Original Research Paper

Designing a Comprehensive Security Framework for Smartphones and Mobile Devices

[1]Nader Jafari, [1]Abeer Alsadoon, [1]Chandana Prasad Withana, [2]Azam Beg and [3]Amr Elchouemi

[1]*School of Computing and Mathematics, Charles Sturt University, Sydney, Australia*
[2]*Collage of Information Technology, UAE University, Al Ain, United Arab Emirates*
[3]*Walden University, USA*

Article history
Received: 04-09-2016
Revised: 26-09-2016
Accepted: 26-09-2016

Corresponding Author:
Chandana Prasad Withana
School of Computing and Mathematics, Charles Sturt University, Sydney, Australia
Email: cwithana@studygroup.com

Abstract: This work investigates issues and challenges of cyber security, specifically malware targeting mobile devices. Recent advances in technology have provided high CPU power, large storage, broad bandwidth and integrated peripheral devices such as Bluetooth, Wi-Fi, 3G/4G to mobile devices, making them popular computing and communication devices. Mobile malware has been targeting mobile devices more than ever and seems to be shifted from their traditional host, the personal computers, to more vulnerable victims. In this study, we mainly focus on malware for Android-based mobile devices. We analyze and discuss related malware and recognize its trends and challenges. We also present a comprehensive security solution that addresses the security from malware threats.

Keywords: Cyber-Security, Mobile Malware, Comprehensive Security Framework, Smartphones, Mobile Device Security

Introduction

The latest breakthroughs in smartphone technology have provided us an “all in one” convenience that the thought of living without them is unimaginable. Unfortunately, the combination of computer technology and presence of old phone systems have attracted hackers and malware developers. A powerful processor, high-speed memory, large storage, high-bandwidth and more importantly the personal and private data make the smartphone a primary focus for most malware. The number of mobile malware has dramatically increased during recent years and will continue to grow, targeting the common vulnerabilities in mobile devices such as Android root exploit.

Among popular mobile devices android platform seems to have inherited the reputation of Microsoft in PC world, of being most vulnerable due to the users’ liberty in installing applications (apps) from outside the Google app store. In addition, rooting capability that is performed by users in order to bypass permission restrictions, adds another layer of vulnerability to Android devices. In contrast, Apple platform is considered more secure due to restrictions and limitations that Apple imposes regarding app installation, making it more difficult for malware distribution through iOS devices. However, the very same restriction drives users to jailbreak their devices to enable them to install desired apps such as Adobe Flash Player, which in return puts the device in danger of malware attacks.

Recognizing the prevalent growth of Android malware, in this study, we investigate the mobile security issues, associated vulnerabilities and potential threats to mobile devices. We also propose a comprehensive security solution to address most of these threats. This paper is arranged as follows. The literature review is explained in section II. Section III outlines the theoretical background followed by the proposed security framework in Section III. The Section IV provides the experimental results followed by the conclusion in Section V.

Related Work

The Android-based devices have dominated the smartphone market with over 78% market share (IDC.com, 2013). Android operating system structurally consists of Linux-based kernel with libraries and APIs and application frameworks running on top of each other. The last layer is application, which runs in Dalvic Java-based virtual machine. This structure helps Android devices to handle multitasking effectively. However android generally does not close applications when the user/architecture is done with them. While this can help prevent excessive interactions by mobile users, it is also considered as a security flaw (Gold, 2011). Thus, it is possible for malware to run silently in the background with high priority for a duration that may last for days, depending on the lifetime of the device power cycle.

Android is also popular among app developers because of its low restrictions and limitations. However, this freedom does not come without a price. Payne (2013) pointed out that lower threshold allows developers to use features that are vulnerable to traditional attacks, such as stack buffer overflows, memory corruptions, heap overflows and race conditions. Based on his suggestions, developers should perform code scanning for any vulnerability and also develop their apps with Address Space Layout Randomization (ASLR) that randomizes where various types of information are kept in the memory.

Moreover, according to (Flegel *et al*., 2013) Android developers often misuse coding idiom in Android platforms due to copying-and-pasting of vulnerable pieces of code. As a result, malicious apps are able to figure out the ordering of system information to perform their attack; they are also able to escalate the privileges or result in Denial of Service (DoS) by crashing an app or the complete OS. The researchers advise developers to avoid using code from untrusted source and when they have to reuse code only from the trusted sources such as Google API's, customization must be performed.

Another popular branch of mobile technology that has been used widely is Near-Field Communication (NFC). The study of (Madlmayr *et al*., 2008) pointed out that NFC allows users to handle their device as e-wallet that is in fact a good prey for hackers. NFC allows transformation of data over 10 cm distance. It uses RFID technology and its enabled mobile device to be used as contactless credit card or bus ticket. Being contactless, an attacker can use an antenna to intercept the NFC signals without being detected. As a prevention method, developers are advised to make sure that sensitive data are not being sent through insecure channels and using HTTPS and TLS instead of simple HTTP.

Apple's iOS is considered to be more secure (than Android) because it prohibits the users from installing third party apps and from places other than Apple app store. However, according to (Hoog, 2008) and (Spaulding *et al*., 2002), this restriction drives users who find themselves in need of installing particular apps such as Adobe FlashPlayer for iOS, to jailbreak their device, which makes their device vulnerable to malicious attacks. Jailbroken devices are susceptible to malware attacks since they do not have access to security patches and there is no restriction enforcement on third party apps. The only advice that researchers have for this situation is to simply not jailbreak the devices. (Mansfield-Devine, 2008) has described the same problem in Android OS devices where inconsistency between new OS and old hardware paves a way for malware infection. Android's new OS cannot be installed on older devices, which leaves older devices susceptible to new threats. Therefore, users tend to root their devices to enable the installation of new OS, usually accompanied with Re-ROMing. ROMing is performed by many users in order to unadorn Android OS, however, these ROM's usually originate from untrusted sources and could be infected with malware.

One of the advantages that makes Android the most popular platform is its capability to install third party applications. It allows the user to install apps from sources other than legitimate ones such as Google Play store. These apps are not certified or scanned by Google Play and could include malicious codes and intended vulnerabilities. Arabo and Pranggono (2008) have suggested a security framework solution to provide security for mobile devices against threats specifically, malware. Their multi-layer integrated security solution consists of four parts: End-User, Network Operators, Market Stores and Apps Developers.

In the End-User section, Arabo and Pranggono (2008) outline the necessity for users to install security controls such as anti-viruses and firewalls. The anti-virus security controls provide the users some protection against known malware based on a malware signature database. Market Stores need to make sure that apps uploaded to the stores are not infected with malware or any kind of malicious codes or activities (Arabo and Pranggono, 2008). In this framework, mobile network providers are responsible for preserving mobile network security by scanning incoming and outgoing SMS/MMS for any malicious propagation by Mobile Network Operators. Developers as a last part of the framework are advised to reduce the built-in access permission in their application and also control access to functions such as (CALL_PHONE) and (SEND_SMS) in Android devices and always ask for user's concession to prevent malicious use such as event listener. They should also perform control permissions technique to fight against "repackaging attack" by including only required and essential data that is required for applications to operate properly. Repackaging is a common technique that Android malware developers use to download legitimate apps from internet, inject some malicious code in them and upload them back onto download websites (Arabo and Pranggono, 2008).

Although app stores nowadays perform intensive malware scanning before developers are able to upload their app for sale on store, malware creators have developed techniques to circumvent these security controls, specifically for Google Store. Two widely used techniques are logic bombs and checks for simulation environment (Ho *et al*., 2014). To fight against these, some researchers suggest behavioural approaches such as Crowdroid proposed by (Burguera *et al*., 2011); Practical Root Exploit Containment (PREC) (Ho *et al*.,

2014) and A Defence Framework Against Malware and Vulnerability Exploits was proposed by (Zhang *et al.*, 2014). Amongst those, PREC appears to be more robust and one of its greatest advantages is that it imposes less than 3% overhead on the mobile device in comparison to other methods with 15-30% overhead.

PREC scans and monitors apps that are being downloaded from app stores. Its job is to detect and report any abnormal and suspicious activities by apps such as attempting to root user devices without their consent. Other suspicious activities can be sending SMS/MMS, email, or making calls to premium accounts. PREC monitors apps from the moment that they are uploaded to the App Stores. This behavioural approach acts intermediately between the end-user and store app components as an integrated security solution.

PREC targets and dynamically identifies system calls from high-risk components specifically third native libraries that are being used for root exploit attack. The benign apps use less than 10% of third native libraries, hence the false positives and the false negatives will be very low. The procedure can be divided into two sections:

- First, it utilizes a "classified system-call monitoring" layout that can recognize system calls according to their origins. This allows system calls from risky components such as third party native libraries to be identified (for instance native libraries that are not part of the Android system but were added by downloading from Internet using applications). It performs its anomaly check only on system calls that were created from third party native code
- Second, it uses a "delay-based fine-grained containment" structure that performs the anomalous system calls from a pool of available segregate threats in order to slow them down and prevent the threat

However, PREC has some disadvantages as well, for instance, it cannot protect user devices if they download an app from third party stores simply because PREC has no observation on them. Moreover, it only targets and observes system calls generated from the third party native code. Thereby, it is theoretically possible that root exploits attack can be generated from the Java code.

In general, a user is a weak link of the any cyber security chain. The common users do not have technical skills required to protect their devices and lack awareness from device manufactures making the matter more complicated. Therefore, there is a need for a comprehensive security framework that provides a first layer of defence for user mobile devices and a second layer that scans apps across app stores both when uploaded by developers and downloaded by users. The second layer security continues to monitor app's behavior through the user device to ensure that the app is not performing any malicious activities.

Theoretical Background

Smart Device Security Threats

In computer security concepts, the user is always considered as a weak link of the chain. The assumption is that everything that comes out of a box is secure enough by default. According to Arabo and Pranggono (2008), the primary challenge with mobile security devices is their ubiquity and lack of awareness of threats associated with these devices. Devices insecure default settings with relaxed security features put both the device and the user in danger of malware infection that ultimately results in the invasion of user's privacy and loss of personal sensitive data or corporate information.

With the decrease in size of mobile devices and increase in capacity in terms of screen size, processing power and storage size they are now suitable to be used for working remotely. Moreover, the mobile devices are now integrated into business more than ever and CIOs are struggling to put in place effective policies to counter-measure possible threats (Courtney, 2014). The widespread concept of Bring Your Own Device (BYOD) is making businesses extremely nervous as they do not have control over corporate information that employees process within those devices. That puts corporate data in danger of information leak since most of these devices do not enjoy proper security control either because of users' lack of awareness or the very nature of the mobile device's platform, specifically Android devices. Moreover, according to (Kaspersky, 2013), the number of stolen mobile devices has increased dramatically and employees are slow in reporting the stolen devices.

Android platform is vulnerable to a variety of attacks, mostly due to its OS layers structure and the fact that it owns the majority of market share. Such attacks can cause several privacy and security risks for users such as phone calls tracking, extraction of SMS/MMS, loss of privacy and exposure of information and overbilling due to call to premium accounts. The vectors of such attacks can be Bluetooth, USB connection, Wi-Fi, 3G and 4G and overall Internet connection (Hunt, 2013). Apple iOS proved to be more secure by restricting users installing apps only from legitimate app stores, iTune and similar. However, that does not prevent the devices to be vulnerable to attacks as many users are unhappy with restriction and jailbreak their device to have freedom of choice. That puts device in danger of a variety of threats including malware.

Wi-Fi is the most used method of connection for a variety of Internet access such as E-mail, online chat, free calling and text messaging through providers such as WhatsApp, Skype and Viber. WLANs that utilize Wi-Fi standard technology have evolved over the years and now benefit from the new security connection protocols such as WPA2 in conjunction with Advanced Encryption Standard (AES) instead of the weaker WPE, which only used 56 bit keys for encryption. However, improvement of a number of powerful brute-force and dictionary attacks on the Message Digest 5 (MD5) and similar encryption methods has kept the security in a shadow.

The so-called Internet of Things (IoT) that comprises a variety of structures and technologies such as NFC, Ultra Wide Band (UWB, etc. is rapidly integrated and implemented in mobile devices for the users' convenience. For instance, Apple has recently implemented NFC in their new iPhone 6, which helps users pay from their smartphone on the go. While this could be a decent replacement for smart cards, the encryption method used in NFC is much weaker as compared to the smart cards; an attacker with a special device in proximity can intercept the communication signal.

Smart Device Malware

In recent years, we have witnessed great advances in mobile device technology that has resulted in tremendous growth in their sales and adaption. The mobility, convenience and the affordable prices are characteristics that accelerated their adaptation and outsold personal computers. However, as anything else in this world, there is a drawback associated with this popularity-a growing outbreak of mobile malware. Among all other devices android is considered as less secure and more prone to malware attack. Based on Kaspersky's recent report, 10 million Android malicious apps were detected between 2010 and 2013 and 4 million of them belong to 2013, something that Kaspersky calls "3 infection attempts per user."

As previously mentioned, mobile devices, specifically smartphones, pose multiple communication interfaces such as G3/G4, Wi-Fi, Bluetooth, USB and others to surf Internet, send/receive emails and visit social media websites (Ghallali *et al*., 2013), The malware uses these peripheral devices to infect and propagate from one device to another. For instance, one popular channel that malware uses to spread is through SMS/MMS that is considered as an email for pc viruses.

Other effective ways of malware propagation are repackaging, tricking users to install malicious app and update attack (Jiang and Zhou, 2013). Repackaging is one of the most popular technique used by malware developers to download the app from app store, unpack it, inject malicious code in it and upload it back on download websites that appear as benign app to the end user.

Luring the users to install malicious apps is another technique that malware developers have used widely. This technique usually uses social media, SMS and online chats to present malicious apps as something interesting to users. Once the app has been installed by the user, the malicious software propagates through peripheral devices, such as Wi-Fi, Bluetooth, 3G/4G and whatever is available for it to spread. An appropriate example to this is a recent Android and iOS malware called Xsser that spreads through WhatsApp application asking users to install the app with the message read: "Check out this Android app designed by Code4HK for the coordination of OCCUPY CENTRAL!" The malware targeted Hong Kong protestors and movement known as Occupy Central (Borbrov, 2014). The attacker(s) impersonated as a hacker group that's helping the Occupy Central movement and tricked protestors to install malicious app that steals user credentials, contact numbers, phone books, SMSs/MMSs and many more.

Comprehensive Security Framework Design

Potential Security Framework Model

Considering all cyber-security threats and especially the malware discussed earlier (Section 2), the need for a comprehensive solution is evident. As pointed out by Arabo and Pranggono (2008), the main cyber-security concern with Android devices is user liberty to install applications (apps) from any sources. As explained previously, this feature provides the user a high convenience while it poses great security issues for users and their devices. As mobile malware become more intelligent, the need for more robust security solution is highly required. As with increase in malware distribution from Google play store, Google added an extra security layer that scans apps for any malicious codes and activities before being introduced to the end user (Arabo and Pranggono, 2008). However, that still doesn't stop malware from spreading as malware creators developed techniques to circumvent these security controls and specifically Google store. Two widely used techniques are logic bombs and check for simulation environment (Ho *et al*., 2014). In addition, users' desires to install third party apps that are not supervised by Google play app's store security measures, poses another security issue that needs to be considered.

This paper proposes a security framework solution that aims to cover most aspects of mobile device security that may be neglected by other security solutions, as illustrated in Fig. 1.

The proposed framework provides security for users and can be divided into three sections.

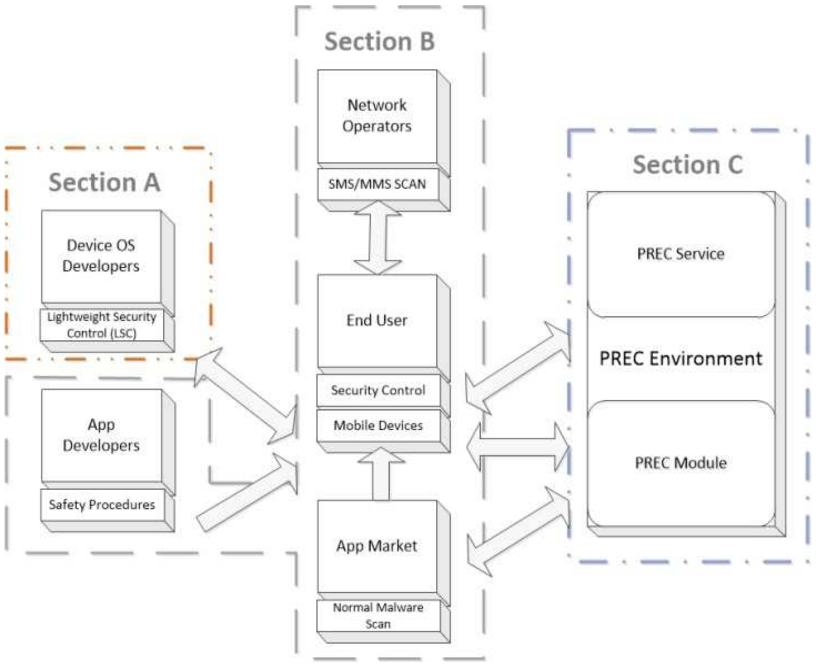


Fig. 1. Comprehensive security solution for mobile and smart devices

Section A

The first section of the model represents the security feature that is embedded in the device's OS and has one main subsection, Device OS Developers.

Device OS Developers

Device OS developers, such as Android OS Developers and iOS developers should integrate some form of lightweight security such as Lightweight Security Control (LSC) as proposed in this research. The idea of LSC is similar to how Microsoft implements primitive security features in its latest OSs such as Windows 7 and Windows 8. Microsoft defines its anti-Malware, Windows Defender, as a first line of defence against malware. The Defender was originally an anti-spyware that then turned into antivirus (Microsoft). It is light and does not have features that most antiviruses possess, but as Microsoft defined it, it is the first line of defence specifically against viruses that Microsoft as a platform developer is aware of.

The LSC should include following features:

- Lightweight firewall performs very basic packet filtering, port opening and closing and similar tasks
- Lightweight signature-based anti-virus protects the device against OSs' vulnerabilities by scanning the device for malware that exploits those weaknesses such as root exploit malware in Android devices. This anti-virus does not have the comprehensive malware's signature database that most mobile anti-viruses possess, thus it is very light and only fights against malware that exploits devices' vulnerabilities
- Social engineering defender alerts users against any social engineering attacks, for instance, when the user wants to click or tap on links within spam emails or emails from unknown sources that are not in users contact list. Moreover, the defender warns the users when they receive SMS/MMS from unknown senders

LSC should be configured in a way that when users first start up their devices, it walks them through a set of short security settings. Alternatively, users can choose to set their overall security settings based on predefined settings. These predefined settings are:

- Low: Where the device is configured with very loosen security measures
- Medium: The device is configured with the medium range security measures
- High: The device is highly restricted and limited and users are warned for any security violation actions from users that LSC may recognize

The Medium predefined security is what is recommended by LSC. Users also have the choice to turn the LSC off or on for any reasons. LSC consists of two major parts, i.e., malware protection and Security Watchdog and consists of two subsections as listed in Fig. 2.

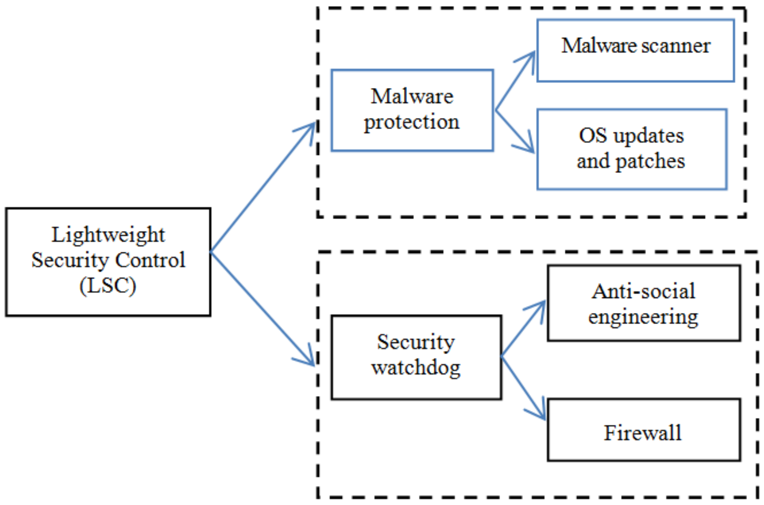


Fig. 2. LSC breakdown structure

The following explains the two sections in more detail:

Malware Protection

Malware Scanner

Scans system regularly for malware that targets OS vulnerabilities

B. OS update and patches

Updates OS and installs patches mostly in the background

Security Watchdog

A. Anti-Social Engineering

Alerts users regarding social engineering attacks by containing links inside SMS/MMS or email messages that have become one of the main channels for malware propagation on mobile devices.

B. Firewall

Provides security settings that walk the users through or suggests pre-configured security settings for user convenience.

To improve functionality, the LSC's malware protection feature (section 1) implementation is compulsory and there is no option available for the user to disable the feature. However to give user some liberty, the security watchdog feature (section II) is not compulsory and the user can disable the feature if required.

Section B:

This section of model mainly concerns those securities protecting users' privacy and credentials while they are performing regular Internet activities, such as Internet-surfing, performing banking transactions and reading emails on their devices.

This section consists of four security components.

End-User

Users are expected to install security controls, such as anti-viruses and firewalls.

As mentioned in section 3, anti-malware code is limited to a signature technique due to resource limitation of mobile devices, such as battery and bandwidth. Yet some believe that even signature-based technique is resource-intensive and hence unsuitable solution to apply in smartphones (Burguera *et al*., 2011; Zhang *et al*., 2014). However, the new generation anti-viruses are lighter, less resource hungry and consume less bandwidth. In addition, some anti-viruses store their databases on cloud, which is a clever method to consume even less bandwidth.

In addition, to test and to examine the anti-viruses' effectiveness and robustness, a set of malware with different functionality and payloads was selected. The list of malware that was used for testing is presented in Table 1.

App Market or Stores

App stores need to constantly scan their app database for any app's malicious activities. The two most popular app stores, i.e., iTunes and Google Play, already contain scanning apps for any abnormal behaviour prior to be presented to the end-user for purchase and download. However, they should also take a further step to scan apps to make sure user's privacy is not violated by the app developers.

Mobile Network Providers

Mobile network providers should keep track of SMS's/MMS's that are used for communication between botnets and botnet master (Arabo and Pranggono, 2008).

Table 1. Malware list

Malware Name	Payload functionality	Level of risk
Httpmon	Contains Trojan horse	Medium-High
Angry BIRD Rio unlocker	Trojan Plankton.,	Medium-High
Android.AVPass (Root exploit Malware)	Contains Trojan horse	High
CABAHHA (Android.Badnews)	Trojan Plankton	Medium-High
Cut the rope unlock	Trojan Plankton	Medium-High
Superuser	Root exploit malware	High

App Developers

In addition of what has already been mentioned in section 2, app developers also encourage writing their apps with Address Space Layout Randomization (ASLR) that randomizes where various types of information are kept in memory (Payne, 2013). This randomization protects apps from buffer overflows, memory corruptions, heap overflows and race conditions attacks. Moreover android app developers sometimes reuse code from trusted sources such as Google API's to save time, thus, when they have to do so customization must be performed to protect against known threats such as buffer overflow. Google suggests changing certain parts of codes specifically when using a significant portion of the Google In-App Billing and License Verification instance code (Flegel *et al*., 2013).

Section C

The most critical level of this model is carried out by PREC (Ho *et al*., 2014). When the app is first submitted by the developer to the app market, it is being scanned by the malware detection system running in a quarantine emulator environment. If it is recognized as malicious it would be rejected, otherwise, the app's "normal profile behavior" is saved and forwarded to PREC services that could be residing on cloud. Once the app is downloaded by the user and starts functioning, PREC retrieves the app's profile and keeps an eye on that app for any root exploit activity and contains it, if necessary.

Experimental Results and Discussions

This section provides discussion and the experimental results of previously mentioned methods, graphs and tables. Figure 1 illustrates the results based on antivirus capability in detecting malware in percentage. This study shows that the average percentage of malware detection of all five anti-viruses is 83.2%. The malware test environment was Android SDK emulator with API 19.0 using Goldfish 3.4 OS, which is a stable and common virtual platform used for Android app development and testing. The steps required for preparing this infrastructure are:

Step 1: Building the host environment (Linux OS); this step is completed by installing and configuring Kali Linux OS on VMware Player virtual machine along with all required update and packages.

Step 2: Building Android development environment in Linux to host Mobile virtual device. It has been done by installing and configuring Android ADT bundle development package android Goldfish source code and all other necessarily packages such as Android NDK and repo in Linux virtual machine.

Step 3: Testing the environment (virtual device) to make sure it works properly. Running an emulator (Android virtual environment or goldfish) and testing all its functionalities to make sure it has been built properly.

Moreover, the effectiveness of five top anti-viruses in the market at the time of this research, were tested by injecting six types of Android malwares that were installed on Android virtual device (goldfish 3.4). Figure 3 the malwares differ in their payloads, potential risks, persistency, malicious behaviors and functionalities, for instance root exploit malware and Trojan Malware. They were acquired form contagiominidump.blogspot.com.au (2014) and www.worldguide.pt (2014).

Figure 3 shows the effectiveness of five best mobile antiviruses in market.

The results of the following graphs are generated using PowerTutor tool that utilizes PowerBooter. PowerBooter is an automated power model construction technique, which uses voltage sensors and battery discharge behaviour utility to monitor power consumption, which for 10-sec intervals, it is accurate to within 0.8% on average with at most 2.5% error (Zhang *et al*., 2010). The explicitly monitors the power consumption based on usage of LCD, CPU and Wi-Fi components, which requires no external measurement equipment. The purpose of this step is to evaluate mobile devices' battery consumption used by antiviruses during scanning and normal operation. This step differs from previous, because testing were conducted on actual android device (Samsung Galaxy SII i9100 with Android OS 4.1.2 with 16 GB memory storage) and reason is simply because Power Tutor needs battery sensors to produce accurate outputs. Note that SD-card scan is not included in these tests.

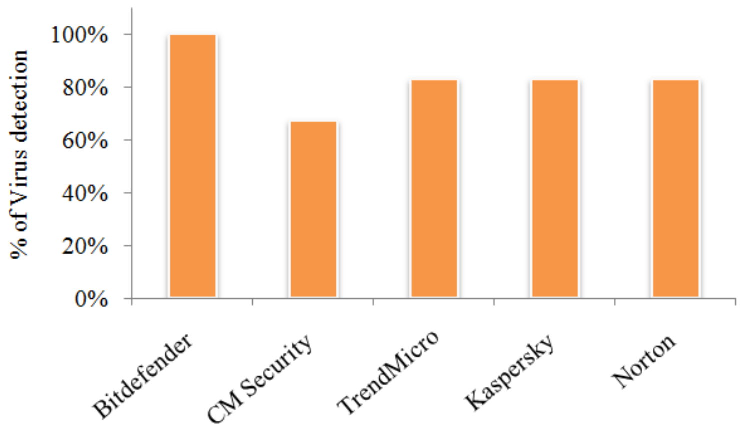


Fig. 3. Anti-viruses malware detection efficiencies

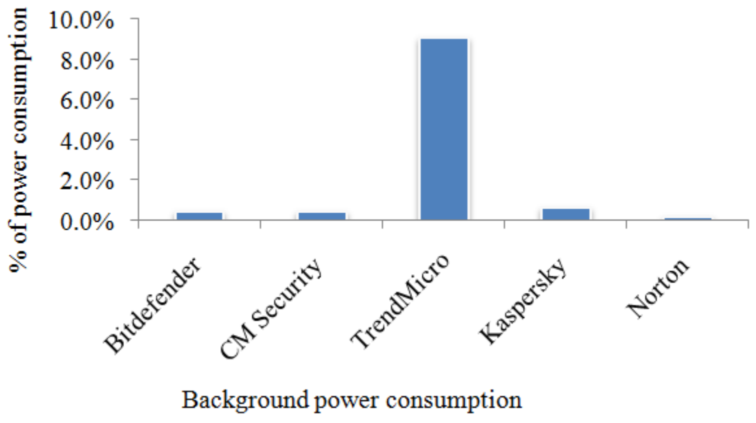


Fig. 4. Percentage of power consumption in background

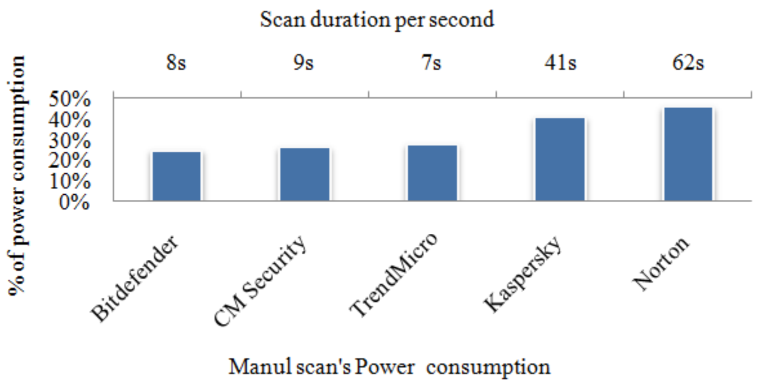


Fig. 5. Percentage of power consumption during manual scanning

This procedure has two steps:

- Step one: Injecting or installing malicious apps in goldfish and then installing anti-virus from different vendors (the vendors' names are mentioned in antivirus table) to determine the level of security that antivirus provides for user. This includes the antivirus ability to detect malware automatically without user intervention, scanning for virus on user demand, covering vulnerabilities and other security measures such as locking SIM, scanning SMS/MMS and monitoring Internet data flow
- Step two: Determine the malware residence in captured memory after being injected in and also for malware persistency after being removed by Anti-virus, using Lime tool to capture the memory and Volatility tool to examine the dumped memory. For more information, please Fig. 7

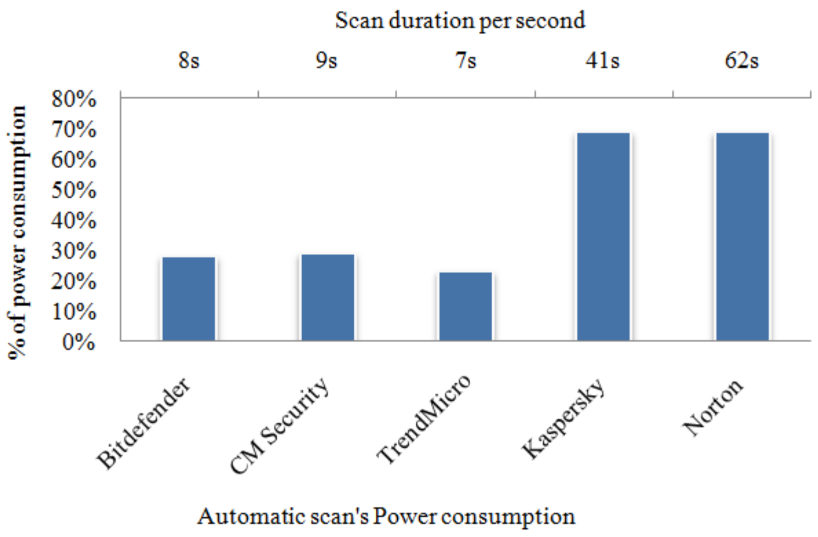
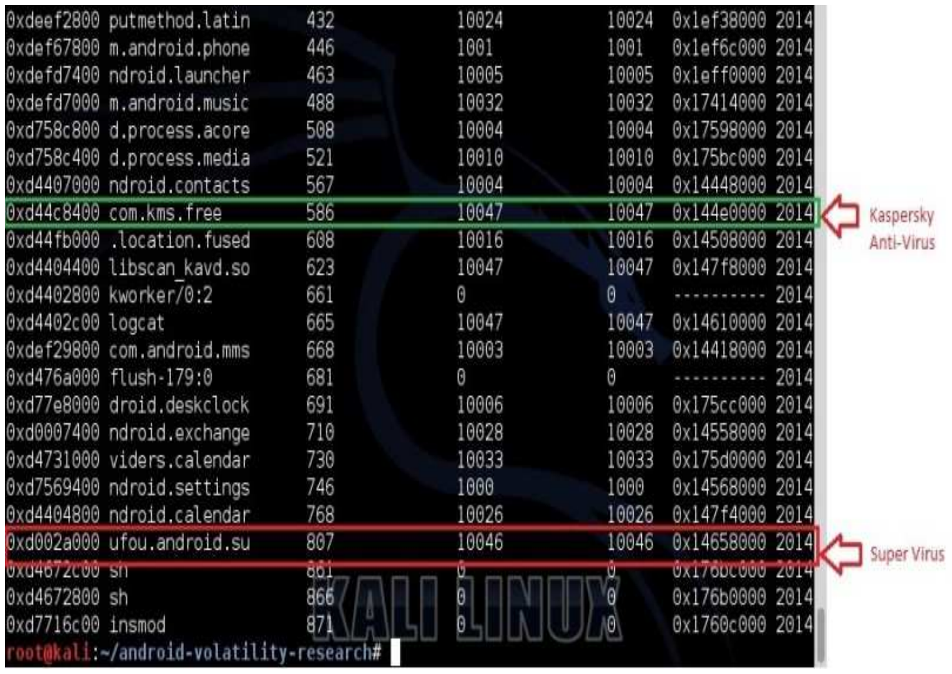


Fig. 6. Percentage of power consumption during automatic scanning



```
0xdeef2800 putmethod.latin    432      10024      10024  0x1ef38000 2014
0xdef67800 m.android.phone    446      1001       1001   0x1ef6c000 2014
0xdefd7400 ndroid.launcher    463      10005      10005  0x1eff0000 2014
0xdefd7000 m.android.music    488      10032      10032  0x17414000 2014
0xd758c800 d.process.acore    508      10004      10004  0x17598000 2014
0xd758c400 d.process.media    521      10010      10010  0x175bc000 2014
0xd4407000 ndroid.contacts    567      10004      10004  0x14448000 2014
0xd44c8400 com.kms.free       586      10047      10047  0x144e0000 2014
0xd44fb000 .location.fused    608      10016      10016  0x14508000 2014
0xd4404400 libscan_kavd.so    623      10047      10047  0x147f8000 2014
0xd4402800 kworker/0:2        661      0          0      ---------- 2014
0xd4402c00 logcat             665      10047      10047  0x14610000 2014
0xdef29800 com.android.mms    668      10003      10003  0x14418000 2014
0xd476a000 flush-179:0        681      0          0      ---------- 2014
0xd77e8000 droid.deskclock    691      10006      10006  0x175cc000 2014
0xd0007400 ndroid.exchange    710      10028      10028  0x14558000 2014
0xd4731000 viders.calendar    730      10033      10033  0x175d0000 2014
0xd7569400 ndroid.settings    746      1000       1000   0x14568000 2014
0xd4404800 ndroid.calendar    768      10026      10026  0x147f4000 2014
0xd002a000 ufou.android.su    807      10046      10046  0x14658000 2014
0xd4672c00 sh                 861      0          0      0x176bc000 2014
0xd4672800 sh                 866      0          0      0x176b0000 2014
0xd7716c00 insmod             871      0          0      0x1760c000 2014
root@kali:~/android-volatility-research#
```

Fig. 7. The figure illustrates the presence of httpmon malware with PID (process ID) 998, UID (User ID) 10047 and GID (Group ID) 10047 in the device memory that was captured using Lime tool and examined with Volatilities tools. The figure also shows that antivirus Kaspersky (PID 998 and PID 1047) was installed at time memory was captured, which indicates the antivirus failure to identify and remove the malware

Figure 4 shows the percentage of power consumption of anti-Malware during ideal activities. The tests were conducted using actual device Samsung Galaxy SII i1900 with Android OS 4.2.1 as a test bed.

Figure 5 illustrates the power consumption during manual scanning; this involves power consumption of two important components, the CPU and the LCD.

Finally, Figure 6 presents power consumption during automatic scan which follows a regular schedule of anti-malware. Note that during this scan the major power consumer is CPU, since the LCD is usually turned off.

As illustrated above, Figures 4-6 illustrate the power consumption performance by chosen antiviruses under three different circumstances. The purpose of this step is to study the power consumption efficiency of mobile antiviruses. The five selected anti-malware were the most popular at time when this research conducted. The conditions that antiviruses were examined based on are:

- Idle time: Measuring the amount of the battery's juice each antivirus uses during normal and background operation in 10 min time interval using PowerTutor tool
- Manual scan: Measuring the battery consumption during heavy battery (scanning for virus) drain by antivirus. The time interval used here is calculated based on actual time that each antivirus takes to scan plus 4 sec for error tolerance. The results are calculated based on the average of 10 test repeats to reduce errors (Table 1 the power consumption Table). The two hardware battery's consumption that accounted here are LCD and CPU. Note that the test assumes that user perform manual scan, which involves consumption of both LCD and CPU battery's power drain
- Automatic scan: This step is calculated based on situation that antivirus runs scan automatically. It is assumed that since user has no intervention, LCD power consumption is 0 and only CPU's that consumes power

Conclusion

The cyber-security threats, specifically malware are spread from their traditional hosts desktop computers to smartphones and mobile devices as these devices are more vulnerable and they contain more personal information. Although, the purpose of mobile malware is perhaps different from their traditional cousin (computer's malware) the concept and functionalities of mobile malware has not changed significantly. This paper has discussed current issues with existing mobile security controls and also proposed a comprehensive model of security solution framework that would address mobile security issues and more specifically Malware.

This work is a part of ongoing research to design and implement a comprehensive security framework model for mobile devices. For the future work, we plan to develop, implement and evaluate the LSC (described in section A) of the proposed model for the Android devices.

Funding Information

The authors have no support or funding to report.

Author's Contributions

Nader Jafari: Investigate issues and challenges of cyber security and presented a comprehensive security solution that addresses the security from malware threats.

Abeer Alsadoon: Supervised/worked closely with Nader Jafari during the analysis, design and experiment phases.

Chandana Prasad Withana: Worked on the setup of the experiments, and gave important suggestions on design of experiments.

Azam Beg: Made important revisions to most sections of the paper.

Amr Elchouemi: Give the final review and approval for the manuscript to be submitted.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Arabo, A. and B. Pranggono, 2008. Mobile malware and smart device security: Trends, challenges and solutions. Proceedings of the 19th International Conference on Control Systems and Computer Science, May 29-31, IEEE Xplore Press, pp: 526-531. DOI: 10.1109/CSCS.2013.27

Borbrov, O., 2014. Lacoon discovers Xsser mRAT, the first advanced Chinese iOS Trojan. Check Point Software Technologies Ltd.

Burguera, I., U. Zurutuza and S. Nadjm-Tehrani, 2011. Crowdroid: Behavior-Based Malware Detection System for Android. Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, Oct. 17-21, Chicago, IL, USA, pp: 15-26. DOI: 10.1145/2046614.2046619

Courtney, M., 2014. Protecting the mobile enterprise. Eng. Technol., 9: 72-76. DOI: 10.1049/et.2014.0409

Flegel, U., E. Markatos and W. Robertson, 2013. Detection of intrusions and malware and vulnerability assessment.

Ghallali, M., A.E. Mir, B.E. Quahidi, B. Bounabat and N.E. Hami *et al*., 2013. Mobile security: Designing a new framework limiting malware spread in the mobile cloud computing. J. Theoretical Applied Inform. Technol., 57: 354-366.

Gold, S., 2011. Android inscesuity. Network Security, 10: 5-16.

Ho, T., D. Dean, X. Gu and W. Enck, 2014. PREC: Practical root exploit containment for android devices. Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, Mar. 03-05, San Antonio, TX, USA, pp: 187-198. DOI: 10.1145/2557547.2557563

Hoog, A., 2008. The battle beyond Malware: Combating the wrong enemy? Evolving threats and new attack surfaces demand mobile security strategies keep pace. Inform. Security.

Hunt, R., 2013. Security testing in android networks: A practical case study. Proceedings of the 19th IEEE International Conference on Networks, Dec. 11-13, IEEE Xplore Press, pp: 1-6. DOI: 10.1109/ICON.2013.6781950

IDC.com, 2014. Android and iOS Continue to Dominate the Worldwide Smartphone Market with Android Shipments Just Shy of 800 Million in 2013. Business Wire.

Jiang, X. and Y. Zhou, 2013. Android Malware. 1st Edn., Springer Science and Business Media, New York, ISBN-10: 1461473942, pp: 44.

Kaspersky, 2013. Mobile malware evolution: 3 infection attempts per user in 2013. Kaspersky Lab.

Madlmayr, G., J. Langer and C. Kantner, 2008. NFC Devices: Security and privacy. Proceedings of the 3rd International Conference on Availability, Reliability and Security, Mar. 4-7, IEEE Xplore Press, 642-647. DOI: 10.1109/ARES.2008.105

Mansfield-Devine, S., 2008. Android architecture: Attacking the weak points. Network Security, 10: 5-12. DOI: 10.1016/S1353-4858(12)70092-2

Payne, J., 2013. Secure Mobile Application Development. IT Professional, 3: 6-9. DOI: 10.1109/MITP.2013.46

Spaulding, J., A. Krauss and A. Srinivasan, 2002. Exploring an open WiFi detection vulnerability as a malware attack vector on iOS devices. Proceedings of the 7th International Conference on Malicious and Unwanted Software, Oct. 16-18, IEEE Xplore Press, pp: 87-93. DOI: 10.1109/MALWARE.2012.6461013

Zhang, L., B. Tiwana, R.P. Dick, Z. Qian, Z.M. Mao and Z. Wang, 2010. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Oct. 28-28, Scottsdale, AZ, USA, pp: 105-114. DOI: 10.1145/1878961.1878982

Zhang, M., A. Raghunathan and N.K.A. Jha, 2014. A defense framework against malware and vulnerability exploits. Int. J. Inform. Security, 13: 439-452. DOI: 10.1007/s10207-014-0233-1