Journal of Computer and Communications, 2014, 2, 8-17
Published Online October 2014 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2014.212002

An Overview of Mobile Malware and Solutions

Fan Wu, Hira Narang, Dwayne Clarke

Computer Science Department, Tuskegee University, Tuskegee, Alabama, USA
Email: wuf@mytu.tuskegee.edu, narang@mytu.tuskegee.edu, dclarke0412@mytu.tuskegee.edu

Received 17 August 2014; revised 10 September 2014; accepted 4 October 2014

Abstract

Mobile Security has been a rapidly growing field in the security area. With the increases of mobile devices and mobile applications, the need for mobile security has increased dramatically over the past several years. Many research and development projects on mobile security are ongoing in government, industry and academia. In this paper, we present an analysis of current mobile security problems and propose the possible solutions to malware threats. Our experiments show anti-malware can protect mobile device from different types of mobile malware threats effectively.

Keywords

Mobile Security, Mobile Malware, Anti-Malware

1. Introduction

Mobile Security has been a growing field in the security area. There are more than 190 countries around the world which use mobile devices [1]. For example, the number of mobile phone users has risen to nearly 200% from Q3/2009 to Q3/2010 [2]. They give people the ability to view a restaurant menu, plan a trip, and play a game and even conduct banking transactions from their mobile devices. Unfortunately, apps are the most common way through which Smartphone and the data stored on them are compromised. As people increasingly download applications, there is an increase chance that other users will gain access to personal information, confirming that security application is becoming a matter of great interest earnest. App downloads are not usually associated with malware in the public consciousness, but it is a growing danger across all mobile platforms and devices [3]. Many mobile devices store sensitive customer and business data [4]. As a result, many federal agencies, companies and institutions are doing research and development in government, industry and academia. Mobile security is going to be a challenging task, and security of mobile devices will create many new job opportunities. A simple task such as just downloading an application or answering an SMS message can be lethal to one's mobile device.

Symantec defines mobile cybercrime as unsolicited text messages that captured personal details, an infected phone that sent out an SMS message resulting in excess charges (typically known as toll fraud), and traditional cybercrime such as e-mail phishing scams. The problem is that while mobile threats may be rising, it's unclear just how prevalent these issues are in the United States. Symantec's statistics, for example, say that 31 percent of mobile users in 2011 received a text message from someone they didn't know or an SMS requesting they click on an embedded link or dial a certain number to get a "voicemail". All of these techniques are tricks the hackers can use to inject malware onto the phone or attempt to trick users into handing over personal data [5]. That is why finding pattern is one of the most primary and important issues in mobile security. So we can find a reasonable and intelligent solution to mobile security in dealing with malware and spyware issue such as, infecting a host and spreading the attack. However, some malware and spyware is disguising itself as some applications, such as play games, banking and so on. The solutions are needed for these issues.

In this paper, we describe our research and experience on current mobile security problems and the development of suitable solutions. We focus on our research on malware. Malware is software programs designed to damage or do other unwanted actions on a computer system [6]. Spyware is the software that "spies" on your computer [7].

The rest of the paper is organized as follows. Section 2 describes mobile security environment setup; Section 3 presents system architecture; Section 4 presents our possible solutions for malware; Our experience results are presented in section 5; Section 6 concludes this paper with suggested future directions.

2. Mobile Security Environment

The mobile security environment comes with different rules and procedures as to how the environment is setup. It follows the base standard formation that expands the user in the hope to solve a problem. The mobile environment needs to setup a system that has all of the components working together from the SDK (Software Development Kit) connecting to the program which connects to APK (Android Application Package) of the mobile device. Other important things to consider are: a successful equipment installation environment testing, and a failed equipment installation environment testing. We will present these factors in the following subsections.

2.1. SDK

SDK is a very important software tool because it allows developers to create the base software structure on mobile devices. It can be considered as the brain of the mobile device, while others may be considered as subunits of the system, which are API (Application Programming Interface), programming documentation of application, and some of the tools that SDK provides for debugging and IDE (Integrated Development Environment).

Now we consider the setup of SDK for mobile security environment. The installation of the SDK is separated into multiple parts. Let us start with the SDK with eclipse. First, we need to download JDK (Java SE Development Kit), which is a tool for JAVA applications development. The JDK includes other tools such as testing, developing, and running Java. Just like with the JDK, ADT (Android Developer Tools) bundle is another important process for SDK and using eclipse. ADT is basically a plugin for eclipse with the purposes to access features in order for the user to develop Android applications. Some of the features ADT bundle provides are trace view, hierarchy viewer, and ProGuard, in the hope to make the development easier for the users. We need to download the ADT bundle, open up eclipse program, and install add repository. During the installation, the user need to check whether the SDK manager is working properly so that the user can install different versions of android interface as well as packages.

Also, when using the SDK with eclipse, our research shows that we need to test if all of the components work together to make sure that we are able to properly run an android project in eclipse. First, we must make sure that all software requirements are completely specified before going for the procedure of testing. The software requirements for the SDK are the android SDK itself, JDK, and eclipse. Without these software requirements, it would be impossible for the android application to work properly. Once, all the software requirements are accomplished, we first create a basic program by starting up eclipse and create a new Android application project. Then, we set the name of the project, target, and package for the project. After the setup for new Android application is completely saved, we then run the project to see the results, which depend on the code that being executed.

For SDK with Netbeans, we have finished the task of downloading JDK, now we need to download JRE (Java Runtime Environment) and Netbeans. JRE is a set of software tools for the applications, which is a part of the SDK. JRE also uses a JVM (Java Virtual Machine), which is a construct layer between the application and platform. JVM allows Java to be portable in order to perform well within mobile platform and hardware applications. Now, we need to configure SDK manager, and Android platform. And then we setup the virtual devices by setting the name of built-in, hardware property, and create an AVD (Android Virtual Devices). We also need to set the plugin for Netbeans, which is an open source IDE for developing programming languages. Some of the components of Netbeans are integrated tools, visual library, and user settings. After the Netbeans plugin is completed, now we are ready to create a project.

2.2. Mobile Device

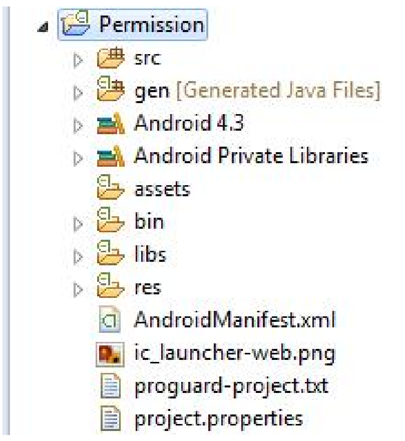
The mobile device is the test platform for mobile security applications. The user can connect the USB plug into the laptop or desktop to get the application just by drag and down or the user can download APK package to install, if the application is available to download in order for the user to connect APK to mobile devices. The mobile devices can come in many different shapes and sizes depending on what the user is trying to accomplish. The devices we used were Motorola Droid X Android version 2.34.

2.3. Programming Tool

The programming tool is the main part of the Android development system. It is a software application that tells the program what to do and how to work. The programming software is included in the OOP (Object Oriented Programming) language for Android mobile devices. OOP language encapsulates attributes and behaviors of objects, as well as inheritance and polymorphism with preset directories and files for each Android application, as shown in **Figure 1**. Java with Android API is easy to use and comes with many tools to help user for applications development.

3. Mobile Malware Infection Channels

The Android system that we have used in our research and implementation is eclipse, which is a worldwide platform for integrating development tools, and is part of JDT. It is one of the open sources IDE, which allows user to create an application for Android. All Android system compatible devices support 32 and 64-bit processing. The entire Android system architecture uses a Linux based system which can be customized. This platform interacts with mobile devices such as phones and tablets. Mobile Malware architecture is illustrated in **Figure 2**. The components of a system are: user accesses site, downloads fake antivirus software, downloads malware, malware transfers funds, transaction approves SMS, and malware forwards approval of SMS. There are many mobile devices, which can be targeted by malware. Especially, the Android system can be attacked by various ways.



```
Permission
    src
    gen [Generated Java Files]
    Android 4.3
    Android Private Libraries
    assets
    bin
    libs
    res
    AndroidManifest.xml
    ic_launcher-web.png
    proguard-project.txt
    project.properties
```

Figure 1. Some preset directories and files.

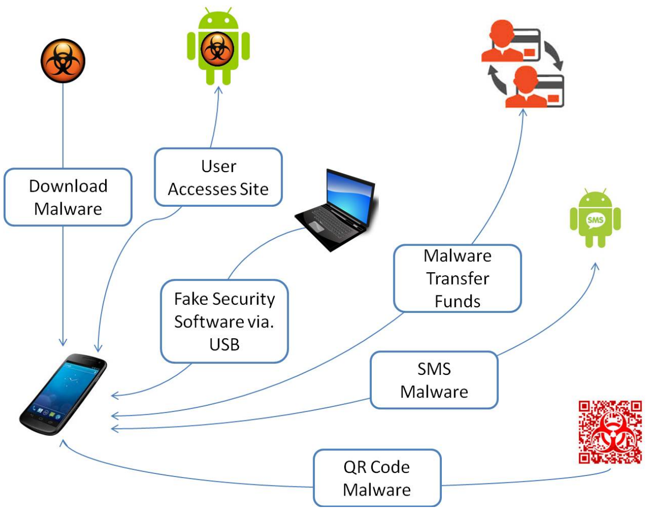


Figure 2. Mobile malware infection channels.

3.1. User Accesses Site

The first mobile malware infection channel, we used within this research was: user accesses site. We look at how a hacker can get access to the user information. User accessing site determines whether or not the users are given privilege. It allows users to register, edit, view profile information, login, and logout of the site. Accounts are set with various permissions depending on the account. Users need to confirm who they are by their username and password. On the mobile device $string controls the permission of supervised nodes, $account checks whether the username is ready to have an account in the system in order for the user to be able to login. $reset resets the user permissions or resets a user account. However, if malware is on the user mobile device or on the non-legitimate site, hackers have access to control user account by login, editing, deleting, or denying user to access their accounts and user information.

3.2. Download Malware

The second mobile malware infection channel, and most important one we used within this research, was downloading the malware. This is the one that gets the most users affected when users mistakenly download malware either directly from non-legitimate website or play store. We downloaded multiple malware implemented in this architecture as they were provided on mobile malware with the requirement we wanted. Other malware was attempted, but they were either online or easy to get rid of. We focused on the most popular malware on the mobile device. It was necessary to download malware because it was easy to see the effects as to how malware acts if it's a direct download to the mobile device. Some of the ways how downloaded malware can affect the mobile device is by preventing mobile device from booting up normally, draining mobile device battery, controlling the user phone, and gaining access to personal information of the user. These attacks are possible when the malware is running on the mobile devices.

3.3. Download Fake Security Software

The third mobile malware infection channel, we used within this research was downloading fake security software. Fake antivirus software is one of the biggest threats to mobile devices. It hides itself within legitimate software, but is actually malware programs that take money from the user in order to fix the computer, phone, or a tablet. Most of the time, fake security software program is installed from the computer to the phone or tablet through USB. Also, this fake antivirus program disables the user's current antivirus program, which makes the malware very difficult to remove. Often, these antivirus programs normally appear when browsing the internet, a display popups and warn the user of fake virus. Another reason why fake antivirus software is so hard to get rid of is that fake antivirus normally updates and controls the latest version. Additionally, if the users already had a virus on their computer, phone, or tablet, they will do anything to get rid of the virus. But some fake antivirus software program will make the user install more malware pretending to be virus solution.

3.4. Malware Transfer Funds

The fourth mobile malware infection channel, we used within this research was, the malware transfers funds. We looked into how hackers can get access to user credit information and gain information through user financial transactions. Malware transferring funds bypassing secure information. This normally happens when a user uses a mobile device to make payments, transaction, or buy something for non-legitimate site or non-legitimate application. The hackers steal user login and password information, not only for banking sites, but also all other sites, where users login to the mobile device. It will empty out the user's bank account. One of the most difficult areas to fix the problem was when the information is sent out using a mobile device. The reason for that is that mobile devices have less security than computer.

3.5. SMS Malware

The fifth mobile malware infection channel, we used within this research was, SMS (Short Message Service). Mobile SMS is one of the most used functions on a mobile device. SMS is a text messaging service which can be used on phones, tablets, and computers. Most of the time, SMS message malware is active from the SMS to the mobile device. Smishing is another type of phishing when a fake message is sent to the user by text message and asks the user for personal information through web link which is a false website, or a phone number. SMS spam is the spam that usually sends the message through an electronic messaging or e-mail, but since we were dealing with mobile device, it was sent through the mobile network. Premium Rate SMS is the most lethal SMS attacks out of the group. The reason why Premium Rate SMS is so lethal is because it attacks a large amount of money in a short time, while the user does not even realize as to what had happened. Premium Rate SMS attacks the same way appears as an infection when the user first purchases some extra phone features for a mobile device without the his knowledge.

3.6. QR Code Malware

The sixth and last mobile malware infection channel, we used within this research was, QR Code Malware for Quick Response. It contained links to websites that had malware hidden inside. Once QR was scanned, the link could send the user to a non-legitimate site or non-legitimate application, without user's knowledge if it had malware downloaded to his mobile device. We researched and tested whether there was any change on the malware, if it was gotten from QR instead of Play Store, with the malware installed by QR malware on the test phone. The hacker found a way to use QR coding to infect mobile device with malware by sending the user to phishing sites, and take the personal information.

4. Malware

Malware is one of the most lethal mobile threats in the mobile security. And malware is set of software programs designed to damage systems of person's computer, phone, or a tablet. Now, common examples of malware are Trojan horses, viruses, worms, and so on. Viruses can cause chaos on the computer, phone, and tablets by deleting all of files or directory within the system. It is unlucky that there are software programmers in the world that have malicious intent, but it is good that most users know about malware. However, most mobile malware is created to disable a mobile device, which allow a malicious users to control the mobile device or even to steal user's personal information which support to be stored on the personal device. Some malware are spyware, Trojan horse, and adware. Spyware is as spy on the systems of the computer, phone, and tablets. Spyware can collect information from the user web browsing history for example, SMS messaging, e-mail, usernames and password, bill payment, and credit card information. If the information is unrestricted, the spyware can transform the information to another user over the mobile device. Now how can spyware get on the mobile device? Spyware is similar to a virus, or infection that can installed when users open an SMS messaging link or e-mail attachment that have malicious software. Another way spyware can be installed into the mobile device when users install another program that have or has spyware within installed program. Because of the dangerous behavior of spyware, most users don't even know when spyware is on their mobile device. Trojan horse steals user personal information and adware display pop-up for personal information and data.

Now the programming and testing to see virus protected can get rid of malware by looking at coding. Within our research we test two malware programs: one is droid kung fu and the other one is DroidDream. These malware applications are very popular for mobile device. Both of them attack the mobile device by taking the user personal information and change how their mobile device behaviors.

4.1. Droid Kung Fu

Droid Kung Fu is a malware that disguises itself as a puzzle game, which can be found on another marketplace. This malware creates a root access of the user's device, which was discovered in 2011. DroidKungFu uses an AES-encrypted "ratc" file (short for "Rage Against The Cage"), that's decrypted at runtime and used to exploit the adb resource exhaustion bug. By doing this, it pulls a privilege escalation stunt and enables it to successfully root any Android 2.1 or Android 2.2 device [8]. However, it is very difficult to detect and removal [9]. The malware has backdoor function that enables attacks by running programs or navigates a specific website without the user's knowledge and steals data of the user as well as collecting all the information from user's stored device. Once the malware is installed, it will collect information such as Android OS version, phone mode, and IMEI number to gain access to the server for the mobile device. Also, Droid Kung Fu turns the phone into a tool that allows attackers to gain access over one's device while snitching important information from its victims. The warning that all downloads from a third-party marketplace are safe for users, is untrue because within a particular application malware is already written in. Frequently, malware is written in the impressive games and tools. The **Algorithm 1** labeled "DroidKungFu Code" shows the code of the infected software getting root access.

```
private void doTimerTask()
{
    long l1 = this.mTickets + 1L;
    this.mTickets = l1;
    if (this.mPermState == 2)
            {
            getPermission2();
            return;
            }
    if (this.mPermState == 3)
            {
            getPermission3();
            return;
            }
    if (!Utils.isConnected(this))
            return;
    long l2 = this.mTickets;
    long l3 = this.mLastTickets;
    if (l2 - l3 >= 60L)
            {
            long l4 = this.mTickets;
            this.mLastTickets = l4;
            doSearchReport();
            return;
            }
    for (int j = 1; ; k = 0)
            {
            SystemClock.sleep(5000L);
            }
    boolean bool2 = localWifiManager.setWifiEnabled(j);
    if (!checkPermission())
            break label248;
    doSearchReport();
    i = 1;
    break;
    boolean bool3 = localWifiManager.setWifiEnabled(1);
    }

private boolean checkPermission()
{
    if (Utils.checkPermission())
    cpLegacyRes();
    for (int i = 1; ; i = 0)
    return i;
}

private void cpLegacyRes()
{
    if (new File("/system/app/com.google.ssearch.apk").exists())
           return;
    try
    {
           StringBuilder localStringBuilder = new StringBuilder("/data/data/");
           String str1 = getApplicationInfo().packageName;
           String str2 = str1 + "/legacy";
           Utils.copyAssets(this, "legacy", str2);
           if (!new File(str2).exists())
                  return;
           boolean bool = Utils.TCP.execute("2 " + str2 + " /system/app/com.google.ssearch.apk");
           return;
    }
    catch (Exception localException)
    }
```

Algorithm 1. Droid Kung Fu Code [10].

The code basically provides instructions to get root access to the mobile device. The program calls one of the getpemission() methods and tries to gain root access. If the malware is successful and gains root access for the user it will, on subsequent timer interrupts, call either doSearchReport (sends a report to the web server) or doSearchTask (request a command to execute [10]. Some of the effects of getpermission() are getpermission1() have on the system are turn off and on the WIFI, and if the checkPermission() is successful, it will only create Search Server. The getpermission2() method tries to install "su" and if that fails, the malware will ask the user for root access. The getpermission3() method adds multiple applications for permissions. However, the malware will still be on the mobile device if it is able to complete its installation, even if it is deleted. So what have been done was, installing an anti-malware, and then downloading the malware Droid Kung Fu. It was then tested to see if the malware Droid Kung Fu could bypass the anti-malware on the mobile device.

4.2. DroidDream

DroidDream is a malware that gains root access to the mobile device in order to get important information from the user's phone. This malware was discovered in 2011. DreamDream was available on the official Android Market. Once DroidDream is installed on one's mobile device, it could also add even more malware application, without the user's knowledge. DroidDream application runs from 11:00 pm to 8:00 am, which is the time when most users are asleep. The **Algorithm 2** labeled "DroidDream Code" shows the codes of the infected software.

```
// Does "/system/bin/profile" exist?
New-instance v2, Ljava/io/File;
Const-string v4, "/system/bin/profile"
invoke-direct {v2, v4}, Ljava/io/File;-><init>(Ljava/lang/String;)
.local v2, f:Ljava/io/File;
invoke-virtual {v2}, Ljava/io/File;->exists()
move-result v4
if-eqz v4, :cond_2e
// If yes, then no need to root the device
```

```
invoke-direct {p0, v5}, Lcom/android/root/Setting;->destroy(Z)
:cond_2d
:goto_2d
return-void

// Else attempt exploid payload
:cond_2e
new-instance v3, Lcom/android/root/udevRoot;
iget-object v4, p0, Lcom/android/root/Setting-;>ctx:Landroid/content/Context;
invoke-direct {v3, v4}, Lcom/android/root/udevRoot;><init>(Landroid/content/Context;)
.local v3, udev:Lcom/android/root/udevRoot;
invoke-virtual {v3}, Lcom/android/root/udevRoot;->go4root()
move-result v4

// Did payload succeed?
if-eqz v4, :cond_3f

// If yes, then don't continue
invoke-direct {p0, v5}, Lcom/android/root/Setting;->destroy(Z)
goto :goto_2d

// If not the attempt rageagainstthecage payload
:cond_3f
new-instance v0, Lcom/android/root/adbRoot;
iget-object v4, p0, Lcom/android/root/Setting;->ctx:Landroid/content/Context;
iget-object v5, p0, Lcom/android/root/Setting;->handler:Landroid/os/Handler;
invoke-direct {v0, v4, v5}, Lcom/android/root/adbRoot;-
><init>(Landroid/content/Context;Landroid/os/Handler;)
.local v0, adb:Lcom/android/root/adbRoot;
invoke-virtual {v0}, Lcom/android/root/adbRoot;->go4root()

.method private destroy(Z)
if-eqz p1, :cond_15

// Check if package "com.android.providers.downloadsmanager" has already been installed
iget-object v0, p0, Lcom/android/root/Setting;->ctx:Landroid/content/Context;
const-string v1, "com.android.providers.downloadsmanager"
invoke-static {v0, v1},
Lcom/android/root/Setting;->isPackageInstalled(Landroid/content/Context;Ljava/lang/String;)
move-result v0
if-nez v0, :cond_15

// If it hasn't, then copy the payload to /system/app/ as "DownloadProvidersManager.apk"
iget-object v0, p0, Lcom/android/root/Setting;->ctx:Landroid/content/Context;
const-string v1, "sqlite.db"
const-string v2, "DownloadProvidersManager.apk"
invoke-static {v0, v1, v2},
Lcom/android/root/Setting;->cpFile(Landroid/content/Context;Ljava/lang/String;Ljava/lang/String;)
:cond_15
invoke-virtual {p0}, Lcom/android/root/Setting;->stopSelf()
return-void
.end method
```

Algorithm 2. Droid Dream Code [11].

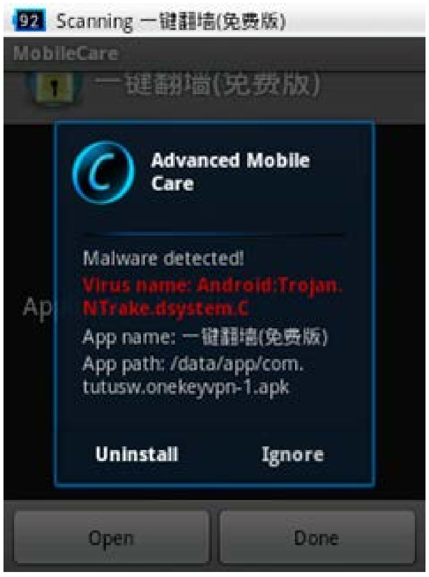
The DroidDream basically tries to get root access to the mobile device similar to DroidKungFu. The first attempt uses "exploid" to attempt to exploit vulnerability in udev event handling in Android's init, or if *exploid* fails to do the job, DroidDream attempts to use *rageagainstthecage*, leveraging vulnerability in adbd's attempt to drop its privileges [11]. Once the malware is complete, it will install a second payload, which will implement other malware commands to the user without their knowledge. So what have been done was, installing an anti-malware, and then downloading the malware DroidDream. It was then tested to see if the malware DroidDream could bypass the anti-malware on the mobile device.

5. Experimental Results

We tested malware programs to see if anti-malware could get rid of malware problems by looking at codes. The reason we looked at the codes was that we needed to understand the problems in the malware. The malware we tested against anti-malware were droid kung fu and DroidDream, which were popular malware on the mobile device. **Figure 3** shows the anti-virus counter that the malware uses for getting into the user's mobile device. Many trails and tests were run to figure out if the current anti-malware was updated and could work on the popular malware. The anti-malware is the best solution to deal with malware from getting in the user mobile device. Based on our test results, legitimate anti-malware seems to be the best solution to deal with malware problems suggested for an average user. Also, the coding for malware can be very complex and difficult for a normal user. The experimental results show that anti-malware can detect most malware on mobile devices. Users should legitimate anti-malware applications from advancing mobile care, Avast, AVG, and Lookout.

6. Conclusion and Future Work

In this paper, we have tested malware in a wide range from USB to mobile devices or direct download to mobile devices. It was difficult to create and maintain the environment without testing the malware, but anti-malware program application was a success in dealing with the malware problems that we looked into. Our research indicated that there could be no problems in handling current malware on mobile devices. But, this may present future problems, since some malware could be avoided easily on the mobile device. Also, most hackers have not yet attacked mobile devices in great numbers. It is difficult to predict how lethal malware could become threats in mobile devices in the future. Our future work will increasingly focus on the types and number of malware to test if anti-malware could deal with multiple malware attack at the same time. The newer and more deadly malware becomes, the more need will arise in testing those malware to see their effects on the mobile devices. Currently, this kind of research is at nascent stage, but with proliferation of mobile devices and attacks; it will become crucial to test malware, and/or develop anti-malware to protect the information and the mobile devices.



Malware detected!
Virus name: Android:Trojan.NTrake.dsystem.C
App name: 一键翻墙(免费版)
App path: /data/app/com.tutusw.onekeyvpn-1.apk

Figure 3. Anti-virus counter the malware.

Acknowledgements

This work has been supported in part by U.S. NSF grant # DUE-1241670 and US Department of Homeland Security Scientific Leadership Award grant # 2012-ST-062-000055.

References

[1] (2009) Android, the World's Most Popular Mobile Platform, 2009. http://developer.android.com/about/index.html

[2] Becher, M., *et al*. (2009) Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices. http://www5.rz.rub.de:8032/imperia/md/content/wolf/ieee_mobile.pdf

[3] Naumann, D. (2014) Mobile App Security Research. http://research.stevens.edu/mobile-app-security

[4] Dimensional Research (2012) The Impact Act of Mobile Devices Information Security: A Survey of It Professionals. http://www.checkpoint.com/downloads/products/check-point-mobile-security-survey-report.pdf

[5] Ian, P. (2012) Mobile Security Threats Rise. PC World. http://www.pcworld.com/article/262017/mobile_security_threats_rise.html

[6] TechTerms (2014) Malware. http://www.techterms.com/definition/malware

[7] TechTerms (2014) Spyware. http://www.techterms.com/definition/spyware

[8] Arsene, L. (2012) An Android Malware Analysis: DroidKungFu. http://www.hotforsecurity.com/blog/an-android-malware-analysis-droidkungfu-4474.html.

[9] Krause, K. (2011) DroidKungFu Malware Evolves to Avoid Detection. http://phandroid.com/2011/07/01/droidkungfu-malware-evolves-to-avoid-detection/

[10] Kindsight (2012) Malware Analysis Report. http://www.kindsight.net/sites/default/files/Kindsight_Malware_Analysis-ZeroAcess-Botnet-final.pdf

[11] Lookout (2011) Lookout Mobile Security Technical Tear Down. https://blog.lookout.com/wp-content/uploads/2011/03/COMPLETE-DroidDream-Technical-Tear-Down_Lookout-Mobile-Security.pdf