Infrastructure for detecting Android malware

Laurent Delosières[1] and David García[2]

[1] Hispasec Sistemas S.L., 29590 Campanillas (Malaga), Spain
ldelosieres@hispasec.com
[2] Hispasec Sistemas S.L., 29590 Campanillas (Malaga), Spain dgarcia@hispasec.com

Abstract. Malware for smartphones have sky-rocketed these last years, particularly for Android platforms. To tackle this threat, services such as Google Bouncer have intended to counter-attack. However, it has been of short duration since the malware have circumvented the service by changing their behaviors. Therefore, we propose a malware taxonomy, a survey of attack vectors to better understand the Android malware, a survey of the modus-operandi of attackers for infecting the smartphones, and the design of components that are responsible for analyzing and detecting Android malware of the NEMESYS infrastructure. This infrastructure aims at understanding and detecting attacks both at the network and smartphone level.

Keywords: Android; malware; infrastructure; detection

1 Introduction

Malware represent a high threat for smartphone users since they affect all the plaftorms [1] by stealing personal data, banking data, transforming the smartphone in bots, etc. Most scaring, their number has tremendously exceeded the predictions for the year 2012 by a factor up to 300% as shown by TrendMicro [2]. Furthermore, in 2012, their number has continuously increased as revealed by the F-Secure report [3].

Even though antiviruses for mobile phones and other services such as the service Google Bouncer for detecting Android malware exist, the number of infections is still very high, especially for Android platforms as shown by McAfee [4].

This paper is about introducing the infrastructure for analyzing and detecting malware, part of the NEMESYS infrastructure [5]. We propose a malware taxonomy and a study of 38 Android malware samples to identify the most common Android families. We also survey the attack vectors i.e., the modus-operandi of attackers to infect the smartphones. Finally, we overview the NEMESYS infrastructure and show the current development of the infrastructure concerning the analysis and detection of Android malware.

The infrastructure comprises components both on the smartphone and on the server. The component on the smartphone logs the behavior of the Android programs which are instrumented by the user and uploads them to a server for an offload analysis, and early detects the malware. As for the component on

the server side, it makes a more complete analysis of the malware and embeds a crawler in order to crawl the web searching for Android applications both goodware and malware.

The contributions of this paper are (1) proposing a taxonomy for the attack vectors, (2) surveying the different Android malware families, and (3) designing the components of the NEMESYS infrastructure that are responsible for analyzing Android malware.

The paper is structured as follows: Section 2 introduces the terms for understanding the article. Section 3 proposes a taxonomy of Android malware. Section 4 surveys the different Android malware families and a study of 38 samples is proposed in order to identify the main Android malware families. Finally, Section 5 overviews the NEMESYS infrastructure with an emphasis on the components for analyzing and detecting Android malware.

2 Background

GooglePlay [6], formely called Android Market, is the official Android application store. It is accessible via the browser or via the Android application "Play Store". Both of them enable to browse Android applications on the official Google repository by their category, their name, and download them. Alternative stores for Android applications exist such as Amazon Appstore [7], GetJar [8], etc.

VirusTotal is a free service for analyzing samples by 44 different antivirus engines. The service issues a report containing the malware name if so, the hash ID of the sample (MD5, SHA1, and SHA-256), the initial filename, the type of file (e.g., an image), etc. For some samples, the sample behavior is also inserted in the report. By behavior, we intend the actions of the malware in the system such as the files that are read, written, the communications that are established, etc.

Recovery mode is a security added by each phone manufacturer in case the main filesystem gets corrupted. Recovery mode consists in dividing the media into two partitions: one for the system and the other one for the copy of the system with the factory settings. The recovery mode is also used for different purposes such as installing a custom image (e.g. CyanoGenMod [9]).

Exploits are used for installing automatically malware on the mobile device or crashing the device making it unavailable. For this, they need a software vulnerability, i.e., a flaw in a program that is exploited in order to inject and run external code. An exploit can be used in every layer of a smartphone from the kernel layer up to the applications layer containing the browser application. By adding more functionalities to the device such as Near Field Communication (NFC), more code is added to the smartphone and therefore it is more likely to increase the number of exploits.

Every Android application is executed inside its own Dalvik Virtual Machine (VM) [10]. It is similar to Java VM but Dalvik byte codes are executed instead of Java byte codes. Unlike Java VM, Dalvik VM is a registered-machine. In other words, registers are used for passing functions' parameters while Java VM uses

the stack instead. Dalvik VM has been used to keep the applications confined but mainly to keep the portability of Android applications between different smart-phones by abstracting their hardware.

Android Debug Bridge (ADB) is the application provided by Google in order to communicate with the phone from the computer. This application enables a user to install, remove, execute, and terminate applications.

3 Attack vectors

A possible taxonomy for the attack vectors is as follows: physical attacks, social engineering, and exploits (applications, drivers and kernel).

3.1 Physical attack

A physical attack consists in accessing physically the device in order to install or remove software. For the Android devices, an attacker can easily access it by connecting it through a USB [3] cable and use either the program *adb* provided by Google or the recovery mode of the phone to install or remove software.

In the former case, an attacker use the most common way for installing and removing software. The tool *adb* can be used for installing a program that will exploit a vulnerability of the Android OS. If the exploit is successful, the attacker can install a rootkit and therefore being stealthy.

In the latter case, the user can take advantage of the recovery mode of the phone to customize the phone firmware. By handcrafting a system image and overwriting the recovery image, the attacker can overwrite the system image by activating the recovery mode which will install the recovery image, i.e., the handcrafted system image. Nevertheless, the attacker needs to know both the manufacturer and the model version for crafting the correct image that will be accepted by the smartphone. Furthermore, this attack will wipe out the user data.

3.2 Social engineering

Social engineering is based on deception. It targets the human instead of the machine. The main techniques in the mobile environment are the phishing and pharming, and the application repacking attacks.

Attackers can use the phishing attack which consists in acquiring information such as usernames, passwords, and credit card details by masquerading as a trustworthy entity. For instance, the attacker sends an email to a user containing a link to a malicious server or to a repository [11]. Pharming [12] is also commonly used. It redirects the user to a fraudulent website by changing the victim's host file or compromising a DNS server, i.e. by poisoning it.

The application repacking is also very common. It consists in decompiling an application, injecting some malicious code, and re-compiling. Most people

[3] Universal Serial Bus

using mobile devices, even though aware of the presence of malware, cannot distinguish malware from goodware which benefits to the attacker. Therefore, the attackers can suggest malicious applications which look like good applications. For attracting users, generally the attackers propose a free version of an Android application.

3.3 Exploits

Since the beginning of Android, several vulnerabilities at every level of the smartphone have been revealed. For instance, vulnerabilities that were found in the drivers, in the NFC, SMS stack, and the Android browser.

Drivers added by the phone manufacturer turn out to be vulnerable. That is the case for instance of the buggy driver developed by Samsung [13]. The exploit uses a bug in the driver to gain root access to the phone i.e., the highest privileges.

Near Field Communication (NFC) has been shown as being vulnerable. Miller et al. [14] have injected and run some malicious code in an Android smartphone by means of the NFC. For this, they have exploited the memory corruption bug of an Android 2.3 to gain the control of the NFC daemon with a specific designed RFID tag. The MWR Labs hacking team [15] has used a zero day vulnerability on the NFC in order to hack the Samsung Galaxy S3. With the exploit, the team had access to the user data of the phone such as e-mails, SMS databases, the address book, the photo gallery, and access to third-party application data.

SMS can also be used to exploit a vulnerability of a smartphone [16]. For instance with the Android 1.5, a malformed SMS message can cause the crash of the system. The user is therefore forced to restart the mobile phone. A continuous reception of such SMS can be used as Denial of Service in which case the mobile phone becomes unavailable.

Android browser has also turned out to be vulnerable. G. Kurtz et al [17] have presented a Remote Access Tool (RAT) which can infect the smartphones with Android 2.2 by exploiting a bug in the Android WebKit browser. Unlike RAT, BaseBridge [18] exploits a breach in ADB for elevating its privileges.

4 Malware taxonomy

There is no standard taxonomy. For example, the taxonomy is different between the three main malware taxonomy standardization namely the Computer Antivirus Research Organization [19], Common Malware Enumeration [20], and Malware Attribute Enumeration and Characterization [21]. Therefore, we propose the following taxonomy w.r.t the malware behavior.

Bot transforms the smartphone in a robot waiting for commands from a Command and Control (C&C) server. Bot might be used to launch a Denial of Service (DoS), a spam campaign, etc.

SMS sender sends SMS which are mostly premium messages without the user awareness.

Infostealer steals private information such as personal user information, GPS location, or device technical data.

Privileges elevator elevates its privileges by exploiting a vulnerability. By gaining root privileges, malware may improve its stealthiness profile and operation capabilities on the resources of the device.

Rootkit uses stealth techniques for covering its traces. However, rootkits need root privileges.

Trojan bank steals banking data. The proliferation of online mobile bank application makes the device a target for scammers.

Adware displays ads, search results, application ratings or comments to the user.

Dropper installs a malware in either one stage or two stages. The one stage dropper consists in installing the malware that is camouflaged in the dropper whilst the second stage consists in downloading the malware from a remote source and installing it.

In order to know the common Android malware characteristics, we have analyzed a set of 38 Android malware. Out of these 38 samples, only 6 of them employ an exploit to elevate their privileges. Almost half of them send premium SMS as a fast way for earning money and transform the smartphone in a Bot. Most of the applications use Social Engineering for infecting the smartphones. Malware seen so far do not embed any anti-debugging or anti-virtual machines as the malware targeting Desktop PCs but we can expect changes in the near future.

5 Infrastructure

The infrastructure defined in the NEMESYS project aims at understanding and counterattacking attacks occurring in the network and in smartphones. The components in which we are involved in encompass a honeyclient and a data collector on the server side, and a lightweight malware detector on the smartphone side. We shall first enumerate the role of each component, and then emphasize the current development of the honeyclient.

5.1 Components of the infrastructure

The lightweight malware detector aims to detect Android malware on the smartphone. It will be efficient and lightweight in terms of memory and CPU consumptions. The detection will be based on two ways: signature and abnormality behavior based. The detector will also interact with the mobile honeypot and the data collection infrastructure in order to download the latest malware signatures and upload traces.

The high-interaction honeyclient development is responsible for interacting with web servers searching for attacks. It is consisted of three components: a queuer which is responsible for creating a list of servers for the client to visit, a client which will run an Android emulator and record traces, and a malware

detector engine which will analyze the traces and detect if an attack has occurred. It will also interact with the data collection infrastructure so as to save the attack traces.

As for the data collector, it aims to collect/provide the data from/to the different entities. The data will be a collection of attack traces detected by the lightweight malware, the honeyclient, etc. This infrastructure will expose an interface so that all the entities can interact with it. It will be scalable and therefore will be able to process a highly number of requests for accessing and storing data. It will also enrich the raw data that has been stored (e.g., OS fingerprint, GeoIP, etc.).

5.2 Honeyclient

The honeyclient consists of a static and dynamic analyzer. The static analyzer is provided by Androguard which will extract the characteristics of an Android application whilst the dynamic analysis is done by DroidBox. Both analyzers will work jointly in order to extract as many Android characteristics as possible.

Androguard [22] provides a framework for analyzing statically Android applications. It can retrieve the Android permissions requested by an Android application, extract the list of called functions, re-assemble an application, detect the presence of ads and obfuscators making the analysis harder, etc.

Droidbox [23] automates the analysis of an Android application in the Android emulator [24]. When executed, the Android application calls Android functions that are provided by the Android framework. A subset of those functions are hooked by DroidBox and outputs a log when executed. This subset encompasses the functions that are sending out data, reading from, or writing into files. The output log is retrieved by means of the tool Android Debug Bridge (ADB) [25] and then parsed in order to extract the logs generated by DroidBox. Droibox also includes the module TaintDroid [26] for tracking the information leaks. In other words, every sensitive information that is sent out (e.g., a phone number in a SMS [4]) is logged.

For hooking the functions, DroidBox w.r.t to its version provides two ways: the former one consists in modifying the Android firmware, i.e., modifying and recompiling the Android firmware while the latter one consists in modifying the Android applications in order to add the hooks inside this application. However like malware for Desktop PCs, Android malware can use techniques for hiding the functions and load them dynamically [27] and therefore bypass the Droidbox hooking. As for the second one, a malware can bypass the hooking at the Dalvik level by instantiating a C program which interacts directly with the native functions. However, for calling most of the native functions, the user must have the root access which is not set by default when an application is launched. As for the applications that use elevation of privileges, they represent a minority and generally exploit vulnerabilities of old Android platforms. Therefore, in order to limit the circumventions, we chose the latter way with a new Android emulator

[4] Short Message Service

Ice Cream Sandwich (ICS). Nevertheless, in order to capture most information about the network traffic, the traffic will be captured outside the emulator.

In order to get an analysis as complete as possible of the malware, the honeyclient is instrumented by a framework integrating a virtual user simulator and a user actions recorder [28]. The simulator is built from a model which is based on eight different reproducible scenarios composed of about five user actions each. An action is a click on a button, a swipe, etc. We chose the application Facebook, Hotmail, Youtube, Calendar, Gallery ICS, Browser, Slide Box Puz, and talk.to since they encompass most of the user actions and are amongst the most downloaded. All those applications were running in the Android ICS emulator. Before recording the user actions, the Android has been set up, i.e., an account has been created for the applications Facebook, Hotmail, and talk.to.

The scenarios were adapted for each application in the respective order Facebook, Hotmail, Youtube, Calendar, Gallery ICS, Browser, Slide Box Puz, and talk.to:

- The user (1) signs in, (2) posts a new message, (3) goes to the list of messages by clicking on the top icon Messages, and then (4) opens the first message.
- The user (1) signs in, (2) clicks on the first message, (3) scrolls down, and then (4) presses the button "Home" to go back to the Android board.
- The user (1) scrolls down the list of videos, (2) plays one video, (3) increases the volume, (4) decreases the volume, (5) goes back to the list of videos by clicking on the button "Back", (6) scrolls up the list, and then (7) plays another video.
- The user (1)signs in (2) selects one day with a long pressing, (3) types the name of an event, (4) clicks on Save, and then (5) goes back to the Android board by clicking on the button "Home". (6) signs out.
- The user (1) selects one album, (2) selects one photo, (3) zooms in, and then (4) zooms out.
- The user (1) selects main menu in Android, (2) clicks on the browser icon, (3) selects keyboard, and then (4) types "www.peugeot.com".
- The user (1) moves the ball left, right, up, and down for the 1st level.
- The user (1) selects a contact, (2) types a message, (3) presses the button Settings, and then (4) presses the item End chat.

Since future Android malware might embed virtual machines detectors by checking the GPS locations or track the accelerometers changes for instance, we will besides the previous scenarios, inject some random events so as to make the phone look more real to the Android applications. For instance, events of the accelerometer or the GPS locations will be injected into the Android emulator.

6 Related works

To tackle the sky-rocketing number of malware these last years, Google has set up a service called Bouncer [29] for scanning and detecting the potential malware present on Google Play. This service analyzes the uploaded Android

applications both statically and dynamically. With the former analysis, Google detects the malware by their signature while with the latter analysis, it detects the malware by their misbehavior. This service also prevents known attackers to submit other Android applications to Google Play. Nevertheless, this service can be easily circumvented as shown by C.Miller et al. [30]. For instance, a malware can mitigate its malicious behavior during the analysis by Google Bouncer.

Google has also introduced the notion of permissions for mobile devices in order to inform the user about the intentions of the program. When the application wants to access a resource of the system (e.g. network), it must ask the authorization of the system, i.e., mandatory access control policy. This mechanism prevents malware from using resources whose access rights are prohibited. The rights are granted by the user upon installing the application. However, most of the time the user does not understand those permissions [31].

Even though mobile antiviruses for mobile phone exist, they turn out to be inefficient as shown by Y.Zhou et al. [32]. In their study that was based on $1,200$ malware samples representing all the Android families from 2010 to 2011, they showed that the best antivirus engine detects at most 79.6% of Android malware while the worst case detects only 20.2%. This also reveals that methods to circumvent detections for Android malware evolve rapidly.

7 Conclusion

This paper states the state of the art in Android malware and describes the current development of the NEMESYS infrastructure aiming to understand and detect attacks both at the network and smartphone level. First, we have surveyed the different attack vectors in Android environment. Secondly, we have proposed a taxonomy of Android malware. From the analysis of 38 malware, we have found that the social engineering is the main attack vector for infecting smartphone and that most malware were bots, infostealers, and SMS senders. Thirdly, we have overviewed the overall architecture of the NEMESYS infrastructure, with an emphasis on the components used for detecting Android malware, namely the honeyclient, the data collector, and the lightweight malware detector. Finally, we have shown the current developement of the honeyclient which is composed of both a static and dynamic analyzer in order to extract as many Android characteristics as possible. The honeyclient is instrumented by a virtual user in order to have an Android application analysis as complete as possible. In the future, we are planning to extract the characteristics of the malware and goodware datasets by the honeyclient, and continue the development of the framework, i.e., building the lightweight malware detector in the smartphone, the malware detector and the queuer in the honeyclient, and the data collector.

Acknowledgements

The work presented in this paper is funded by the European Commission FP7 collaborative research project NEMESYS (Enhanced Network Security for Seam-

less Service Provisioning in the Smart Mobile Ecosystem), no. 317888 within the Trustworthy ICT domain.

References

1. Global smartphone installed base forecast by operating system for 88 countries: 2007 to 2017 (2012). URL https://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer\&a0=7834
2. Android under siege: Popularity comes at a price (2012). URL http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt-3q-2012-security-roundup-android-under-siege-popularity-comes-at-a-price.pdf
3. Mobile threat report q4 2012 (2012). URL http://www.f-secure.com/static/doc/labs_global/Research/MobileThreatReportQ42012.pdf
4. Mcafee threats report: Third quarter 2012 (2012). URL http://www.mcafee.com/it/resources/reports/rp-quarterly-threat-q3-2012.pdf
5. The nemesys project (2012). URL http://www.nemesys-project.eu/nemesys/index.html
6. Google play (2013). URL https://play.google.com/store
7. Amazon appstore for android (2013). URL http://www.amazon.com/mobile-apps/b?ie=UTF8\&node=2350149011
8. Getjar (2013). URL http://www.getjar.com/
9. Cyanogenmod (2013). URL http://www.cyanogenmod.org/
10. D. Bornstein. Dalvik virtual machine internals (2008). URL http://de.youtube.com/watch?v=ptjedOZEXPM
11. Netcraft. Angry birds impersonated to distribute malware (2013). URL http://news.netcraft.com/archives/2013/04/12/angry-birds-impersonated-to-distribute-malware.html
12. C. Karlof, U. Shankar, J.D. Tygar, D. Wagner, in *Proceedings of the 14th ACM conference on Computer and communications security* (ACM, New York, NY, USA, 2007), CCS '07, pp. 58–71. DOI 10.1145/1315245.1315254. URL http://doi.acm.org/10.1145/1315245.1315254
13. The samsung exynos kernel exploit - what you need to know (2012). URL http://www.androidcentral.com/samsung-exynos-kernel-exploit-what-you-need-know
14. Black hat hacker lays waste to android and meego using nfc exploits (2012). URL http://www.extremetech.com/computing/133501-black-hat-hacker-lays-waste-to-android-and-meego-using-nfc-exploits
15. R. Naraine. Exploit beamed via nfc to hack samsung galaxy s3 (android 4.0.4) (2012). URL http://www.zdnet.com/exploit-beamed-via-nfc-to-hack-samsung-galaxy-s3-android-4-0-4-7000004510/
16. Google fixes sms crashing bug in mobile os (2009). URL http://www.theregister.co.uk/2009/10/12/google_android_security_update
17. Android smartphones infected via drive-by exploit (2012). URL http://www.h-online.com/security/news/item/Android-smartphones-infected-via-drive-by-exploit-Update-1446992.html
18. Revealed! the top five android malware detected in the wild (2012). URL http://nakedsecurity.sophos.com/2012/06/14/top-five-android-malware/
19. Caro naming scheme (2013). URL http://www.caro.org/naming/scheme.html

20. Cme common malware enumeration (2013). URL http://cme.mitre.org/about/faqs.html
21. Maec malware attribute enumeration and characterizacion (2013). URL http://maec.mitre.org/
22. Androguard (2013). URL https://code.google.com/p/androguard/
23. Droibox (2013). URL https://code.google.com/p/droidbox/
24. Using the android emulator (2013). URL http://developer.android.com/tools/devices/emulator.html
25. Android debug bridge - android developer documentation (2012). URL http://developer.android.com/tools/help/adb.html
26. W. Enck, P. Gilbert, B.G. Chun, L.P. Cox, J. Jung, P. McDaniel, A.N. Sheth, in *Proceedings of the 9th USENIX conference on Operating systems design and implementation* (USENIX Association, Berkeley, CA, USA, 2010), OSDI'10, pp. 1–6. URL http://dl.acm.org/citation.cfm?id=1924943.1924971
27. H. Tamada, M. Nakamura, A. Monden, K.i. Matsumoto, in *Proceedings of the IASTED International Conference on Software Engineering* (ACTA Press, Anaheim, CA, USA, 2008), SE '08, pp. 125–130. URL http://dl.acm.org/citation.cfm?id=1722603.1722627
28. P. Moschonas, N. Kaklanis, D. Tzovaras, in *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry* (ACM, New York, NY, USA, 2011), VRCAI '11, pp. 31–40. DOI 10.1145/2087756.2087760. URL http://doi.acm.org/10.1145/2087756.2087760
29. Android and security (2012). URL http://googlemobile.blogspot.com.es/2012/02/android-and-security.html
30. Circumventing google's bouncer, android's anti-malware system (2012). URL http://www.extremetech.com/computing/130424-circumventing-googles-bouncer-androids-anti-malware-system
31. P.G. Kelley, S. Consolvo, L.F. Cranor, J. Jung, N. Sadeh, D. Wetherall, in *Proceedings of the 16th international conference on Financial Cryptography and Data Security* (Springer-Verlag, Berlin, Heidelberg, 2012), FC'12, pp. 68–79. DOI 10.1007/978-3-642-34638-5_6. URL http://dx.doi.org/10.1007/978-3-642-34638-5_6
32. Y. Zhou, X. Jiang, in *Proceedings of the 2012 IEEE Symposium on Security and Privacy* (IEEE Computer Society, Washington, DC, USA, 2012), SP '12, pp. 95–109. DOI 10.1109/SP.2012.16. URL http://dx.doi.org/10.1109/SP.2012.16