International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-11)

Research Article | **November 2015**

Interpretation & Evolution of Malware Apps in ANDROID Platform

[1]Pooja Singh, [2]Pankaj Tiwari, [3]Dr. Santosh Singh
[1] Computer Science, AMET University, India
[2] Computer Science, JJT University, India
[3] Computer Science, University of Mumbai, India

Abstract –

The increase in the popularity of smartphones there is simultaneous increase in mobile malware especially on the platforms such as Android, IOS etc. Due to this rapid growth there is need of developing effective solutions to this malwares. However due to rapid growth and introduction of new malware daily in market it is very difficult to propose any solution and understanding of the same. In this paper we discuss current Android Security issues and intend to characterize the existing Android malware. Researchers and antimalware companies have analyzed that traditional signature based and static analysis methods are not secure. We believe that the security of android with focus on malware growth, study of analysis technique and existing detection methods needs and extensive coverage. We focus on Android security enforcement methods ,threats to the existing system and malware growth between 2010 to 2015.This review gives an insight into the strengths and shortcomings of the known research methodologies and provides a platform to the researchers and beginners for proposing the next generation Android security and analysis and detection of malware

Keywords— Android Security, Android Malware,

I. INTRODUCTION

A. Mobile Application Security vs Web and Desktop Applications

Personal computer and laptop safety often referred to as "desktop security", iabsolutely essential to ensuring a robust, reliable, and secure environment. But with the entry of smartphones in evolving and fast growing market of mobile devices in the past few years it has been a challenge to secure it too. The obvious question which arises is –
"Does mobile application security is different than web or desktop applications?"
On top of the common threats on desktops, Laptops and servers; mobile devices deal with a wider risk, being exposed to even more security threats like –

- Mobile Devices tend to be misplaced, lost or even stolen - An attacker might need not more than a few minutes of physical access to a smartphones in order to extract information or perform activities on behalf of its original holder.
- Mobile malware can steal personal information, can send SMS on user's behalf, and can access private photos and post in user's name.
- Keeping personal data or more secure requires addressing concerns like encryption/ decryption, validation, authorization and securing communication with the server side.

Unfortunately, recent research has proven that there is a long way to go.

B. Smartphone and Malware Relation

Smartphones have become explosively dominant for private or commercial use in recent years. The worldwide smartphone market grew 13.0% year over year in second quarter of 2015, with 341.5 million deliveries, according to data from the International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker.
These smartphones run applications that have exceptional access to private information, including contacts, emails, location info, private and commercial files, and much more. There is also obvious monetary risk associated with these smartphones since phone calls, SMS, and data usage can cost money. More precisely, these smartphones often have access to user's bank accounts over an application or as a means to validate to a bank, and in the future it seems likely that phones may act as a digital wallet, openly gaining the access to bank accounts as part of the functionality. While the existence of this information and access generates much of the value originate in a smart mobile device, it also makes these devices easy targets for malicious entities.
The prototype for program distribution on these smartphones also differs from that of the customary PCs. Developers those are developing the apps are frequently releasing it to one or few central application markets or app stores. Although there are third party application stores available but currently all popular mobile device -platforms have central application stores as the primary mechanism of application delivery. Android has Google Play as the primary store,

Kindle uses the Amazon App store for Android, iOS has iTunes App Store, Windows RT has the Windows Store, and BlackBerry has App World. This new paradigm offers both challenges and opportunities for malware defense. As an alternative most programs coming from a quite small number of trustworthy vendors, which assists protection based on whitelisting and signed software deliveries. In mobile devices there are many developers, which have insufficient history to establish reputation. On the other hand, centralized markets provide chances for techniques to analyze applications by extracting some set of quantifiable features, and identifying possibly malicious applications in the set.
Android security researchers and agencies reports an worrying rise of malware from just three malware families with 100 samples in 2010 to over 2 million new Android malware strains is realistic.

- The global market stake of Android smartphones and tablets used only for Internet access surpassed 61 percent in the first quarter of 2015. Nearly 60.85 percent of users globally used a mobile device with an Android operating system to go online.
- Decisive malware numbers for Android devices: G DATA security specialists recognized and analysed 440,267 new malware samples in the first quarter of 2015. This signifies an increase of 6.4 percent compared to the fourth quarter of 2014 i.e. 413,871. The number of malware worries rose by 21 percent compared to the first quarter of 2014 i.e. 363,153.
- Financially inspired Android malware makes up about half of the malware analysed (50.3 percent). This type of malware includes banking Trojans, ransomware, SMS Trojans etc.
- Many apps, particularly free ones, depend on the display of advertising. But at what stage should such applications be well-thought-out adware? When apps have a capabilities to cover themselves in the background, obscure functions from the user or feed genuine apps with supplementary advertising networks, the experts categorise them as PUPs i.e. potentially unwanted program.

II. PREDICTIONS AND DRIFTS IN RECENT MALWARE

A. Significant Number Of New Malware Samples Increases Significantly

The year 2014 has seen an exponential progress in the figures of Android malware that have been discovered and reported. In the three years since 2011, the research lab of Quick Heal Threat has countersigned a 304 times growth in Android malware. This rise in the malware is directly accredited to the high rise in Android smartphone sales and usage across the globe. These expanding drifts also notice that the authority of mobile malware attacks will sustain in 2015 as well.

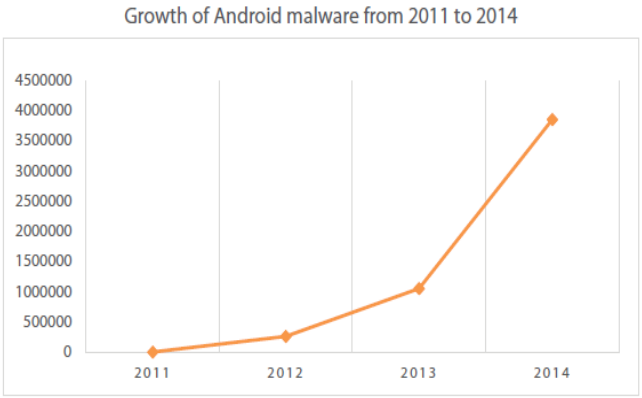

Fig.1

B. Mobile Application Security State

The outcomes of this research are no less than alarming. The average number of vulnerabilities per application based investigation of hundreds of applications of all types, stands at 9.041 vulnerabilities per application. This number consists of all severity levels.
The table below presents the distribution by severity:

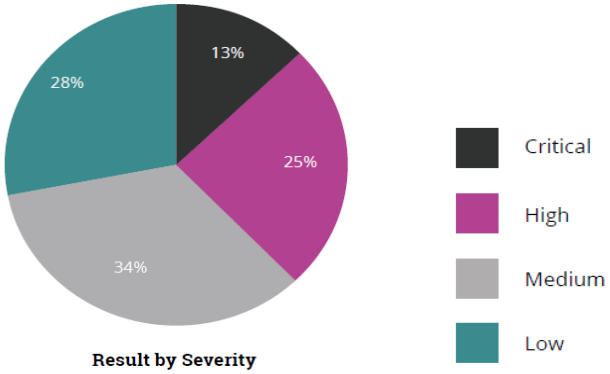

Fig. 2

C. Reported Android Malware Threat Observation

Below Fig.3 shows the time-line of some distinguished malware families of Android during 2010–2013. Among them, SMS Trojans have key contribution; some of these have infected the Google Playstore [50]. A huge number of malware apps have exploited root-based attacks such as rage-against-thecage, gingerbreak [5] and z4root [45] to gain super user honors to control the device. The most latest Android exploit is the master-key attack [45], which has the versions starting from 1.6 to 4.2.2 vulnerable.

The present Android threat observation and malware infection rate has a huge reported deviation between the commercial anti-malware and autonomous studies.

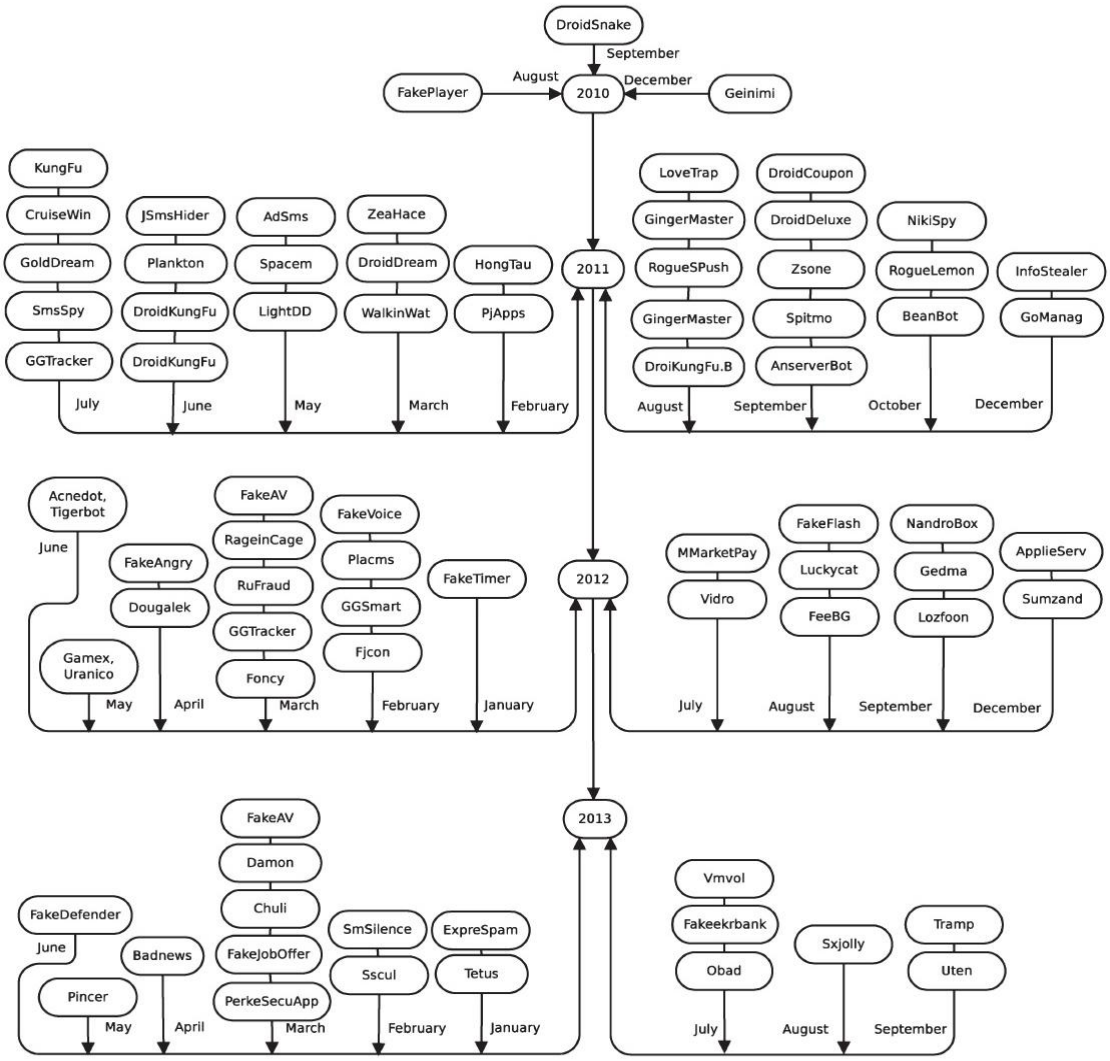

Fig. 3. Android Malware Family Chronology

Top 10 Android Malware Variants discovered in 2014

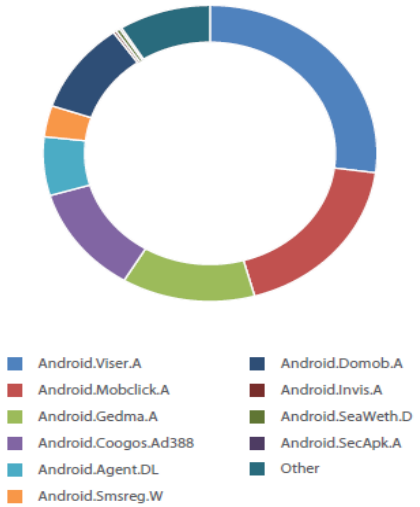

Fig. 3

D. Reported Android Malware Threat Perception

1. Trojan

Trojans masquerade as benign apps, but they perform harmful activities without consent or knowledge of the users. Trojans disclose the secret user information, or they may "phish" the user and steal the sensitive information such as passwords. Till the end of 2012, majority of the android variants belonged to various SMS Trojan families. SMS Trojan apps are able o to send SMS to best rate numbers without the information and/or permission of the user incurring monetary loss to the owner. Apart from that, such Trojans also reveal contacts, messages, IMEI/IMSI numbers. FakeNetflix [65] pretend itself as popular Netflix app, phishing the user to enter their login details. Fakeplayer [2], Zsone [3] and Android. Foney[4] are a few famous Android Trojans incurring financial loss to the user.

2. Backdoor

Backdoor helps r malware to silently enter the system facilitating them to void normal security procedures. Backdoor can use root exploits to increase the super user advantage and hide from the anti-malware scanners. A number of root exploits such as rage-against-the-cage and gingerbreak [5] get full-control of the device. Basebridge [1], KMin [1], Obad [22] are important example of the backdoors.

3. Worm

Worm app can create exact or similar copies of itself and spreads them through network and/or removable media.For example, Bluetooth worms can exploit bluetooth functionality and send copies to the paired devices. Android-Obad.OS [22] is well known Bluetooth worm.

4. Botnet

Botnet apps cooperate with the device to create a Bot, so that the device is managed by a remote server, called Bot-master, through a sequence of commands. Network of such bots is called a Botnet. By sending simple commands private information to remote-server can be accessed or as complex leading to denial of service attacks. Bot also contain commands to download infected payloads involuntarily. Geinimi [1], Anserverbot [1], Beanbot [1] are important Android botnets.

5. Spyware

Spyware may present itself as a good utility, but has a hidden agenda to surreptitiously monitor contacts, messages, location, bank mTANs etc. that leads to undesirable consequences. It can send the collected information to the remote server. Nickyspy [50].

6. Aggressive Adware

Android provides common and very well grained location services. Some advertisement associate networks exploitation such location services and send modified advertisements to the user device to generate monetary benefit. Aggressive adware can create shortcuts on the home-screen, take bookmarks, and change the default search engine settings and approaching unnecessary notifications hinder the efficient device usage. Plankton [3] is a known aggressive adware.

7. Ransomware

Ransomware is a lock through which the user device is made unapproachable until some ransom amount is paid through online payment service. For example, FakeDefender.B [68] masquerades as avast! [69] anti-malware and displays the fake malware alerts to persuade the user to install this trick malware. In addition, it locks the device and stress ransom to unlock the device.

III. MALWARE PENETRATION AND SURVIVAL TECHNIQUES

In this section we explore the android malware penetration technique and proposed solutions

A. Disassembling the popular apps:

It is the process of decompiling the popular free or paid apps from the popular market places, inserting malware code, appending the malware payload .reassemble the Trojan infect ted applications and distribute them in less monitored applications stores. The apps can be repacked using reverse engineering tools. Reverse engineering is illustrated in fig.
The following steps explain the process of dissembling:

- Download the apps from app store
- Disassemble them using tools such as apktool[70]
- Generate the malicious payload in dalvik bytecode or java using dextojar[71] tool
- Add the malware payload into benign app. Modify the AndroidMenifest.xml
- Assemble modified source code again using apktool
- Distribute the app in third party app market which is less monitored

Repacking is the most basic malware app generation technique.

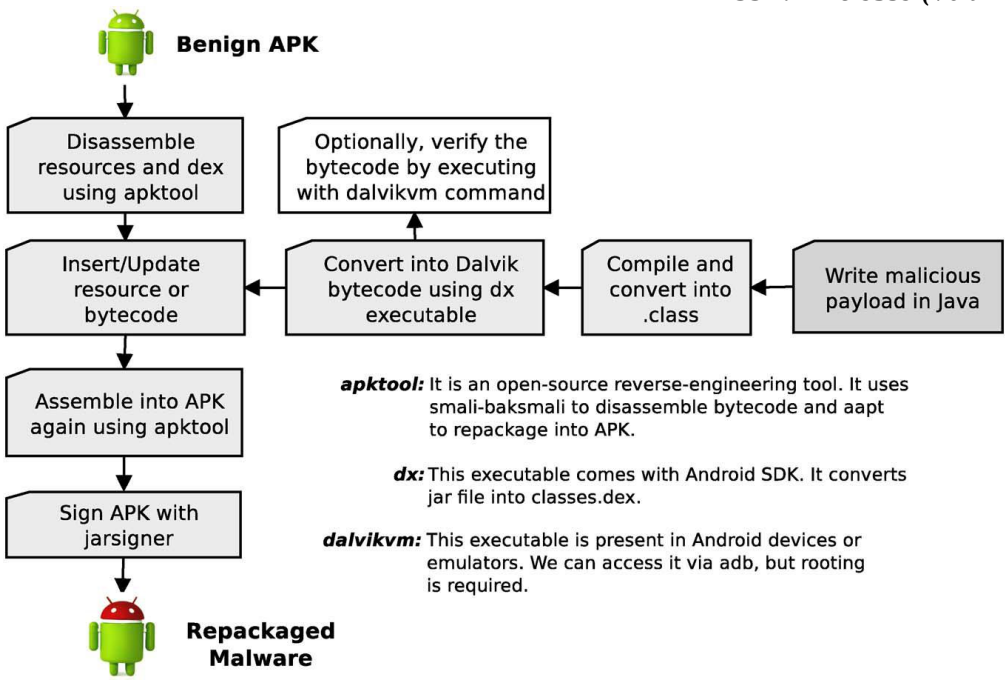

Fig. 4. App Repackaging Process.

B. Drive-by Download

An attacker can employ social engineering, aggressive advertisements and click a malicious URL, provoke user to download malware involuntarily. Optionally, a drive by download may mask a valid application and persuade the user to install an app. *Android/NotCompatible* [25] is a notable drive-by download app.

C. Dynamic Payload

An app can also be inserted in malicious payload as an executable apk/jar in encrypted or plain form within the APK resources. Once it is installed, the app decrypts the payload. If the payload is a jar file, malware loads DexClassLoader API and execute dynamic code. However, it can wheedle the user to install the embedded apk by disguising as an important update. The app can execute native binaries using Runtime. Exec API, an equivalent of Linux fork()/exec(). *BaseBridge* [50] and *Anserverbot* [50] malware families . Some malware families does not insert malicious payload as a resource, but rather download them from the remote server and successfully escape detection. *DroidKung- FuUpdate* [50] is a distinguished example of executing payload. Such techniques go hidden with static analysis methods.

D. Stealth Malware Techniques

Android OS is developed for resource controlled environment keeping in mind the availability of limited battery availability of the underlying smartphone. On device anti-malware apps cannot perform the real-time analysis as compared to their desktop equivalent. Malware authors exploit these hardware constraints preventive the anti-malware and complicate the malicious payloads to spoil the commercial anti-malware.

.

IV. APPROACHES FOR MALWARE DETECTION

Malware analysis techniques are divided into: 1) Static; 2) Dynamic and 3) Hybrid. Static analysis methods analyze code without actually running it, hence they are quick, but they have to deal with false-positives. Dynamic analysis techniques monitor the executed code and inspect its interaction with the system. Though time-consuming they are effective against malware obfuscation. Hybrid approaches leverage the good of both the static and dynamic analysis method

A. Static Approach*:* Static analysis based approaches work by just disassembly; de-compilation without actually running it, hence does not infect the device. This approach is undermined by the use of various code transformation techniques.

B. Dynamic Approach*:* Static analysis and detection approaches are quick, they fail against the encrypted, polymorphic and code transformed malware. Dynamic analysis methods execute the app in a protected environment, providing all the emulated resources it needs, thereby learning its interaction identify malicious activities. Some dynamic analysis methods have been implemented, but the resource constraints of a smartphone limit such execution methods.

V. TOOLS & TECHNIQUES FOR ANDROID APP MALWARE DETECTION

A. Reverse-Engineering Tools

Content of Android package (APK) is stored in the binary format. Before assessment, analysis or detection task initiates, it is important to disassemble it for further processing. There are a number of tools to disassemble and/or decompile the Android app. In the following section, we discuss some known reverse engineering tools considering their strengths.

1) apktool [9] can decode binary content of an APK into nearly original form in project-like directory structure. It disassembles the binary resources and converts bytecode within classes.dex into the smali [114] bytecode for easier reading and manipulation. After making the changes, it can also repackage it back into an APK. This tool is one of the best open source reverse-engineering tool.

2) dex2jar [12] is a disassemble to parse both the .dex and optimized dex file, providing a light-weight API to access it. *dex2jar* can also convert dex to a jar file, by re-targeting the Dalvik bytecode into Java bytecode, for further manipulation. Moreover, it can also re-assemble the jar into a .dex after the modifications.

3) Dare [13] project aims at re-targeting Dalvik bytecode within classes.dex to traditional .class files using strong type inference algorithm. This .class files can be further analyzed using a range of traditional techniques developed for Java applications, including the decompilers. Octeau *et al.* [101] demonstrated that *Dare* is 40% more accurate than *dex2jar*.

4) Dedexer [26] disassembles the classes.dex into Jasmin-like syntax and creates a separate file for each class maintaining the package directory structure for easy reading and manipulation. However, unlike the *apktool*, it cannot re-assemble the dis-assembled intermediate classfiles.

5) JEB [27] is a leading professional Android reverseengineering software available on Windows, Linux and Macintosh platforms. It is a GUI-based interactive decompiler to analyze the reversed malware app content. App information such as manifest, resources, certificates, literal strings can be examined in Java source by providing an easy navigation through the cross-references. JEB converts the Dalvik bytecode directly into Java source by utilizing dalvik bytecode semantics. Exceptionally, JEB
can also de-obfuscate Dalvik bytecode to make disassembled code more readable in comparison to its counterparts [70]. JEB supports Python scripts or plugins by allowing access to the decompiled Java code Abstract Syntax Tree (AST) through API. This feature is helpful in automating the custom analysis. According to us, it is the best reverse-engineering tool so far.

B. AndroSimilar

In [83] authors proposed AndroSimilar, an automatic signature generation approach that extracts statistically rare syntactic features for malware detection. Apart from existing malware, AndroSimilar is able to reasonably detect fuscated malware with techniques like string encryption, method renaming, and junk method insertion and changing control flow, idely used to evade fixed anti-malware signature, thus it can detect unknown variants of existing malware. AndroSimilar approach is based on Similarity Digest Hash (SDHash) [17] used in digital forensics to identify similar documents. Intuitively, completely unrelated apps should have lowerprobability of having common features. When two unrelated apps share some features, such features should be considered weak as using these shall lead to false positives [18]. Fixedsize byte-sequence features are extracted based on empirical probability of occurrence of their entropy values, then popular features are searched among them according to rarity in neighborhood [18].

C. Andrubis

Andrubis [19] is a web-based malware analysis platform, built upon some well-known existing tools Droidbox [20], TaintDroid [24], apktool [9] and Androguard [25]. Users can submit suspicious apps through the web based interface. After analyzing the app at the remote-server, Andrubis returns detailed static and dynamic analysis reports as a web page. Andrubis also provides app behavior rating between 0–10, where 0 indicates benign and 10 specifies malicious rating. To study the Andrubis functionality, a custom SMS based botnet was uploaded on the Andrubis web service. This research prototype rated custom SMS bot with a score 9.9/10. However, none of the commercial anti-malware on the virustotal portal were able to detect this unseen malware. This demonstrates the effectiveness of Andrubis behavior rating against the zero day malware. However, Vidas *et al.* [106] demonstrated that Andrubis virtual environment is detected with anti-analysis techniques and identified the analysis sandbox.

D. APKInspector

- APKInspector [21] is a full-fledged Android static analysis tool, consisting *Ded*, *smali/baksmali* [11], *apktool* [9]
- and *Androguard* [25]. It provides a rich GUI and has following features:
- App meta-data
- Analysis of sensitive permissions
- Displays Dalvik bytecode and Java source code
- Displays control-flow graph
- Displays call-graph, displaying call-in and call-out structures
- Static instrumentation support by allowing modification to the *smali* code

E. Bouncer

Google protects its own app-store, Google Play, with a system called Bouncer. It is a virtual machine based dynamic analysis platform to test the uploaded third party developer apps, before availing them to the users for download. It executes app to look for any malicious behavior and also compares it against previously analyzed malicious apps. Though no documentation of internal functioning is available, Oberheide *et al.*[39] presented their analysis of Bouncer environment by implementing a custom command and control app. Dynamic codeloading techniques can evade the Bouncer [22] scrutiny.

VI. CONCLUSION AND DISCUSSION

In this article we have surveyed Android security issues. We have given some statistics to analyze the alarming threat. We have also compared and reviewed most well-known security solutions for Android platform. Our survey additionally shows the advantages and limitations of these solutions. We also have proposed current directions for research in this area. Besides, we have developed a list of requirements for "prospective" full-system analysis solution for Android platform, as well as the architecture of the solution.

REFERENCES

[1] Z. Yajin and J. Xuxian, "Dissecting android malware: Characterization and evolution," in *Proc. 33rd IEEE Symp. Security Privacy*, Oakland, CA, USA, 2012, pp. 95–109
[2] G. Andre and P. Ramos, "Boxer SMS Trojan," ESET Latin American Lab, Bratislava, Slovakia, Tech. Rep., 2013.
[3] C. A. Castillo, "Android malware past, present, future," Mobile Working Security Group McAfee, Santa Clara, CA, USA, Tech. Rep.,2012.
[4] F. Shahzad, M. A. Akbar, and M. Farooq, "A survey on recent advances in malicious applications analysis and detection techniques for smartphones," National Univ. Comput. Emerging Sci., Islamabad, Pakistan
[5] GingerBreak, (Online; Last Accessed Feb. 11). [Online]. Available: http://forum.xda-developers.com/showthread.php?t=1044765
[6] Backdoor.AndroidOS.Obad.a, (Online; Last Accesed Dec. 25, 2013). [Online]. Available: http://contagiominidump.blogspot.in/2013/06/ backdoorandroidosobada.html
[7] C. A. Castillo, "Android malware past, present, future," Mobile Working Security Group McAfee, Santa Clara, CA, USA, Tech. Rep., 2012.
[8] Fakedefender.B—Android Fake Antivirus, (Online; Last Accesed Dec. 25, 2013). [Online]. Available:. in/2013/11/fakedefenderb-android-fake-antivirus.html
[9] APKTool, Reverse Engineering with ApkTool, (Online; Accessed Mar. 20, 2013). [Online]. Available: https://code.google.com/android/apk-tool Inc., Class to Dex Conversion with Dx, (Online; Last Accessed Mar. 5, 2013). [Online]. Available: http://developer.android.com/tools/help/index.html
[10] Android/NotCompatible Looks Like Piece of PC Botnet, (Online; Last Accesed Dec. 25, 2013). [Online]. Available: http://blogs.mcafee.com/mcafee-labs/androidnotcompatible-looks-like-piece-of-pc-botnet
[11] BakSmali, Reverse Engineering with Smali/Baksmali, (Online; Accessed Mar. 20, 2013). [Online]. Available: https://code.google.com/smali
[12] Dex2Jar, Android Decompiling with Dex2jar, (Online; Last Accessed May 15, 2013). [Online]. Available: http://code.google.com/p/dex2jar/
[13] DARE: Dalvik Retargeting, (Online; Last Accessed Feb. 11, 2013).[Online]. Available: http://siis.cse.psu.edu/dare/
[14] Dedexer, (Online; Last Accessed Feb. 11, 2013). [Online]. http://dedexer.sourceforge.net/[117]
[15] JEB Decompiler, (Online; Last Accessed Feb. 11, 2013). [Online].Available: http://www.android-decompiler.com/
[16] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, "AndroSimilar:Robust statistical feature signature for Android malware detection," in *Proc. SIN*, A. Eli, M. S. Gaur, M. A. Orgun, and O. B. Makarevich, Eds., 2013, pp. 152–159. [Online]. Available: http://dblp.uni-trier.de/db/conf/sin/sin2013.html#FarukiGLGB13
[17] V. Roussev, "Data fingerprinting with similarity hashes, advances in digital forensics," in *Proc. Int. Conf. Digit. Forensics*, 2010, pp. 207–226.
[18] V. Roussev, "Building a better similarity trap with statistically improbable features," in *Proc. 42nd HICSS*, 2009, pp. 1–10.
[19] Andrubis, 2012. [Online]. Available: http://anubis.iseclab.org/ [122] A. Desnos and P. Lantz,
[20] "Droidbox: An android application sandbox for dynamic analysis, 2011. [Online]. Available: https://code.google.com/p/ droidbox/
[21] APKInspector, 2013. [Online]. Available: https://github.com/honeynet/apkinspector.
[22] Google Bouncer: Bad guys may have an app for that, Feb. 2012.[Online]. Available: http://www.techrepublic.com/blog/it-security/google-bouncer-badguys-may-have-an-app-for-that/7422/
[23] Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid:Behavior-based malware detection system for Android," in *Proc. 1st ACM Workshop Security Privacy Smartphones Mobile Devices*, 2011,pp. 15–26.
[24] E. William, G. Peter, C. Byunggon, and C. Landon, "TaintDroid: An information flow tracking system for realtime privacy onitoring on smartphones," in *Proc. USENIX*, 2011.
[25] BlackHat, Reverse Engineering with Androguard, (Online; Accessed Mar. 29, 2013). [Online]. Available: https://code.google.com/androguard.
[26] Dedexer, (Online; Last Accessed Feb. 11, 2013). [Online]. http://dedexer.sourceforge.net/
[27] JEB Decompiler, (Online; Last Accessed Feb. 11, 2013). [Online].Available: http://www.android-decompiler.com/
[28] *http://www.idc.com/prodserv/smartphone-market-share.jsp.*

