A Mobile Operating System Commerical Malware Detection Survey

Imani Palmer
Department of Computer Science
University of Illinois at Urbana-Champaign
ipalmer2@illinois.edu

Jordan Martin
Department of Computer Science
University of Illinois at Urbana-Champaign
jmmarti3@illinois.edu

Abstract

Android is a software framework for mobile devices. The Android operating system is based on the Linux kernel designed as a touchscreen for mobile devices. Android is open source and licensing allows the software to be modified freely. The Android community has a large number of developers writing applications through the primary language, Java. There are approximately 25 billion Android applications available in the Google Play store. And as of May 2013 Android has been deemed the most popular platform, holding 71% of the mobile developer population.

The popularity and openess of this mobile operating system increases the risk of exposure of malware.[13] However, most commercialized approaches fail to provide beyond basic signature detection. Signature detection is the scanning of files provided by the application, and checking them for known virus signatures. Signature-based detection only protects against known viruses and fails to prevent new viruses from being installed. This leads to the question of the true benefits of security applications and the potential solutions to the constant threat of malware. [10]

The goal of this paper is to compare commerical tools against known threats and find weak points that can be improved. We gathered empirical data on several commercial anti-malware applications from GooglePlay. This gave us a knowledge base to use when making the decision of which anti-malware products to install. Malware was loaded onto the device using an Android injector on an unprotected device as well as with six different anti-malware applications. Based on the results, we determine that several common commercial anti-virus applications fall short of full protection, even against malware documented from previous years.

1 Introduction

The Android mobile device is built off of the Linux kernel. On top of the Linux kernel are the middleware, libraries and APIs, and the application software running on the application framework. This framework also includes Java-compatible librares. Android uses the Dalvik virtual machine with a just-in-time compilation to run Dalvik dex code.

The Android operating system does have its own built in security and privacy. All Android applications run in a sandbox. A sandbox is an isolated area of the system that typically do not have access to the rest of the system's resources. The only way to get access to these resources are through permissions. Before an Android application is installed, the Google Play Store displays all requested permissions. After reviewing these permissions, the user can choose to accept or refuse the requests. The application will only be installed if they accept the requests. An example of this is a basic game application. A game application will need permission to save data to the SD card and possibe to have access to the accelerometer. This concept appears to be simple, however, many developers seem to have developed confusion through poor documentation leading to applications constantly requesting permissions that are unnecessary. Access to privilege permission such as a user's contact list or wifi connectivity is the main definition of malware on Android devices.

With mobile device usage growing at an alarming rate, the increase in devices and the increase in profitability in malware targeted to mobile platforms, Android has become a victim of many attacks. There have been four major malware attacks. First, fake banking applications lure customers into entering their bank account information login details. Second, Android.Geinimi has corrupted many legitimate Android games from Chinese download sites. Third, DroidDream infects devices, breached the android security sandbox and stole data. Fourth, the AndroidOS fake

player appears to be a media player and silently sends SMS to premium SMS numbers.

The protection of private data and allowing users to interact with the online community without fear of their device being compromised is a must. The Public Intelligence published a report that says 79 % of all mobile threats in 2012 targeted Android. Many anti-malware programs have been created, each touting itself as the best on the market. However, there is a lack of proper evidence to prove which commerical anti-malware programs are best.

After performing the study, we were able to determine certain things about Android mobile security. First, there is need for access to current android malware in order to properly test security tools. Second, there is need of open source collaboration between academic and industry in order to share and create security testing tools for android. Third, industry may need to allow open source versions of their security applications in order for better research to be conducted in this area. Lastly, the best way to enhance mobile security would be from the operating system and not from external applications. Malware security is a threat to the privacy of its users and enahcned mobile operating system security is determined as the best method.

In this paper we will further review many security topics and possible solutions on a mobile device. In section two, we will talk about related work. In section three, a review of the android threat model and the possible threats specific to Android devices. In the fourth section, we will go over how we conducted our basic experiment. In the fifth section, we will review the results of the experiment. In the sixth section, a discussion of the security techniques already invested in the Android mobiile operating system. Finally, section seven will go over the conclusion we have drawn and the theoritcal solutions we have developed.

2 Related Work

There are many works related to mobile malware detection. In 2005, Yap and Ewe created a prototype of the simulation that malicious software and detection software on mobile devices. They also proposed a basic solution that detection software should have three main components: scanning, behavior checkers, and integrity checkers. The comparison of these requirements to todays commercial mobile malware detection demonstrate that most lack at least two of these three requirements. However, all include the basic scanning.[14] In 2007, Cheng, Wong, Yang and Lu developed the SmartSiren a collaborative virus detection and alert system for smartphones. Even with it feasibility and effectiveness in trace-driven simulation it appears that industry has failed to include academia in it development of such tools.[3] At North Carolina State University Yajin Zhou and Xuxian Jiang have started the Android Mawlare Genome Project. In this project they focus on the Android platform and aim to systematize existing Android malware.[11] There effort hopes to aid the call for better development of next generation anti-mobile malware solutions.

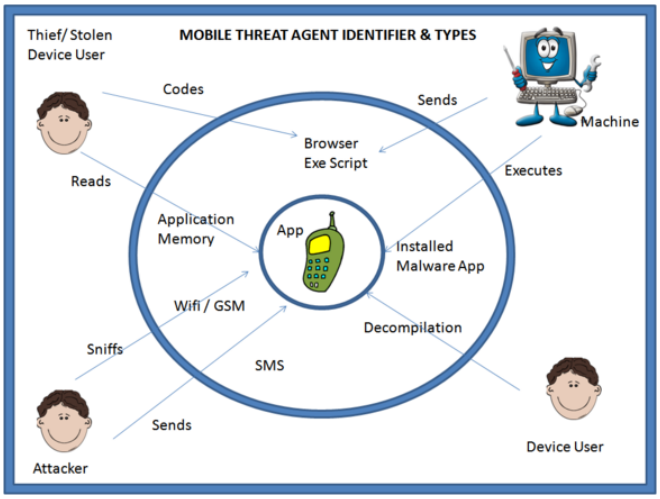

Figure 1: Mobile Threat Agent Identifier And Types [5]

3 Android Threat Model

The mobile operating systems are facing new challenges daily. These challenges are noticeably similar to problems encountered earlier in the development of the personal computer.[14] There are many threat agents for mobile operating system. These threats are repsented in Figure 1. The first category of threat agents deal with human interaction. These threats include the stolen device user, the owner of the device, the common WiFi network user, the malicious developer, the organization internal employees and the app store approvers/reviewers. The stolen device user is a user who obtained unauthorized access to the device trying to get enhanced access to sensitive information. The common wifi network user is an agent

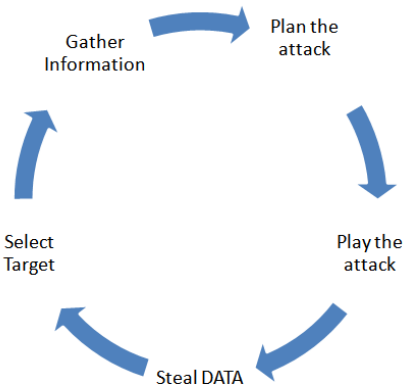

Figure 2: How to Attack a Mobile Operating System [5]

who either intentionally or unintentionally sniffs the wifi network used by a victim. The information from the sniffing may be used to launch further attacks. The malicious developer is a human user who writes applications with harmful intent. This agent is looking to steal information the device in order to possibly use it to perform a denial-of-service attack. There are subcategories under this particulate agent. First, the organization internal employees who have privileges to perform an action on the application. Second, the app store approvers and reviewers that fail to remove dangerous applications from their store. The most popular human interaction based threat agent is the owner of the device is a user who unknowingly installed a malicious application on their own phone which is able to get access to the devices application memory.[5] Threat agents are automated programs. The automated programs we are concerned about are the malware on the mobile device. This is any mobile application that performs suspicious activity.The main areas of concern on android mobile devices includes malicious SMS and malicious applications. The malicious SMS is an incoming SMS that triggers suspicious activity on the mobile device. The malicious application is the failure to detect potentially harmful code.[5] Threat agents aid in the development in mobile applications. Threat agents allow developers to focus on specific attacks that would be more significant on mobile operating systems and than on personal computers. The understanding of threat agents aids in the development of mobile security.

Figure 2 demonstrates how a threat agent attack a mobile operating system. First, the attacker plans the attack. A specific threat of most Android attacks are the development of applications that request permissions to sensitive data, such as the contact list. The attacker than plays the attack by add the this malicious application to the Google Play Store. A user downloads the application and their personal data is stolen. Lastly, the attacker gathers information and uses the sensitive information maliciously.

4 Experiment

The experiment was conducted on an ASUS Nexus 7 with a quad-core Qualcomm Snapdragon S4 Pro processor and 2GB of RAM. The Android application that were tested in this study are some of the top mobile security applications from Googles Play Store. These applications were: AVG, Avast, Lookout, Norton, MacAfee, and Malware Bytes. For security reasons the malware used to conduct this experiment will not be named. Each malware detection application was downloaded and installed from the Google Play Store and than ran while there was no malware injected onto the device. Once the original scan was performed we injected a known malware on to the device and allow the malware detection application to scan the known malware. Our hypothesis was that sense all the malware had already been named and discovered that their would be a 100% detection rate amongst all of the applications. Along with the detection rate results each application contained some notable features. Avast had no mobile operating system integration. This means it worked solely on the Android framework and did not alert the operating system that it had located a virus. Lookout had specific text pop ups that included more information about the malware trying to be installed. MacAfee was the slowest and seemed to have many graphical user interface issues. Overall it appeared to have a poorest implementation. Malware Bytes offered the user a whitelist option, so that even if it detect a malicious malware the user could still allow it on the device. Lastly, AVG focused on the prevention of even installing the application.Even though there were many differences, the Android malware detection applications all still excuted basic functions.

Android malware commonly executes the following functions: collect a variety of information on the infected mobile phone, including the IMEI number, phone model, as well as the Android OS version, and attempt to get root access on the phone using two

separate exploits. Some of the research questions that have arisen from this are: How to best test android malware applications? What are the limits of mobile security? Why there arent many mobile security testing procedures?

One note about the malware used to test the antivirus applications is that the majority of the malware had names that a user should know better than to install. Most malware detection is signature-based, and future work will include altering known good programs to contain certain malicious signatures without altering the name and/or manifest to reflect these changes.

While loading the malware onto the device, there were two malicious programs that failed to load even when no antivirus was installed. It seems that even malware developers dont always test their code before deploying. Loading the malware was a quick process via the injector and using the injector saved several hours of effort compared to using standard tools to recompile source and load onto the device by the android development suite available through eclipse.

Finally, we chose to use an actual device for these tests to better understand how the malware and antivirus software would work en vivo rather than depend on the emulator available through the android development kit. This gave us an opportunity to see the actual install time and reaction time of the antivirus software as opposed to depending on the emulator environment to base our results on, which often runs very slow even on high power machines.

In addition to running top antivirus solutions, we also began work on a new type of tool. The idea, to be discussed further in the experimental solution section, was to view memory dumps of the OS to assist in determining processes currently running. Using the tool Volatility, we scanned phone memory and were able to obtain a complete list of processes even if the process had been hidden from normal viewing. However, time limitations stopped us from being able to complete our tool. The goal of the tool was to be able to detect maliciously running processes similar to intrusion detection systems for the standard operating system.

5 Results

Below is a table of the results of our experiments. For each antivirus, the twenty malware applications were

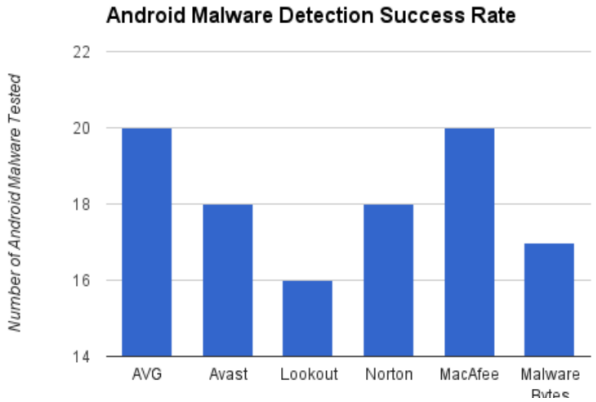

Figure 3: Android Malware Success Rate

sent to the device via the injector. As can be seen in the chart, only AVG and McAfee found all possible infections. In addition to correctly detecting and removing malware, there were two other metrics involved. First, speed of reaction time was taken into account. Secondly, the pro-active nature of the application was observed. Figure 5 is a chart for these secondary metrics. These results are encouraging in the case of AVG but show that most of the popular android antivirus applications are lacking in major ways. All of the malware was at least 2 years old. The fact that any antivirus would miss a piece of malicious software that had been out for 2 or more years greatly reduces our confidence in most of the current antivirus solutions.

Reaction time and proactivity were also eye opening. While most applications responded within about a second of the malware being downloaded, MacAfee was woefully inefficient, needing 2-3 seconds to respond and taking even more time to fully open and allow a user to interact. AVG on the other hand acted before the malware was even downloaded, stopping the malware from download unless the user interacted and accepted the action. Even if the user accepted the action, AVG popped up another warning in under a second for the user to remove the malware. All other applications took about 1 second from the time the malware was downloaded to the time that a message appeared and the user was notified that malware had been installed.

The performance of the tool currently being developed in parallel with this paper is encouraging. By using a list of known malware, we can immediately check apps that are being downloaded for known malicious soft-

Anti-Virus	Failures
AVG	0
Avast	2
Lookout	4
Norton	2
MacAfee	0
Malware Bytes	3

Figure 4: Android Malware Failures

Anti-Virus	Reaction Time	Proactive
AVG	< 1 second	Yes
Avast	1 second	No
Lookout	1 second	No
Norton	1 second	No
MacAfee	2-3 seconds	No
Malware Bytes	1 second	No

Figure 5: Reaction and Proactive Results

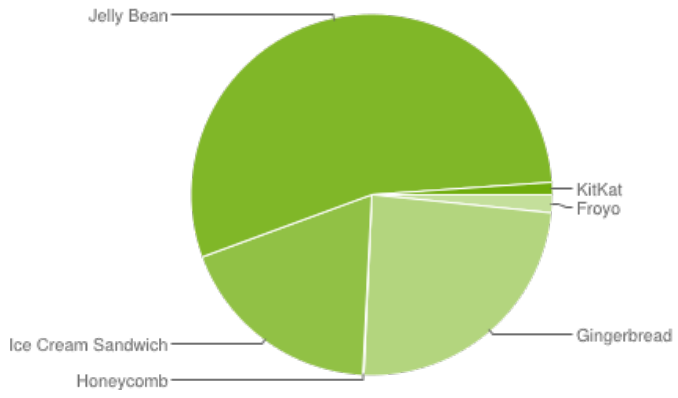

Figure 6: Android Mobile Operating System Usage on Smartphones

[18]

ware. The techniques used in Volatility allowed us to view any process running, even if it was hidden. This in turn gives us a possible heuristic to classify malware, the thought being that if the application is not on a white list and is hiding its activity, it has a higher chance of being malicious. Due to time constraints, this tool is not yet complete, but the initial results are strong enough to warrant further work.

6 Mobile Operating System Security Techniques

The Android operating system has many security techniques implemented. The security is improved with each upgrade to the operating system. However, as demonstrated in Figure 6 most people use Jelly Bean, and have not upgraded to the latest operating system KitKat.

Jelly Bean has a couple of security feactures. Jelly Bean uses SELinux, a mandatory access control (MAC) system in the Linux kernel to augment the UID based application sandbox. This protects the operating system against potential security vulnerabilities. However, in KitKat the updates to the SELinux configuration goes from "permissive" to "enforcing." This means potential policy violations within a SELinux domain that has an enforcing policy will be blocked.[18]

The Jelly Bean KeyChain API provides a method that allows applications to confirm that system-wide keys are bound to a hardware root of trust for the de-

vice. This provides a place to create or store private keys that cannot be exported off the device, even in the event of a root or kernel compromise.[18] Kitkat introduces a keystore provider and APIs that allow applications to create exclusive-use keys. Using the APIs, apps can create or store private keys that cannot be seen or used by other apps, and can be added to the keystore without any user interaction. The keystore provider provides the same security benefits that the KeyChain API provides for system-wide credentials, such as binding credentials to a device. Private keys in the keystore cannot be exported off the device.[18]

KitKat has other advantages over Jelly Bean. For example the /system partition is now mounted nosuid for zygote-spawned processes, preventing Android applications from executing setuid programs. This reduces root attack surface and likelihood of potential security vulnerabilities.[18]

KitKat has improved its security further by adding support for two more cryptographic algorithms. Elliptic Curve Digital Signature Algorithm (ECDSA) support has been added to the keystore provider improving security of digital signing, applicable to scenarios such as signing of an application or a data connection. The Scrypt key derivation function is implemented to protect the cryptographic keys used for full-disk encryption.[18]

KitKat also uses virtual private networks (VPNs). VPNs are now applied per user. This can allow a user to route all network traffic through a VPN without affecting other users on the device. Also, Android now supports FORTIFY SOURCE level 2, and all code is compiled with those protections. FORTIFY SOURCE has been enhanced to work with clang.[18]

7 Conclusions

In conclusion, we discovered that commerical malware fails to ensure proper security for the average Android mobile user. The standard commerical anti-virus at most performs basic signature detection, in which may be sufficient for the time being will definitely be unsatisfying for the mobile phone user in the long run. Mobile anti-virus solutions appear to fail in even fully scanning the Android manifest file which request unneessary permissions. There is work in testing how many applications in the Google Play Store request unnecessary permissions however, they fail to provide a solution.

7.1 Solutions

Security Through The Operating System

Since the confusion comes from the numerous developers were propose the Android operating system provide permission checking based on the hard-coded rules already developed. This has begun to be enforced by KitKat however, as demonstrated in Figure 6 most users are still on the Jelly Bean operating system. Mandatory Updates

Power Consumption

Anti-Virus solutions have other problems. From a mobile device perspective power consumption is a major consideration.Anotable future work would be to also test the power consumption of running an anti-virus on a device, in order to track the loss of performance on the device. but almost no consideration with a traditional Traditional anti-virus solutions can use far more processing power and run longer than those on mobile devices without worrying about draining the battery. Additionally, classic anti-virus doesnt seem to mind hogging all system resources, while mobile device users become disgruntled when the devices slows down. With full integration into browsers and the resources to run applications in sandboxes, standard operating system based anti-virus solutions have much greater flexibility in virus detection and prevention.

Human Computer Interaction

While conducting our research we realized another area to consider is the human computer interaction (HCI). HCI is the study, planning, and design of the interaction between users and computers. It is often regarded as the intersection of computer science, behavioral sciences, design and several other fields of study. The term connotes that, unlike other tools with only limited uses a computer has many affordances for use and this takes place in an open-ended dialog between the user and the computer.

Because humancomputer interaction studies a human and a machine in conjunction, it draws from support-

ing knowledge on both the machine and the human side. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, linguistics, social sciences, cognitive psychology, and human factors such as computer user satisfaction are relevant. Engineering and design methods are also relevant. Due to the multidisciplinary nature of HCI, people with different backgrounds contribute to its success.

Attention to human-machine interaction is important because poorly designed human-machine interfaces can lead to many unexpected problems. Mobile devices brings a whole knew concept. Due to the limited screen size and less ability to do fine grained manipulation on a mobile device, the interfaces are often far less powerful and give the user less control over the actions taken. This is especially noticeable in McAfees mobile solution which tries to to replicate its horrific grandeur from traditional systems. It slows down the system considerably.

7.2 Experimental Solution

We suggest further work on the tool started in parallel with this paper. Current malware detection techniques are lacking in their ability to discover hidden processes and associate certain activity spawned by a program with malicious intent. Using the techniques from volatility, we believe that we can fully integrate full memory scans to determine when processes a hidden or are executing subprocesses that could potentially be malicious. These actions include, but are not limited to: mass texts, sending of information from the phone that is not generated in the app, spawning hidden processes, spawning multiple new processes in a short period, sending messages to system processes or other apps that are unrelated, and blocking access to antivirus sites or system updates. I hope would be that this tool could be integrated into the mobile operating system for enhanced security and better resource distribution.

After performing the study, we were able to determine certain things about Android mobile security. First, there is need for access to current android malware in order to properly test security tools. Second, there is need of open source collaboration between academic and industry in order to share and create security testing tools for android.Third, industry may need to allow open source versions of their security applications in order for better research to be conducted in this area. Lastly, the best way to enhance mobile security would be from the operating system and not from external applications. Malware security is a threat to the privacy of its users and enahcned mobile operating system security is determined as the best method.

References

[1] Buennemeyer, Timothy, *Battery-Sensing Intrusion Protetion System (B-SIPS),* 2008.

[2] Bose, Abhijt, Hu, Xin, Shin, Kang, and Park, Taejoon, *Behavioral Detection of Malware on Mobile Handsets,* 2008.

[3] Cheng, Jerry, Wong, Starsky, Yang, Hao, and Lu, Songwu,*SmartSiren: Virus Detection and Alert for Smartphones,* 2007.

[4] Enck, William, Ongtang, Machigar, and McDainel, Patrick, *Understanding Android Security* 2009.

[5] Eston, Tom, Mannino, Jack, Ashokkumar, Sreenarayan, Deshmukh, Swapnil, Knight, Brandon, Jensen, Steve, Modi, Shimon, Marcos, Rodrigo, Clark, Brandon, Quemener, Yvesmarie, Paralikar, Yashraj, and Taank, Ritesh, *OWASP Mobile Security Project - Mobile Threat Model,* 2012.

[6] Felt, Adrienne, Finifter, Matthew, Chin, Erika, Hanna, Steven, and Wagner, David, *A Suvery of Mobile Malware in the Wild,* 2011.

[7] Jacoby, Grant, *Battery-Based Intrusion Detection,* 2008.

[8] Martin, Hsiao, Michale, Ha, Dong, and Krishnaswami, Jayan, *Denial-of-Service Attacks on Battery-powered Mobile Computers,* 2004.

[9] Maiorana, Emanuele, Campisi, Patrizio, Gonzalez-Carballo, Noelia, and Neri, Alessandro, *Keystroke Dynamics Authneitcation for Mobile Phones,* 2011.

[10] Oberheide, Jon and Jahanian, Farnam, *When Mobile is Harder Than Fixed (and Vice Versa): Demystifying Security Challenges in Mobile Environments,* 2010.

[11] Zhou, Yajin, Jiang, Xuxian, *Android Malware Genome Project,* 2012.

[12] Schmidt, Aubrey-Derrick, Peters, Frank, Florian, Lamour, and Albayrak, Sahin, *Monitoring Smartphones for Anomaly Detection,* 2008.

[13] Shabtai, Asaf, Fledel, Yuval, Kanonov, Uri, Elovici, Yuval, and Dolev, Shlomi, *Google Android: A State-of-the-Art Review of Security Mechanisms.*

[14] Yap, Teck and Ewe, Hong, *A Mobile Phone Malicious Software Detection Model with Behavior Checker,* 2005.

[15] Zhou, Yajin and Jiang, Xuxian, *Dissecting Android Malware: Characterization and Evolution,* 2012.

[16] Zhou, Yajin, Wang, Zhi, Zhou, Wu, and Jiangm Xuxian, *Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets,* 2012.

[17] Glynn, Fergal, *Android Security Guide to Android OS, Mobile Application Security — Veracode,* 2013.

[18] *Android,* www.android.com.