XXXXXXXX
CSC 470 D Android Application Development
Fall 2014

A Layered Approach to Android Security

The Smartphone is a fairly new concept in the world of computer technology. Likewise, malware for this platform is also in the early stages of its smartphone evolution. Computer technology has been around for quite a bit longer and it's no surprise that malware developed for Windows/Mac/Linux OS platforms has become extremely dangerous and stealthy. It is not difficult to predict that mobile OS malware will continue to mature with mobile OS technologies. Likewise, malware tends to "pick on" the most ubiquitous Operating System, which in the case of mobile, is Android. Unfortunately, the availability of source code is Android's strongest and weakest point. Although it allows for a greater understanding of how the OS functions and provides an open door for developers world-wide, bad guys have the blueprint for ins-and-outs as well.

In order to stay ahead of (or just catch up with) this malware cat and mouse game, antivirus researchers and companies have been looking for novel approaches to analysis. The question becomes: "How does one analyze, test, and identify new malware destined for the Android platform and can this be done efficiently?" The answer is that researchers are borrowing ideas from existing technologies and practices in the PC world. As each new model of phone and mobile OS update is brought to market, it seems PC-like functionality is outweighing the mobility aspect. It is obvious that "phone" only describes a miniscule percentage of what these devices can do and consumers have come to expect a convenient yet powerful PC experience. This gives viruses, trojans, and worms more opportunities for infections as users are doing more from their phone than the home computer. This not only pertains to smart phones, but tablets as well, since we know that Android is also a player in this market.

A successful rudiment for malware analysis should contain both static and dynamic analysis. This idea is consistent throughout much of the research literature devoted to malware. According to Blasing et al. [1], *Static Analysis* "involves various binary forensic techniques, including decompilation, decryption, pattern matching, and static system call analysis". This is the type of analysis you would expect with Antivirus software that utilizes a database of signatures against which files and memory can be scanned. Unfortunately, it is extremely difficult to automatically detect polymorphic code this way due to the fact that it involves encryption and mutation. This can be remedied by using *Dynamic Analysis*, which means "running an application in a controlled environment and monitoring its behavior [1]". By using this technique, the program's actions can be monitored for anomalous behavior while it runs. Unfortunately, with great flexibility also comes a drain on resources, which is why the author describes this potential service as cloud-based [1].

The safest environment for working with both types of analysis is known as a sandbox. However, it is especially important for *Dynamic Analysis* because it basically enforces boundaries within which a malicious program can execute. Based on previous research, it has been determined that a kernel-level sandbox is preferred because it's easier for malware to detect one at the user-level. A kernel-level sandbox allows a researcher to capture system calls at the lowest level of the Linux-based system [1]. In this way, nothing gets past the monitoring. Perhaps a future direction of research would be finding a way to develop an anti-malware program that is lightweight on device, a minimal resource drain, and works at or below (BIOS) the kernel level. Since there has been an increase in rootkit threats within the last several years, we know that companies like McAfee have already developed software that works below the OS for PCs [2], so the idea of getting to this low-level is a natural progression. The malware is getting there, so the development of adequate defenses should follow.

Combining static and dynamic analysis is nothing new and is perhaps best illustrated in an example by Dai et al. *Droidlogger* is an instrumentation tool that decompiles the code, puts it into an intermediate language, uses an instrumentation tool to search for suspicious API's, logged to file, then it is compiled, re-packaged, and run again [3]. This approach has the advantage of providing in-depth analysis of API calls once the application is broken down. In this way it acts as a type of a host-based intrusion detection system, although not in real time. Additionally, in the case of a polymorphic virus, the hybrid nature of this tool allows for the capture of plain text normally hidden through encryption. This is done by extracting it from runtime data during the "dynamic" phase of its run. As with any new technical research tool, this also has drawbacks. Since some malware does an integrity check before executing, it may not run since this tool breaks that integrity. Thus, the program is not reviewable. Another issue is that any native code run by malware cannot be detected. This tool was created so that program calls from above the kernel (Java/Dalvik VM) could be efficiently intercepted; not calls originating from the lowest layers. From this research, it is clear that *Droidlogger* is an appropriate representation of layered security analysis. Unfortunately, it is not currently realistic as an on-device protection option due to the amount of manual intervention that appears to be involved. This idea, however, is certainly on the right track to a suite of security tools that would be very useful for mobile malware research.

In order to understand what types of techniques might be available to researchers and those on the front lines of the growing mobile malware threat, it is a good idea to categorize what is already known about them. Amamra et al. describes a classification system that breaks down different techniques into *Signature-Based* and *Anomaly-Based* [4]. *Signature-Based* is further broken down into *Static* and *Behavior-Based*. Here, static means that signatures are kept in an updated database that allows software to scan against it. This is a common component in commercial AV software and has the advantage of being efficient and CPU-friendly. However, this alone is not able to detect unknown malware and may not be able to detect obfuscating malware. *Behavior-Based* uses complex meta-structures that are better equipped to deal with this type. This can also be split into classifications of *Static*, which can detect malware before execution, and *Dynamic*, which can profile the exact behavior of malware during execution. The other parent classification, *Anomaly-Based*, consists of building a reference baseline for behavior, and is also broken down into *Static* and *Dynamic*. These techniques are complimentary to the others in that they excel in zero-day attacks and unknown malware detection. The main drawback is that they are complex and resource intensive, which doesn't bode well for limited resources of smartphones. However, I believe that we will see this category of defense play a more important role as mobile devices continue to gain resource improvements (more RAM, CPU cores, storage, etc.) with each new generation.

While it is easy to see the importance of classifying analysis and detection techniques, it is equally important to mention the crossroads between mobile device structure and security. Lou et al. describes a hierarchical security framework which proposes general system improvements and security tools in relation to various layers of the device [5]. One interesting point is that system improvements are noted along with security tools as a component to overall security. Suggestions such as secure boot and OS-provided firewall are given as a way to bolster protection. One other suggestion that falls under the system improvement category is User Confirmations during installations. From my own experience, I know that users can decide to install an app based on the requested permissions found in the application manifest. This is a weak way to enforce security because most users pay no attention and don't really understand the request anyway. This could be greatly improved by requiring developers to disclose why certain permissions are required for the application; not simply what the app is requesting. Alternatively, the Play Store might be able to require some type of analysis like *Droidlogger* where the API calls are tested vs. what's provided in the manifest. It is obvious that the instrumentation tools can be developed and enhanced for this type of automation. Perhaps a future update to Android will include a framework for detailing requested access permissions. I could envision an automated process that kicks off upon an App submission to the Play Store. Once the analysis has determined the status of the program, it makes a decision whether or not to allow final approval.

I predict that mobile malware will continue to test the limits of mobile OS's and security software companies that are attempting to protect them. As time moves forward, we will continue to observe a convergence of these programs from the PC and mobile realm. Malware developers recognize more users are opting for a tablet or smartphone as opposed to a desktop or laptop computer. This can be clearly seen in one of the world's most ubiquitous Operating Systems: Windows. Windows 8, although used with PC's, tablets, and smartphones, is clearly geared toward mobile. Apple has already trended this way for quite some time. There is always a progression of bad with the good, and the same can be said for the current state of mobile. However, if developers can continue to improve upon research and reach the point of an efficiently combined package that includes an on-host, layered approach, the mobile security landscape would look much more inviting.

1 Bläsing, T.; Batyuk, L.; Schmidt, A.-D.; Camtepe, S.A.; Albayrak, S., "An Android Application Sandbox system for suspicious software detection," Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on , vol., no., pp.55,62, 19-20 Oct. 2010

2 McAfee/Intel, "Root Out Rootkits, An inside look at McAfee Deep Defender", McAfee Whitepaper, 2012

3 Shuaifu Dai; Tao Wei; Wei Zou, "DroidLogger: Reveal suspicious behavior of Android applications via instrumentation," *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*, vol., no., pp.550,555, 3-5 Dec. 2012

4 Amamra, A.; Talhi, C.; Robert, J., "Smartphone malware detection: From a survey towards taxonomy," Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on , vol., no., pp.79,86, 16-18 Oct. 2012

5 Hongwei Luo; Guili He; Xiaodong Lin; Xuemin Shen "Towards hierarchical security framework for smartphones", Communications in China (ICCC), 2012 1st IEEE International Conference on, On page(s): 214 – 219