Androsafe: Online malware analysis with static and dynamic methods

Krishnadeva Kesavan, Chethana Liyanapathirana , S.A.W.S Sampath , Y.M. Sureni Koshila,Chamod Premarathne ,Sahan Wanigarathna, Chamira Priyamanthi Nawarathna, Prabhath Lakmal Rupasinghe

ABSTRACT

With an estimated market share of 70% to 80%, Android as becoming the most popular operating system for smartphone and tablet. Cyber criminals naturally expanded their various activities towards Google's mobile platform.An additional incentive for mobile malware authors to target Android instead of another mobile platform is Android open design that allows users to install the application from a variety of sources. "Androsafe" is an online malware analysis tool which can analyze malware in an isolated environment without any damaging to the mobile device by using both existing and new anomaly based and behavioral analysis. Through this combination, we can analyze a large number of malware families because some malware families may only perform signature base or behavioral. Then the sandboxes based on signature will not have analysis malware families that only perform a behavior and the sandboxes based on behavior will not analysis signature-based malware families."Androsafe" sandbox will be hosted in the Honeynet Project's cloud. Dynamic Analysis will be queued and run in the background, and an email which contains malware analyzing report will be sent to the user when the analysis is over. This method is very efficient more than offline kernel and app base sandbox.

1. INTRODUCTION

In the past couple of years, mobile devices have become sophisticated computing environments with increased computing power and network connectivity.

Android is a modern mobile platform that was designed to be open. Android applications make use of advanced hardware and software, as well as local and served data, exposed through the platform to bring innovation and value to consumers. To protect that value, the platform must offer an application environment that guarantees the security of users, data, applications, the device, and the network.

This open design easily allows attackers to perform attacks, such as social engineering attacks to mislead device users to install malware and attack third-party applications on Android. So we need to secure Android mobile device from this kind of common attacks. Securing an open platform requires strong security architecture.

When Android users download android applications from App stores to the mobile device, the user cannot test whether the application is secure or not. There can be hidden malicious codes inside the applications. To avoid such situations, we can test downloaded applications using sandboxing technology. Those sandboxes check applications for malware in offline mode. There's a risk behind that because offline checking can be harm to the mobile device. Attached malicious codes can breach the boundaries of the sandbox while analyzing the downloaded applications for malware.

There are two kinds of malware analyzing methods. One is a static method, and the other one is a dynamic method. Sandboxes currently available are based on either static or dynamic method. Then the analyzing scope will be less or limited. To overcome this limitation, we use both static and dynamic techniques to analyze malware.

We allow users to analyze downloaded applications by using the online tool "Androsafe." This reduces large impact over mobile devices by redirecting to an isolated environment apart from the mobile device to test downloaded applications, and there won't be any harm to the mobile device.

2. RESEARCH OBJECTIVE

The **outcome** of the research is an online Android sandbox "Androsafe" which is used to test suspicious programs that may contain a virus or other malicious code, without allowing the application to harm the mobile device.

We develop "Androsafe" to detect malware inside the downloaded applications. This online android application sand box is capable of analyzing malware based on both static and dynamic methods.

Among static and dynamic malware analyzing tools, we decide to select Androguard as a static analyzing tool and Droidbox as a dynamic analyzing tool. Then we develop an Androguard algorithm inside the Droidbox algorithm in order to produce a combined module.

We first analyze the Androguard algorithm and find the malware analyzing techniques. The existing Androguard is able to detect only less number of malware families, and most of the time Androguard cannot detect the entire malware family. To address that problem we will develop the existing algorithm to detect those unknown malware families. Then the "Androsafe" algorithm is developed by inheriting the features from the "Droidbox" to the above-mentioned modified "Androguard" algorithm.

Droidbox will do dynamic analysis, and this will check for the behavior of malware. Droidbox is a tool to analyze Android apps. However, it lacks support to track native API calls. In fact, the current dynamic analysis methods running out of a method to track the native API calls during dynamic analysis. So we introduce the native API calls tracking mechanism to the current Droidbox algorithm.

We provide an isolated workspace in "Androsafe" to test any APK as an example chatting applications, gaming applications, etc. based on Android platform.

3. METHODOLOGY

Androguard is a python tool which is using static analysis method to detect malware in Android devices. Currently, Androguard detects several malware families such as Obad, Geinimi, DroidKungFu, Bracebridge. However, Androguard is lagging behind in detecting all the malware families such as FakeInstaller, etc. Our target is to enhance the current androguard algorithm to identify all malware families.

Androguard works with:

- Dex/Odex Dalvik virtual machine, .dex disassemble DE compilation.
- APK Android application.
- Android is binary XML.
- Android Resources.

Androguard has the following features:

It Maps and manipulates DEX/APK format into full Python objects.

- Disassemble/Decompilation/Modification of DEX/APK format.
- Decompilation with the first native directly from Dalvik byte codes to Java source codes Dalvik decompiler.
- Access to the static analysis of the code basic blocks, instructions, permissions and create analysis tool.
- Analyze a bunch of Android apps.
- Diffing of Android applications.
- Check if an Android application is there in a database.
- Open source database of Android Malware
- Reverse engineering of applications
- Transform Android's binary XML like AndroidManifest.xml into classic XML.

Basically, in Androguard, reverse engineering process is following the below steps.

1. Analyzing the Android-Manifest for permissions and activities

2. Unpacking of the Android application (apk file) to get all files and especially the classes.dex

3. Translating the Dalvik-Bytecode to Java-Bytecode

4. Analyzing the generated code

However, the problem is when Androguard gets the APK and try to unzip the APK, some malware generates anti-debugging tricks to crash Androguard. To overcome this problem we try to develop an algorithm to detect this kind of malware with anti-debugging tricks.

DroidBox consists of two parts, which can be referred to as the Host and the Target. The Target part launched on the emulator that is based on Android to monitor the data in low level.

The Host part is a set of Python scripts that connect with the emulator and receives all possible information from the Target regarding the application being analyzed and displays it in text or graphic format. DroidBox will output its results as a

JSON file.

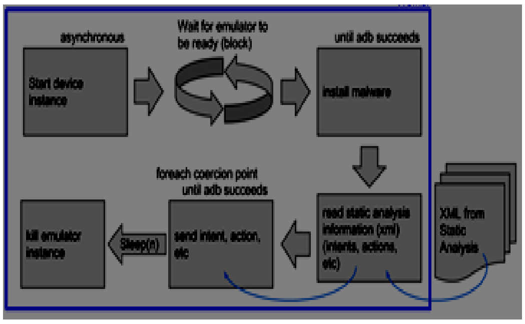

Figure 1: DroidBox Architecture

Dynamic analysis can monitor an APK's behavior utilizing following techniques in Droidbox.

- Taint tracking: Taint tracking tools are used in the dynamic analysis to detect potential misuse of users' private information.

- Virtual machine introspection (VMI): This intercepts events that occur within the emulated environment. Dalvik VMI-based systems monitor the execution of Android APIs through modifications in the Dalvik VM. Qemu VMI-based systems are implemented on the emulator level to enable the analysis of native code.

- System calls monitoring: Frameworks can collect an overview of executed system calls, by using, for instance, VMI, "strace" or a kernel module. This enables the tracing of native code.

- Method tracing: Frameworks can trace Java method invocations of an Apk in the Dalvik VM. It looks for malware family samples which are not detected by existing Droidbox.

According to the research done based on detecting Android Malware on Network Level in 2011, they found out several malware that are detected and not detected.

Not Detected Malware:

- Flexispy – Trojan horse family
- lovetrap - Trojan horse family
- kungfu – DroidKungFu family
- droiddeluxe – Droiddeluxe family
- basebridge – Basebridge malware family
- ggtracker - ggtracker family
- netisend and droiddream.
- Spygold and zone executed,

Table I – Detected Malware

Name	Identifying information
Crusewind	crusewind.net, HTTP-text, IMEI
walka	incorporateapps.com,
Tonclank	searchwebmobile.com, http-json IMEI
Bgserv	www.youlubg.com, IMEI, phone number
Smspacem	biofaction.no-ip.biz, HTTP-soap phone number
Lovetrap	cooshare.com, http-text positionrecorder.asmx, IMSI
(DL/installer)	api.go108.cn, http-xml no-store no-cache, love more and more

4. RESULT AND DISCUSSION

We are planned to get two sets of malware samples from different malware families for dynamic and static analysis separately. Then we analyze one sample using Androguard, and another sample will analyze using Droidbox. So far we have identified several malware samples from different malware families were not detected by Droidbox. We have considered malware samples from Trojan horse family, DroidKungFu family, Droiddeluxe family, Basebridge malware family, ggtracker family, netisend, and droiddream. Now we are planning to move this samples to check whether Androguard analyze them or not.

We use malware free Android applications from the third party app store called "China App store" to analyzing purpose. As well as we will maintain a malware signature database for static analysis. New Signature findings will send and store in the signature database. We use "Santoku 0.5" virtual machine in Linux Operating system.

We develop "Androsafe" over Linux platform and nowadays we are developing "Androsafe" web application which is used to submit the unverified APK file by users. At the same time, we are identifying useful and essential features that we can add from Androguard to Droidbox and redirect them to advance online malware analysis tool called "Androsafe."

"Androsafe" generates a report including personal or sensitive device information, Security appropriate actions, and logs regarding the access to hardware modules or sensors and communication information. We analyze, does the app try to access to the local address book, local calendar, stored pictures, configured accounts, local SMS or MMS messages, device identifiers and SIM card identifiers as personal or sensitive information.

As well as check security relevant actions like does the app use Crypto, load external libraries, try to modify device settings or try to install additional apps. We evaluate hardware modules or sensors deeply whether the app tries to use the camera, microphone, try to locate the device using the GPS sensor or network triangulation and does the app communicate with the Internet or the cloud services, Does the app try to send SMS messages, try to start a phone call, or try to open local ports.

5. CONCLUSION

Android is a free and open platform. This is also an advantage for developers as well as users. However, this will redirect many more security causes. This openness is a blessing as well as a curse for Android users. Anyone can put their Android applications in Android Marketplace. We cannot trust these applications are actually from developers or is there any hidden malicious code behind these third-party applications before install to our mobile devices. Most existing malware analyzing mechanisms are Kernel-based. In these mechanisms, the downloaded applications are checked for malware inside the mobile device, and it is very dangerous because sometimes infected malware can be activated inside the mobile device while analyzing and it can corrupt the entire system.

In our research, we develop an online Android application sandbox "Androsafe" by combining enhanced Androguard and Droidbox inside Droidbox to detect a significant amount of malware within an isolated environment hosted on a server. It is used to test unverified programs (APK) that may contain a virus or other malicious code, without allowing the application to damage the mobile device. Finally "Androsafe" generates an analyzing report including a summary of finding malware inside the downloaded APK. "Androsafe" is user-friendly, convenient, freely available online malware analyzing tool that anyone can access without any fear instead install downloaded Android applications blindly to your mobile device.

6. FUTURE WORK

This research is mainly focused on Android platform, but in the near future, it can be deployed for other mobile platforms. Currently, we are developing "Androsafe" web application, after that in the second step; we are supposed to develop an Android application. Our web application is developed in a way that any audience can use it with minimal effort. Also, we hope to develop this kind of features inside the "Androsafe" mobile application as well. We are aware of Malware and will develop "Androsafe" to identify the malware within the application. We are providing a user-friendly environment to the users in order to report any issues regarding the application like problems regarding malware. We are going to maintain a separate blacklist and whitelist. After we test the reported applications of users, we are going to group the applications separately. Applications with malware and bugs are labelled as blacklist applications, and the applications without any unauthorized modifications are labelled as whitelist applications. Users can download the application from our whitelist in the future. This will reduce the risk of downloading infected malware applications.

7. REFERENCES

[1] J. Bergeron, M. Debbabi, J. Desharnais, M. M. Erhioui, Y. Lavoie, and N. Tawbi. Static detection of malicious code in executable programs. In Proceedings of the Symposium on Requirements Engineering for Information Security (SREIS'01), 2001.

[2] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In Proceedings of the 23rd Annual Computer Security Application Conference (ACSAC), pages 421–430, 2007.

[9] C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using cwsandbox. IEEE Security and Privacy, 5(2[3] P. Szor. Virus Research and Defense. Addison-Wesley, 2005.

[4] A.-D. Schmidt, J. H. Clausen, S. A. Camtepe, and S. Albayrak. Detecting Symbian os malware through a static function call analysis. In Proceedings of the 4th IEEE International Conference on Malicious and Unwanted Software (Malware 2009), pages 15–22. IEEE, 2009.

[5] Thomas Bl¨asing, Leonid Batyuk, Aubrey-Derrick Schmidt, Seyit Ahmet Camtepe, and Sahin Albayrak "An Android Application Sandbox System for Suspicious Software Detection", in Technische Universit¨at Berlin 2010

[6] A. Desnos and G. Gueguen, "Android: From Reversing to Decompilation," in Black Hat Abu Dhabi, Dec. 2011.

[7] S. Forrest, S. Hofmeyr, and A. Somayaji. The evolution of system-call monitoring. In ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference, pages 418–430. IEEE Computer Society, 2008.

[8] M. A. Bishop. The Art and Science of Computer Security. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[10] A. Dewald, T. Holz, and F. Freiling. Moreover, sandbox: Sandboxing javascript to fight malicious websites. In Symposium on Applied Computing (SAC) 2010, Sierre, Switzerland, March 2010.

[11] T. Raffetseder, C. Kruegel, and E. Kirda. Detecting system emulators.

[12] M. Becher, F. Freiling, and B. Leider. On the effort to create smartphone worms in windows mobile. In Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC, pages 199–206, 20-22 June 2007.

[13] Bundesamtf¨urSicherheitin derInformationstechnik. Mobile endger¨ateund mobile application: Sicherheitsgef¨ahrdungenund schutzmassnahmen, 2006.

[14] W. Enck, M. Ongtang, and P. McDaniel. Understanding Android Security. IEEE Security and Privacy, 7(1):50–57, 2009.

[15] S. Forrest, S. Hofmeyr, and A. Somayaji. The evolution of system-call monitoring. In ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference, pages418–430. IEEE Computer Society, 2008.

[16] A. Rubini. Kernel system calls. http://www.ar.linux.it/docs/ksys/ksys.html. [Online; accessed 01-March-2010].

[17] GSM Association. IMEI allocation and approval guidelines, 2010. Available online at HTTP:// www.gsmworld.com/documents/DG06_v5.pdf; visited on December 4th, 2011.

[18] Eric Chien. Motivations of recent Android malware. Available online at http://www. symantec.com/content/en/us/enterprise/ media/security_response/whitepapers/ motivations_of_recent_android_malware. pdf; visited on December 4th, 2011.

[19] Gerry Eisenhaur, Michael N. Gagnon, Tufan Demir, and Neil Daswani. Mobile malware madness and how to cap the mad hatters, 2011. Available online at https://media. blackhat.com/bh-us-11/Daswani/BH_US_ 11_Daswani_Mobile_Malware_Slides.pdf; visited on December 4th, 2011

[20] Google management discusses q3 2011 results, 2011. Available online at http://seekingalpha.com/article/ 299518-google-management-discusses q3-2011-results-earnings-call-transcript; visited on December 4th, 2011

[21] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. IMEI allocation and approval guidelines, 2009. Available online at http://www.cs.ucsb.edu/~seclab/ projects/torpig/torpig.pdf; visited on December 4th, 2011