Static and Dynamic Analysis of Android Mobile Malware

Ana Patricia Becerra
Master of Engineering in Computer Engineering
Jeffrey Duffany, Ph.D.
Electrical and Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— In the last years, mobile malware has become a serious threat to thousands of users. The massive increase in the use of smartphones with the Android platform makes the need for malware analysis of this platform a critical issue. It's necessary though, to understand how the Android Malware works, and also to find out how to defend this platform from malicious attacks. In this project, we use the reverse engineering as a tool to understand the structure and functionality of the malware. We will define the characteristics of the android malware through the objectives that it was created for. As an example, we present the results of reverse engineering of NotCompatible because the malware has the functionality of infected 20,000 devices for day and attack the android markets over and over in the last three years, and Arspam, which was designed for political purposes to recognize the orientation of the prayers.*

Key Terms *— Android Malware, Malicious Code, Mobile Malware, Smartphone Security.*

INTRODUCTION

During the last three years, Android OS has incremented its popularity becoming the most preferred mobile by users.

This has brought a consequence that with malware for android mobile, cybercriminals have discovered a new business model.

The phenomenon of malicious code aimed at android mobile is starting to developed more complexity. The attacks of android mobile have actually increased in numbers on the last two years.

The evolution of cybercrime found another target to attack of android mobile. The malicious authors can to incorporate their attacks in the android application without users noticing.

Understand what the methodology uses to change or add malicious code into de android applications can allow us to introduce new security tools on android mobile.

In this paper, we learned malware analysis as a process where we will study its code structure, operation and functionality.

Among of the objectives of this project are:

- Understand the different tools used to reverse engineering of malware for android platform.
- Penetrate into the used data to know how the android platform is compromised.
- Study different attacks to try to get similar features that allow you to obtain patterns between them.

Our goal is to understand how Android malware works. For them, we created an isolated virtual environment, and we acquired more than 121 examples of android malware but we used the most important and sophisticated.

BACKGROUND

Android is an operating system initially developed by Android Inc., a firm purchased by Google in 2005, in collaboration with the Open Handset Alliance.

Open Handset Alliance is an alliance of dozens of organizations committed to bringing "better" and more "open" mobile phone to market [1]. This operating system is based on a modified version of the Linux kernel. Unlike other mobile operating systems like iOS or Windows Phone, Android is developed in an open and accessible to both the source code and the list of incidents which are reported unresolved problems and new problems to report.

Therefore, Android is the first open source mobile application platform that provides a base system, an application middleware layer, a Java software development kit (SDK), and a collection of

system application [2]. Definitely to be developed in an open source, this is an advantage for those who develop their applications as their users. You can customize and modify the maximum phone functions by simply installing an application.

Another advantage of Android is the incredible confidence that you are receiving from manufacturers. As a result, this platform has become the most used in what refers to mobile devices.

ANDROID PLATFORM

Android is born of the union of the Linux operating system and a Java-based platform called Dalvik [3]. Basically, software developers write their applications in the Java programming language and tools of Google, for example the Android SDK, allows Java programs running on the platform Dalvik on Android devices. It is unclear why Google chose to use a non-standard Java platform (the machine Dalvik) to run their applications. Possibly it was to avoid patent infringement.

Every Android application runs on its own virtual machine, like Java applications do, and each virtual machine is isolated in its own Linux process. This ensures that no process model can access the resources of any other process (unless the device is unlocked).

While the Java virtual machine was designed to be a safe place, sandboxed, that is a system capable of containing potential malware, Android is not based on your virtual machine technology to intensify its security. Instead, all the protection is based directly on the Linux-based operating system.

The Android security model is based primarily on: traditional access control, isolation, and a security model based on permissions.

However, it is important to note that the security of Android does not rely only on the implementation of their software. Google released the source code to complete programming for all ages, allowing the project to get analysis of all security Android community. Google argues that this openness helps discover defects and leads to improvements in security.

ANDROID APPLICATION

The Android applications mostly are written in Java and use XML files for its configuration [4]. The XML file is known as AndroidManifest.xml. The Android compiler is the Dalvik VM that it compiles the Java files into class file and after that, into dex files, which are bytecode. Like the dex file, the xml files also are converted to a binary format. Both dex and xml files and other resources are packed into .apk file. This .apk package is signed with a developer's key and uploaded to the android market for its distribution.

Figure 1 shows the Angry Birds game for android opening with the 7-Zip file program. As displays, the android application have the extension .apk but it is just ZIP files. The .apk contains AndroidManifest who declares which permissions to access the application will have to operate, also the resources and the classes.

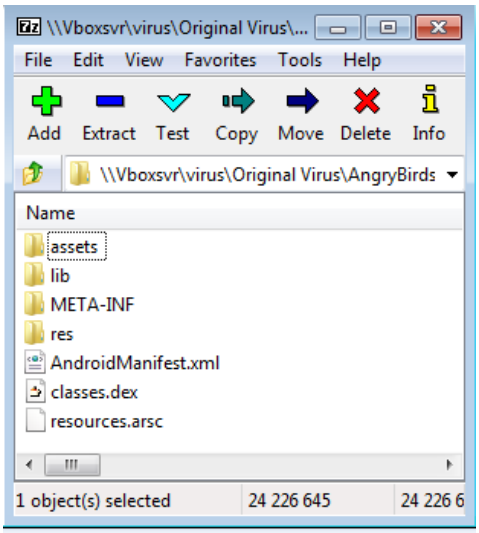

Figure 1
Inside the .apk

THE REVERSE ENGINEERING TOOLS

There are different tools for reverse engineering that can be used to understanding the mobile malware, but for purpose of our project, we used Dex2Jar [5] to decompile the classes.dex in a Java byte code. We can open this classes.dex into JD-GUI [6]. Apktool [7] is used to dissemble the manifest.xml file and WinMerge [8] to compare the

apk-file infected with the original apk-file, where all of them run in a virtual machine (VirtualBox Program) [9] on Windows (Windows 7) environment.

Android Malware Characterization

In this section, we describe a brief characterization of existing malware according to their target attack, but we want to mention that other authors in the area characterize the existing android malware for its installation, activation or/and malicious payload [10].

In our research, we wanted to emphasize that when we write about of the android attacks it is important to consider what the malware authors wants to achieve with the malicious application. After analyzing the existing android malware we can characterize the malicious payload into four different categories: privilege escalation, remote control, financial charges and personal information stealing [10].

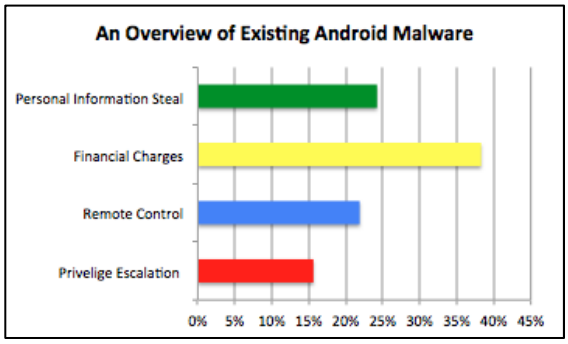

Figure 2
An Overview of Existing Android Malware

In the existing malware recollected el 38% of the families malware was designed to cause finance charges to infected users. If we compare this percentage with the percentage analyzed in 2012 for Mobile-Sandbox in over 300,000 Android applications they reported the following distribution of malicious behavior [11]:

- 51.3% Steal personal information
- 30.1 Send Premium rated SMS messages
- 23.5 Characteristics of a Botnet
- 18.3% Contain Root Exploits.

These attacking objectives are not exclusive between them, its like saying that in an Android Malware can exist various attacking objectives. We would like to highlight that a lot of times steal personal information or the personal profile from the users represents a lot of money for the social media company. Steal personal information includes address book entries, IMEI, GPS position of the user between others

The second distribution of malicious behavior mentioned is sending SMS messages rates with 30.1%, most common to make money immediately is sending these messages to premium rated. Another malicious behavior that seems important to us is that 23.5% of malware families have the ability to connect to a remote server to receive and execute commands that is what we know as botnet.

Common Techniques of Infect Apps

One of the common techniques that malware authors use to insert malicious codes into the apps is the repackaging. The repackaging consist in download popular apps dissembles them with the tools mentioned in the section Reverse Engineering tools, enclose malicious contents, and then re-assemble and submit the new apps to the official or other different Android Markets [10].

Yajin Zhou and Xuxian Jiang found in your research that the total of 1260 malware examples, 1083 of them are repackaged. The 86% of the total malware collected was infected by this technique, in addition it was found that the malware authors have chosen a variety of apps for repackaging that include paid apps, popular games apps, powerful utility apps, as well as porn related apps [10].

AnserverBot is an example of repackaging and was considered as one of most sophisticated Android malware because it employs several sophisticated techniques to evade detection and analysis [12]. Other examples of repackaging include BaseBridge, CoinPirate, DogWars, DroidDream, DroidKunFu, Geinimi, GingerMaster, and Zsone. The popular Angry Bird app was infected with the repackaging

technique and when you compare the Angry Bird Original download for the Official Android Market with the infected app download of the third party app markets. Just comparing the AndroidManifest.xml of both apps you may notice that the package is called different in the app infected. Rovio Entertainment creates the Angry Birds app and the structure that uses for its package are:

- com.rovio.angrybirdsspaceHD;
- com.rovio.angrybirdsspace.ads;
- com.rovio.angrybirdsrio;
- com.rovio.angrybirds;
- com.rovio.angrybirdsseasons;
- com.rovio.angrybirdsspace.premium

As show in Figure 3 and Figure 4, in the Angry Birds app infected the package call com.rovio.new.ads.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1010" android:versionName="1.0.1" android:installLocation="auto" package="com.rovio.angrybirdsspace.ads"
```

Figure 3
Angry Birds Original Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1112" android:versionName="1.1.2" android:installLocation="auto" package="com.rovio.new.ads"
```

Figure 4
Angry Birds Infected Manifest File

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_LOGS" />
```

Figure 5
Modified Permission in the Android Manifest File

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Figure 6
The Original Permission in the Andoid Manifest File

Figure 5 and Figure 6 show other thing that you can observe is that the only permission that is not included in the original app is the READ_LOGS. The READ_LOGS allows an application to read the low-level system log files. The READ_LOGS permission is not granted to thirty-party apps anymore with the version Android 4.1, JellyBean.

The second technique that is made it difficult for detection is update attack. This technique, instead of enclosing the payload as a whole, it only includes an update component that will fetch or download the malicious payloads at runtime. There are four malware families that adopt this attack: BaseBridge, DroigKunFuUpdate, AnserveBot, and Planton [11].

The third technique applies the traditional drive-by download attacks to mobile space [11]. The malware is downloaded when users are redirected to the malicious website. Examples of families malware under this technique are GGTracker, Jifake, Siptmo and Zitmo. The last two are created to steal user's sensitive banking information.

CASE STUDIES: NOT COMPATIBLE AND ARSPAM

On this section will make a brief analysis of two malware that attacked the operational system of Android this year. The first is called Not compatible, and we chose it for our analysis because is not the first time that this kind of attack infects Android devices and also this malware has the capacity to infect 20,000 devices per day. The other malware that we will analyze is Arspam in which infects through the Alsalah app and what it does is secretly send text messages with links to political.

NotCompatible

In the middle of 2012 it was discovered in the android platform the malware called NotCompatible [13]. This malware is designed to infect Android mobiles and turn them into unwitting Web proxies. The malware pretends to be a system update in order to get unwitting users to install it. The system seems to give access to protected networks through infected Android devices. NotCompatible was named for its apparent command-and-control server (C&C) with domain notcompatibleapp.eu

To understand what this malware do, we need to look first the information in the AndroidManifest.xml. The AndroidManifest.xml contains basis information of Android applications, such as permissions, activities and services.

Unlike other malware NotCompatible not asking a lot of permits and also not used an exploit to get root permissions. To execute request access to the Internet, network state, and is also notified when it completes the system startup as show in Figure 7 under number 1. Also the Figure 7 we can observe under number 2 that the file contents of AndroidManifest.xml are filtered by start Operating System (BOOT_COMPLETED) or user interaction (USER_PRESENT) to trigger the execution of OnBootReceiver Class that will start the execution of the risk as show in Figure 8 into de classes.dex open with the Java Decompiler. These two events in particular execute the threat in the system. BOOT_COMPLETED ensures that the malicious code is executed when the system startup and USER_PRESENT can allow the malicious code to interact with the user. The target of this malware is no steal information from the user but seeks to trick the user for installing bogus security updates in the system which could lead to infection of different types of threats by using Social Engineering.

The main goal of this malware is to create a connection to a remote server from where to download suspected updates and receives commands.

Another important point of this malware is the file that contains the data connection and its content encrypted. The static analysis of the Trojan allows us to see how the process loads the configuration file information (see Figure 9 public void on Create and Config class that contains the parameters by default), and get the key that allows to access to the configuration data.

In the constructor of Config class, as show Figure 10 show how the hash function is applied to the key. In this case the key to encrypted the information is "ZTY4MGESYQo" and finally, uses encryption algorithms AES to decrypt the file with the connection data.

Arspam Alsalah

This malware is designed to target particular devices of the Middle East region, particularly those with high presence of Muslims, because it install itself as a useful application to recognize the orientation, which addressed their prayers. The target of malware is political.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="1.0" package="com.Security.Update"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <application android:debuggable="true">
        <service android:name=".SecurityUpdateService" android:enabled="true" />
        <receiver android:name=".OnBootReceiver" android:enabled="true" android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
                <action android:name="android.intent.action.USER_PRESENT" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

1

2

Figure 7
NotCompatible AndroidManifest.xml

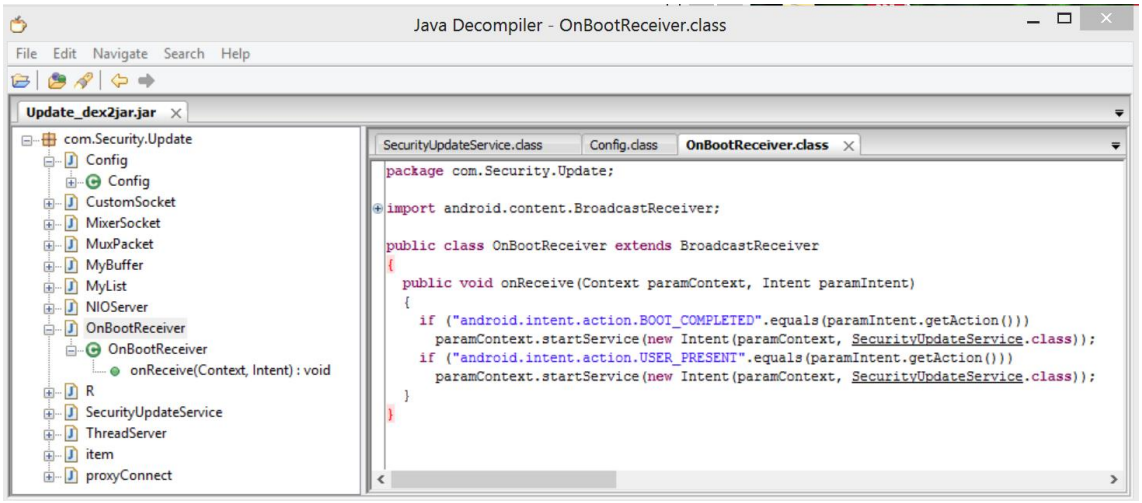

Figure 8
NotCompatible Class.dex

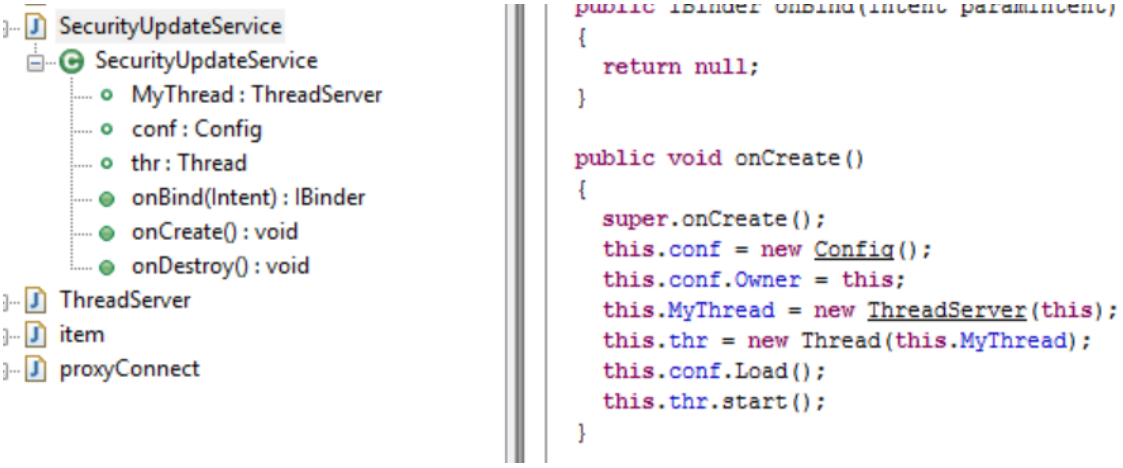

Figure 9
SecurityUpdateService Class

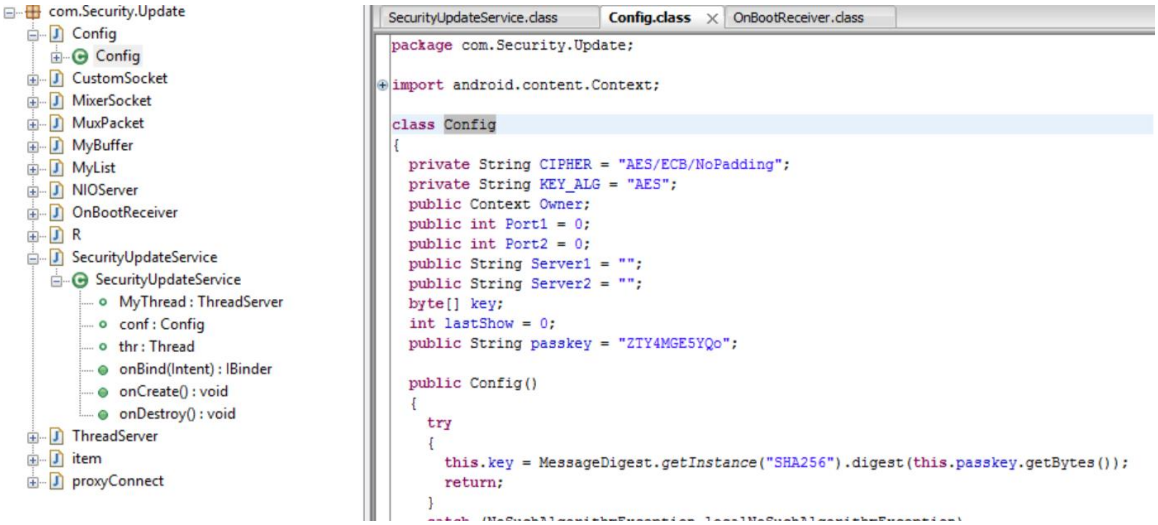

Figure 10
Config Class

When installing the application it request the following massive set of permission:

INTERNET
ACCESS_FINE_LOCATION
ACCESS_NETWORK_STATE
INTERNET
WRITE_EXTERNAL_STORAGE
READ_CONTACTS
CHANGE_WIFI_MULTICAST_STATE
CLEAR_APP_USER_DATA
BIND_INPUT_METHOD
WRITE_CONTACTS
CLEAR_APP_CACHE
AUTHENTICATE_ACCOUNTS
READ_PHONE_STATE
SET_PREFERRED_APPLICATIONS
INTERNAL_SYSTEM_WINDOW
MANAGE_ACCOUNTS
PERSISTENT_ACTIVITY
FLASHLIGHT
ACCESS_NETWORK_STATE
ACCESS_MOCK_LOCATION
SEND_SMS
HARDWARE_TEST
ACCESS_CHECKIN_PROPERTIES
DISABLE_KEYGUARD
READ_SYNC_STATS
READ_INPUT_STATE
EXPAND_STATUS_BAR
BLUETOOTH
BIND_APPWIDGET
ACCESS_LOCATION_EXTRA_COMMANDS
BROADCAST_SMS
DIAGNOSTIC
BLUETOOTH_ADMIN
DEVICE_POWER
CHANGE_CONFIGURATION
DELETE_PACKAGES
BROADCAST_WAP_PUSH
REBOOT
WRITE_SMS
ACCESS_WIFI_STATE
ACCESS_COARSE_LOCATION
STATUS_BAR
MOUNT_UNMOUNT_FILESYSTEMS
GLOBAL_SEARCH
READ_SMS
CONTROL_LOCATION_UPDATES
MANAGE_APP_TOKENS
DELETE_CACHE_FILES
BATTERY_STATS
READ_SYNC_SETTINGS
SET_TIME_ZONE
READ_HISTORY_BOOKMARKS
MOUNT_FORMAT_FILESYSTEMS
SIGNAL_PERSISTENT_PROCESSES

MASTER_CLEAR
READ_LOGS
BRICK
SET_ACTIVITY_WATCHER
RECEIVE_SMS
GET_ACCOUNTS
CALL_PHONE
READ_CONTACTS
RESTART_PACKAGES
READ_CALENDAR
RECEIVE_BOOT_COMPLETED
CAMERA
ACCESS_FINE_LOCATION
SUBSCRIBED_FEEDS_READ
WAKE_LOCK
RECORD_AUDIO
INSTALL_PACKAGES
INJECT_EVENTS
RECEIVE_WAP_PUSH
USE_CREDENTIALS
ACCOUNT_MANAGER
SET_ALWAYS_FINISH
RECEIVE_MMS
WRITE_SECURE_SETTINGS
MODIFY_AUDIO_SETTINGS
WRITE_CALENDAR
WRITE_SYNC_SETTINGS
INSTALL_LOCATION_PROVIDER
SYSTEM_ALERT_WINDOW
MODIFY_PHONE_STATE
WRITE_SETTINGS
INTERNET
ACCESS_SURFACE_FLINGER
CHANGE_NETWORK_STATE
CALL_PRIVILEGED
CHANGE_COMPONENT_ENABLED_STATE
DUMP
SET_WALLPAPER
GET_TASKS
WRITE_EXTERNAL_STORAGE
PROCESS_OUTGOING_CALLS
WRITE_OWNER_DATA
WRITE_GSERVICES
SET_WALLPAPER_HINTS
BROADCAST_STICKY
READ_FRAME_BUFFER
GET_PACKAGE_SIZE
FORCE_BACK
UPDATE_DEVICE_STATS
WRITE_APN_SETTINGS
BROADCAST_PACKAGE_REMOVED
SET_ANIMATION_SCALE
SET_ORIENTATION
SET_DEBUG_APP
FACTORY_TEST
REORDER_TASKS
SET_PROCESS_LIMIT
READ_OWNER_DATA
CHANGE_WIFI_STATE
VIBRATE
SUBSCRIBED_FEEDS_WRITE
RECEIVE_BOOT_COMPLETED

This malware is installed through the Asalah, in which is an application that calculates the salah timings. The Trojan will gather the contacts on the compromised device and send each one of the following URLs [13]:
www.dhofaralaezz.com/vb/showthr
www.i7sastok.com/vb/showthr
www.dmahgareb.com/vb/showthr
mafia.clubme.net/t2139
www.4pal.net/vb/showthr
www.howwari.com/vb/showthr
forum.te3p.com/46461
www.htoof.com/vb/t18739
vb.roooo3.com/showthr
www.alsa7ab.com/vb/showthr
www.riyadhmoon.com/vb/showthr
forum.althuibi.com/showthr
www.2wx2.com/vb/showthr
www.mdmak.com/vb/showpo
www.too-8.com/vb/showthr
www.3z1z.com/vb/showthr
www.w32w.com/vb/showpo
forum.65man.com/65man33

CONCLUSION

The focus of this paper was to present the methodology that the malicious author used to

add or modify the android applications. The tools used allow unpacking the android application in a simple way and allow code analysis. One of the most important aspects is the permission that the operational Android system granted when they use and install the app. The usage has the ability to accept or reject the permissions that the app requires. We could think on putting flags inside the Android system that could alert the usage of how much he risk could be to it accept and install that type of application. Some researchers are developing models that can evaluate the potential security risk from untrusted apps by analyzing whether dangerous behaviors are exhibited by these apps [14].

ACKNOWLEDGEMENT

This project would not have been possible without Dr. Jeffrey Duffany, who is my mentor and has encouraged me to continue the work and allow the opportunity to perform it. I would also like to acknowledgment Dr. Alfred Cruz for providing the opportunity to obtain the Nuclear Regulatory Commission (NRC) Grant Fellowship Award NRC-27-10-511.

REFERENCES

[1] Ableson, F., Sen, R., King, C., & Ortiz, C. "User Interfaces", *Android in Action*, Third Edition, 2011, pages 65-101.

[2] Erick, W, Ongtang, M. & McDaniel, P. "Understanding Android Security*", IEEE Security & Privacy Magazine*, vol.7, no. 1, January/February 2009, pp 50-57.

[3] Google Android, "Android the world's most popular mobile platform", Google Inc., Retrieved on 24 August, 2013, http://developer.android.com/guide/basics/what-is-android.html.

[4] Google Android, "Application Fundamentals", Google Inc., Retrieved on 24 August, 2013, http://developer.android.com/guide/basics/what-is-android.html.

[5] Google Android, "User Guide: dex2jar", Google Inc., Retrieved on 23 August, 2013, http://code.google.com/p/dex2jar/wiki/UserGuide.

[6] Google Android, "Get the Android SDK", Google Inc., Retrieved on 23 August, 2013, http://developer.android.com/sdk/index.html.

[7] Google Android,"Andoid-apktool", Google Inc., Retrieved on 22 August 2013, http://code.google.com/p/android-apktool/.

[8] WinMerge Software, "WinMerge", WinMerge Organization, Retrieved on 28 August 2013, http://winmerge.org.

[9] Virtual Box Software, "Virtual Box", Oracle Inc., Retrieved on 24 August, 2013, http://www.virtualbox.org/wiki/Downloads.

[10] Zhou, Y. & Jiang, X., "Dissecting Android Malware: Characterization and Evolution", *SP'12 Proceedings of the 2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 95-109.

[11] Mobile phone forensics and mobile malware, "Our Android Malware Summary for the Year 2012", Forensic blog, Retrieved on 17 September, 2013, http://forensics.spreitzenbarth.de/2013/01/02/android-malware-summary-2012/.

[12] Zhou, Y. & Jiang, X., "An Analysis of the AnserverBot Trojan", 2011, downloaded from the Worl Wide Web, www.csc.ncsu.edu./faculty/jiang/pubs/AnserverBot_Analysis.pdf.

[13] Parkour, Mila, "Contagio Mobile", Contagio Blog, Retrieved on September 1, 2013, http://www.contagiominidump.com.

[14] Grace, M., Zhou, Y., Zhang, Q., Zou, S. & Jiang, X., "RiskRanker: Scalable and Accurate Zero-day Android Malware Detection. *Proceedings of the 10th International Conference on Mobile Systems, Application Services*, June 25-29, 2012, pp. 281-294.