0.000018
52439 5023,1410660817.845011,"execve",,0,3,""/sbin/su",,"["su"]",,"[/ 14 vars *]",,,,,"-1",,0.000153
52439 5023,1410660817.845415,"execve",,0,3,""/vendor/bin/su",,"["su"]",,"[/ 14 vars *]",,,,,"-1",,0.000042
52439 5023,1410660817.845691,"execve",,0,3,""/system/sbin/su",,"["su"]",,"[/ 14 vars *]",,,,,"-1",,0.000036
52439 5023,1410660817.845959,"execve",,0,3,""/system/bin/su",,"["su"]",,"[/ 14 vars *]",,,,,"-1",,0.000032
52439 5023,1410660817.846122,"fcntl",,1,2,"0, 3200000",,"NULL",,,,0,0.114338
52438 4918,1410660817.742424,"fcntl",,1,3,"44",,"0x4700",,"0xbe9be890",,,,0,0.104111

```

Figure 5.16 Execute the File

x. **Delete File**

The *unlink()* deletes a name from the filesystem. **Figure 5.17(a)**, **Figure 5.17(b)** and **Figure 5.17(c)** show the function in the malware.

AnserverBot

```
42775 6667,1411117222.505916,"chmod","IO",0,2,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml""",0,0.000072
42780 6667,1411117222.504263,"stat64","IO",0,2,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml""",{"st_mode=S_IFREG|0666, st_size=318, ...}",0.000033
42781 6667,1411117222.504722,"stat64",0,2,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml.bak""",{"st_mode=S_IFREG|0666, st_size=318, ...}",0.000035
42782 6667,1411117222.505239,"unlink","IO",0,1,""/data/data/com.keji.danti623/shared_prefs/first_app_preferences.xml.bak""",0.000162
42783 6667,1411117222.505669,"gettimeofday",0,2,"{1411117222, 505693}","NULL",0,0.000026
```

Figure 5.17(a) Delete the File

DroidDream

```
30610 5071,1411032978.812471,"futext",0,3,"0x40f6a250",0x81 /* FUTEX ??? */,"1",0.000035
30611 5071,1411032978.812629,"stat64",0,2,""/data/data/com.droiddream.sex/files/mobclick_agent_cached_com.droiddream.sex""",{"st_mode=S_IFREG|0660, st_s
30612 5071,1411032978.813268,"unlink","IO",0,1,""/data/data/com.droiddream.sex/files/mobclick_agent_cached_com.droiddream.sex""",0.000119
30613 5071,1411032978.813710,"getpid",0,0,"",5041,0.000035
```

Figure 5.17(b) Delete the File

DroidKungfu

```
55331 5023,1410660823.027913,"close","IO",1,1,"3",0.003154
55332 4926,1410660818.613158,"ioctl",1,3,"10",0xc0186201,0x52046e28,0,4.417981
55333 5023,1410660823.033140,"unlink","IO",0,1,""/data/data/eu.chainfire.supersu/requests/901508""",0.004724
55334 5023,1410660823.038185,"open","IO",0,2,""/system/usr/share/zoneinfo/Asia/Kuala_Lumpur""",O_RDONLY|O_LARGEFILE,0.000037
```

Figure 5.17(c) Delete the File

It can be concluded that malicious activities is identified when the malware was trying to accessing file, accessing database, change file properties and others. Thus, the common traces from system call are shown in **Table 5.2**.

Table 5.2 List of Traces in System Call

Traces
Determine file accessibility
Open the file
Get the information of file
Change permission mode
Execute the file
Write to the file
Read the file
Rename the file
Receive a message from a socket
Delete file

5.3.3 Process of Collecting Traces from Logcat

The logcat shows messages in real time and also keeps a history thus it can be viewed anytime. In this logcat, malicious activity is identified as shown in **Table 5.4:**

Table 5.4 List of Traces in Logcat

Parameter
PID
Priority
Message

First, malware apk file is installed and launched in Genymotion Android Emulator. Some activities such as phone call, texting and browsing had made. Next, application is terminated and the data is collected. The data is opened with NotePad ++ and read line by line as to identify the attribute(traces) of the malware. The malicious activities are determined based on priority, message and PID. If there is no any malicious activity,

repeat Step 1 to Step 4 until the malicious activity is found. The process of collecting logcat data is simplified as shown in **Figure 5.18**.

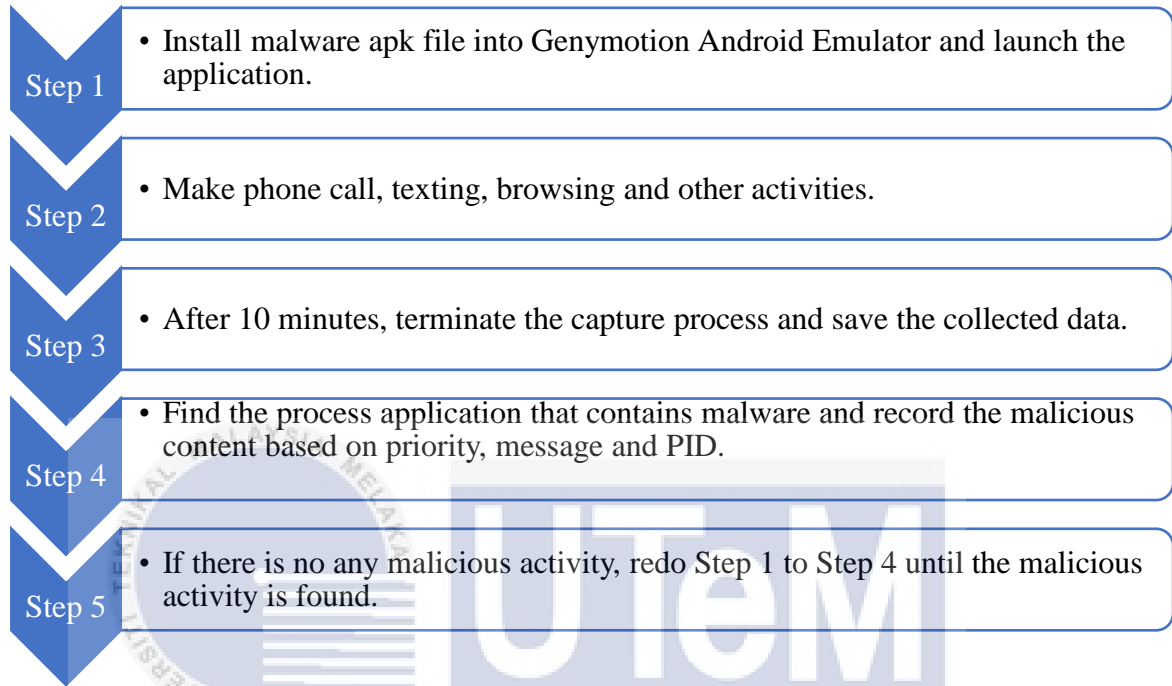


Figure 5.18 Steps of Collecting Data from Logcat

Based on **Figure 5.19(a)**, **Figure 5.19(b)** and **Figure 5.19(c)**, there are several priorities that include in this project which are I, W and D. I is info while W is warning and D is debug.

```
[ 09-18 15:58:19.320 2034: 2076 I/PackageParser ]
com.droiddream.bowlingtime: compat added android.permission.WRITE_EXTERNAL_STORAGE
```

Figure 5.19(a) Priority, Message and PID (DroidDream)

```
[ 09-19 16:56:54.239 2028: 2075 W/PackageManager ]
Not granting permission android.permission.READ_LOGS to package com.keji.danti623 (protectionLevel=50 flags=0x8444)
```

Figure 5.19(b) Priority, Message and PID (AnserverBot)

```
[ 09-14 11:32:34.565 10892:13893 D/PackageBroadcastService  
Received broadcast action=android.intent.action.PACKAGE_ADDED and uri=com.evilsunflower.farmer
```

Figure 5.19(b) Priority, Message and PID (DroidKungfu)

The tracing processes begin at network traffic followed by system call and logcat. This procedure is to identify the traces left in the sources. The tracing procedure for tracing the incident traces from network traffic begins with tracing the traces attributes mentioned in the proposed traceability matrix. The traces attributes are IP address (destination IP address and source IP address), port number (destination port and source port), time, protocol as well as domain name.

To show the correlation or relationship of the evidence of the incident discovered during the investigation process, the incident traces found from the tracing procedures are mapped.

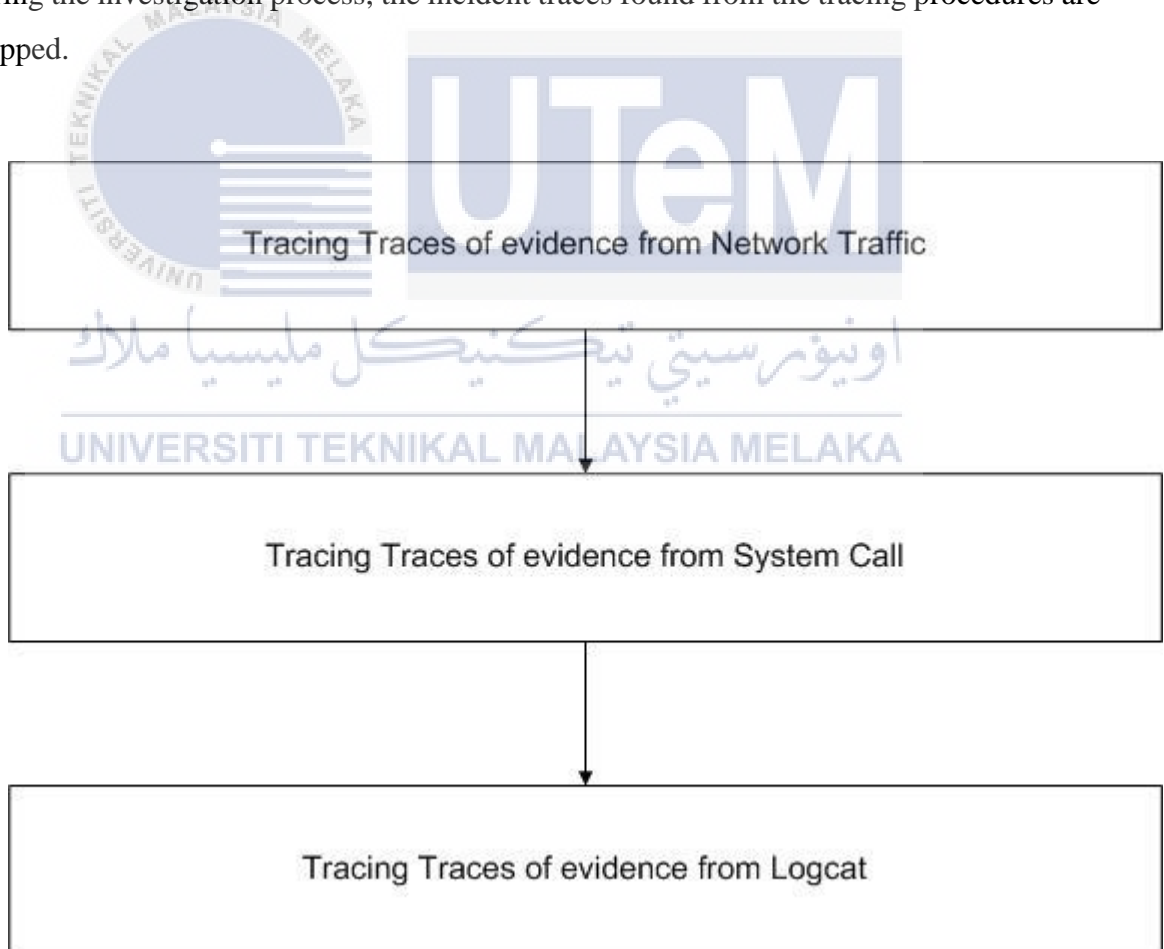


Figure 5.20 Tracing procedures for tracing evidence of malware incident.

As shown in **Figure 5.20**, the mapping procedures of the incident traces discovered from the tracing process. At first, the traces that are discovered from the tracing process are mapped within sources. The traces discovered in network traffic are mapped with the traces discovered in the same log. Second, the traces that are mapped from network traffic are mapped to the traces discovered in system call. Finally, the traces mapped from network traffic and system call are further mapped to logcat as summarized in **Figure 5.21**.

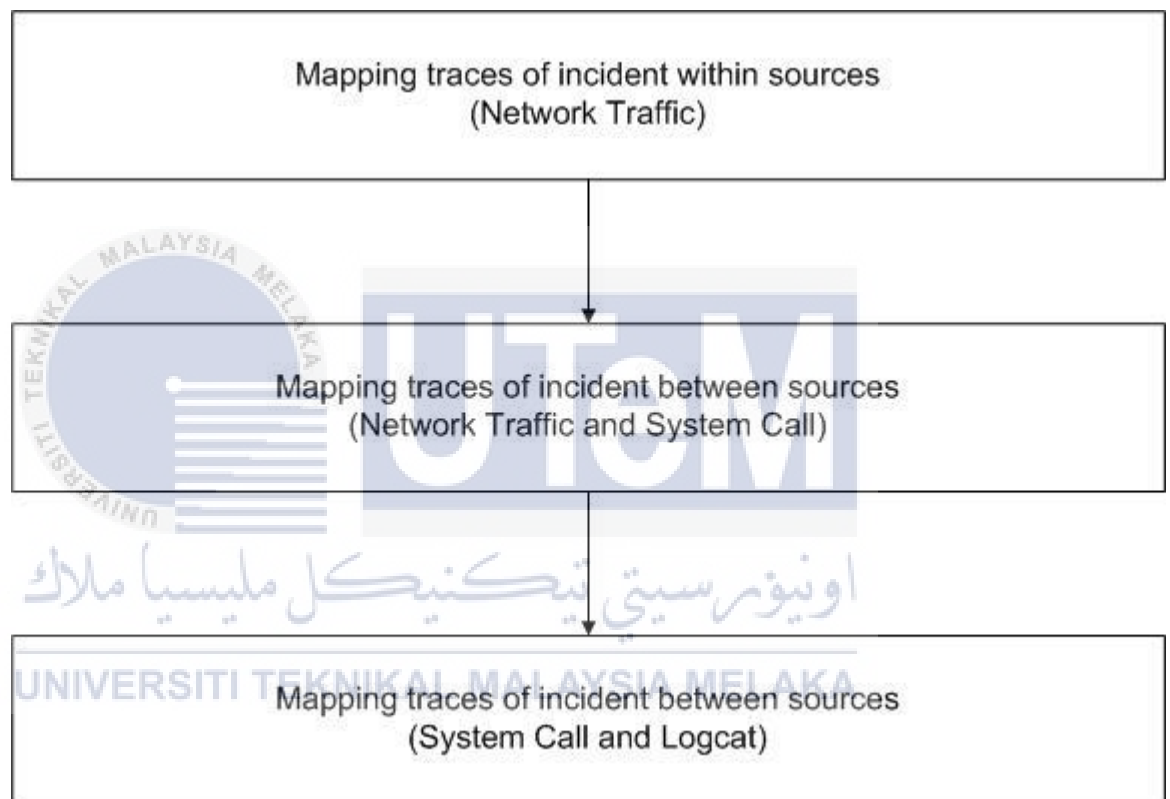


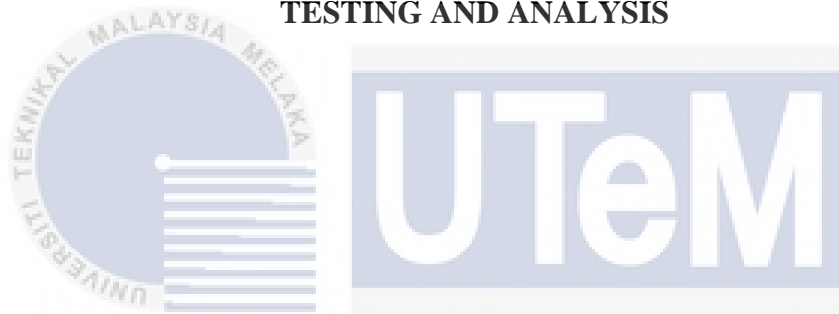
Figure 5.21 Mapping procedure for incident traces

5.4 Conclusion

Generally, this chapter discussed on experimental set up and process of collecting data in network traffic and system call. Besides, the malicious activities in both sources are identified. The result on this project will be discussed in the next chapter which is Chapter 6.

CHAPTER VI

TESTING AND ANALYSIS



6.1 Introduction

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

In previous chapter, process of collecting data from system call, network traffic and logcat in this project had been described. For this chapter, a discussion on conducting analysis and testing will be explained.

6.2 Test Organization

The testing plan is organized to make sure the testing of the data set can be done in ordered way and able to achieve the objective of testing. The function of the traceability matrix is to map and trace the origin of digital crime. The purpose of testing phase is to test the usability of the traceability matrix by using another set of data. If the analysis result in Chapter 5 is similar as the output of traceability matrix, it means that the data had successfully tested.

6.3 Test Design

As mentioned earlier, the goal for testing phase is to test the usability and effectiveness of the traceability matrix. The effectiveness of this project is defined as the accuracy and the completeness whether this project achieves objectives or not while the usability is defined as the ease of use of the proposed traceability matrix as it compiled all the sources of the evidences in a file.

Firstly, network traffic is analysed and http request is filtered as shown in **Figure 6.1**. Next, TCP Stream is followed as displayed in **Figure 6.2**. From **Figure 6.2**, domain name is b4 cookier.co.cc:8080 and malicious file is com.keji.danti623. After that, malicious file is searched in system call as shown in **Figure 6.3** and it can be seen that the malicious is trying to access first_app_preferences.xml and the PID is 6667. Based on PID (6667)in logcat as displayed in **Figure 6.4**, the malicious file was debugged and the process named DebugListener.

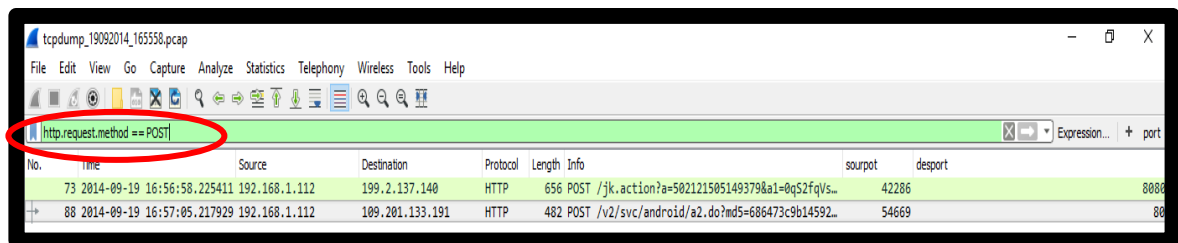


Figure 6.1 Filter HTTP Request


```

Wireshark · Follow TCP Stream (tcp.stream eq 2) · tcpdump_19092014_165558
POST /jk.action?a=502121505149379&a1=0qS2fqVs0qSLfqlBfNrB8c=5&c1=01__&d=4.1.2&d1=0vQs9wV_&e=samsung&e1=rCkgr430zY_&f=GT-
P6800&f1=il1gyJ7QfJS_&g=8&g1=PS_&h=351522052440107&h1=fNys0qV2fJy2093_&f4&i=16&i1=fq7_&j=1411117017296&j1=fqlsfqjs0NSs0NVB0
n_&k=0&k1=fS_&l=0&l1=fS_&m=0&m1=fS_&n=0&n1=fS_&p=com.keji.danti623&p=7CMg9IgrRHID0ztkOutDCfwf_&key=fqlsfqjs0NSs0Nrcf1_
HTTP/1.1
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.2; GT-P6800 Build/JZ054K)
Host: h4.cookieer.co.cc:8080
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Type: application/x-www-form-urlencoded
Content-Length: 0

```

Figure 6.2 TCP Stream Result

```

6667.141117037.450045."futex".1.3."0x400a56bc".0x81."FUTEX ???"/",1,,,,1,0.001481
6667.141117037.452281,"access","IO",0,2,"/data/data/com.keji.danti623/shared_prefs/first_app_perferences.xml","F_OK",,-1,0.000094
6667.141117037.452708,"open","IO",0,3,"/data/data/com.keji.danti623/shared_prefs/first_app_perferences.xml","O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE",0600,,,,74,0.000155
6667.141117037.453104,"fstat64",,0,2,74,"{st_mode=S_IFREG|0600, st_size=0, ...}",,,,0,0.000037

```

Figure 6.3 System Call Result

```

[ 09-19 16:57:05.549 6667: 6697 D/debugListener ]
str==当前服务器返回码是:200

```

Figure 6.4 Logcat Result

6.4 Test Result

The test is done using GoldDream Android Malware. Basically, GoldDream has the capability to logs all incoming and outgoing phone calls and received SMS. The log captured is then sent to an external server. Device's information such as IMEI number, phone number and device's model are also captured and send to an external server. **Table 6.1** shows result from network traffic. It is proven that the device's information are captured and send to an external server which is ade.wooboo.com.cn. **Table 6.2** shows the result from system call. It is proven that the malware is associating with sms logging and accessing file. Table 6.3 shows the result from logcat. There are warning message and debug activity.

Table 6.1 Result from Network Traffic

NO	IP ADDRESS		PORT		METHOD	URL	Description
	SOURCE	DESTINATION	SOURCE	DESTINATION			
N1	192.168.1.111	122.11.61.106	35979	80	POST	ade.wooboo.com.cn	Device model: GT 9600 IMEI: Phone number: 60126731249
N2	192.168.1.111	122.11.61.106	54663	80	POST	ade.wooboo.com.cn	

Table 6.2 Result from System Call

NO	Access()	Open()	fstat()	Chmod()	Write()	Execve()	Rename()	Unlink()	Read()	Recv()	Malicious file	PID
S1	✓	✓	✓	✓				✓			com.GoldDream.pg03	5489



Table 6.3 Result from Logcat

NO	PID	PRIORITY	MESSAGE
L1	5489	I	WebClipBoard
L2	5489	I	Ads
L3	5489	W	webview_proxy
L4	5489	D	WebCore

6.5 Conclusion

As a conclusion, this chapter had discussed in-depth on result and analysis of this project. This chapter presented the proposed traceability matrix. Next chapter which is the last chapter in this project, a holistic conclusion will be discussed.



CHAPTER VII

PROJECT CONCLUSION



اونيورسيتي تيكنيكل مليسيا ملاك

7.1 Introduction

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

The whole analysis of this project will be summarized in this chapter. Besides, this chapter also included project contribution, limitation and future works.

7.2 Project Summarization

Nowadays, Android is replacing the need for a personal computer because it has same capabilities of traditional computer particularly when it offers the connectivity to various universal service. Furthermore, people tend to store as well as organize a vast range of business and personal, also make them constantly connected to the Internet. Even though the amount of data stored in Android is less than the amount of data stored in computer, the small amount of data can be a useful, crucial and beneficial

information. Thus, all of this information inside Android phone can be used as a source of evidence in digital forensic investigations.

Nevertheless, the ongoing increment of popularity of Android phone has cause a problem in discovering complex and enormous volume of evidence. Consequently, this bring to the difficulty in cross referencing and linking meaningful correlation between the incident events. In the past, only the scope of an attack became the main idea of the study rather than the actual attacker and assessing the liability of an organization without dealing with issue of origin identification and cross referencing in investigation process.

As a consequence, this project will construct the traceability matrix in android forensic investigation to identify the correlation between the components of forensic investigation and the incident event discovered. As to make it real, this project began with conducting an experiment, collecting and analysing the data that affected with android malware attack. The traceability matrix is constructed after the attributes and the correlation of the attack are identified. The construction of the traceability matrix is important to ensure the sources of evidences are fully covered and to understand the relationships of incident evidence in android forensic investigation process to fulfil the needs of defense industries and the law enforcer in identifying the sources of evidence.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

7.3 Project Contribution

As to establish the relationship between the sources and evidence traced, this new traceability matrix for digital forensic investigation will assist the practices (Digital forensic investigator, PDRM, CyberSecurity Malaysia). Hence, this traceability matrix helps the pratices in forwarding and present the evidence in the court of law in order to identify the offender of the crime.

7.4 Project Limitation

As we know every project has its limitation and constraint, for this project, the limitation and constraint are additional effort required to create and maintain the traceability matrix and update it regularly. Besides, capturing requirements traceability becomes complex and expensive as the source of evidence grows in size and complexity. In addition, manual traceability methods can lead to incorrect requirements traceability data.

7.5 Future Works

In future, the effectiveness of the evidence tracing is evaluated through calculation or algorithm to make the result more accurate. Besides, the number of sources also can increase the effectiveness of the traceability matrix.

7.6 Conclusion

As a conclusion, the traces have been identified and the traceability matrix has been constructed. Thus, the objectives of this project have been achieved.

REFERENCES

- Aquilina, J. M., Casey, E., & Malin, C. H. (2008). *Malware forensics: Investigating and analyzing malicious code*. Burlington, MA: Syngress Pub.
- Nelson, B. (2006). *Guide to computer forensics and investigations* (2nd ed.). Boston, MA: Thomson Course Technology.
- Graham, J., & Howard, R. (2009). *Cyber fraud: Tactics, techniques, and procedures*. Boca Raton, FL: Taylor & Francis.
- Sammons, J., & Safari Technical Books. (2012). *The basics of digital forensics: The primer for getting started in digital forensics*. Amsterdam: Elsevier/Syngress.
- Skoudis, E., & Zeltser, L. (2004). *Malware: Fighting malicious code*. Upper Saddle River, NJ: Prentice Hall PTR.
- 2016 Internet Security Threat Report. Retrieved from Symantec Corporation 2016 Website: <https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf>
- Rover, R. (2015). *Threat Report 2015*. Retrieved from F-Secure Corporation 2016 Website: https://www.f-secure.com/documents/996508/1030743/Threat_Report_2015.pdf
- Arora, A. (2014). *Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices*, 66–71. <https://doi.org/10.1109/NGMAST.2014.57>
- Yuhui, F., & Ning, X. (2015). *The analysis of android malware behaviors*. International Journal of Security and Its Applications, 9(3), 335–346. <https://doi.org/10.14257/ijseia.2015.9.3.26>
- Zaki, M., Sahib, S., Abdollah, M. F., Selamat, S. R., & Ahmad, R. (2013). *Profiling Mobile Malware behaviour through Hybrid Malware analysis Approach* 2013 9th International Conference on Information Assurance and Security (IAS), 78–84.

APPENDICES

1. HOME

ANDROID MALWARE TRACEABILITY MATRIX IN DIGITAL FORENSIC INVESTIGATION				
	DATA NAME	Source 1: Network Traffic	Source 2: System Call	Source 3: Logcat
TEST CASE ID				
T1		<u>N1</u>	<u>S1</u>	<u>L1</u>

2. NETWORK TRAFFIC

SOURCE 1: NETWORK TRAFFIC

[HOME](#)

NO	IP ADDRESS		PORT		METHOD	URL
	SOURCE	DESTINATION	SOURCE	DESTINATION		
N1						
N2						
N3						
N4						
N5						
N6						
N7						
N8						
N9						

3. SYSTEM CALL

SOURCE 3: SYSTEM CALL

[HOME](#)

NO	Access()	Open()	fstat()	Chmod()	Write()	Close()	Rename()	Unlink()	Read()	Recv()	Malicious file	PID
S1												
S2												
S3												
S4												
S5												
S6												
S7												
S8												
S9												
S10												

4. LOGCAT

SOURCE 3: LOGCAT

[HOME](#)

NO	PID	PRIORITY	MESSAGE
L1			
L2			
L3			
L4			
L5			
L6			
L7			
L8			
L9			
L10			