Digital Investigation xxx (2017) 1 11

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

A methodology for the security evaluation within third-party Android Marketplaces

William J. Buchanan [a, *], Simone Chiale [b], Richard Macfarlane [b]

[a] The Cyber Academy, Edinburgh Napier University, United Kingdom
[b] The Cyber Academy, United Kingdom

ARTICLE INFO

Article history:
Received 28 January 2017
Received in revised form
10 July 2017
Accepted 10 October 2017
Available online xxx

Keywords:
Malware analysis
APK
Android
Marketplace

ABSTRACT

This paper aims to evaluate possible threats with unofficial Android marketplaces, and geo localize the malware distribution over three main regions: China; Europe; and Russia. It provides a comprehensive review of existing academic literature about security in Android focusing especially on malware detection systems and existing malware databases. Through the implementation of a methodology for identification of malicious applications it has been collected data revealing a 5% of them as malicious in an overall analysis. Furthermore, the analysis shown that Russia and Europe have a preponderance of generic detections and adware, while China is found to be targeted mainly by riskware and malware.

Introduction

In a context where Android is spread worldwide as the most used OS, even if it is still a young one, this paper aims to help to evaluate the real growth of threats along with it. In 2012 a good deal of research was performed to try to study an increasing amount of malware in Android. Zhou et al. categorized and analysed a large number of malicious applications downloaded from different sources in order to evaluate risk evolution since Android was born (Zhou, 2012).

Currently little research is focused on mobile security outside of the main Google marketplace and there is a lot of disinformation in this field among average users. In recent years there has been a large increase in the number of unofficial third party marketplaces, both in number and variety, due to the large demand placed by users. Along with them, the number of fake and malicious apps also increased. As a relevant study revealed in 2012, between 5% and 13% of a sample number of apps downloaded through third party marketplaces were modified from the original version via repackaging techniques (Zhou et al., 2012).

With the fast growth of network technology, mobile devices and mobile internet are becoming a trend for the information ages. Since 2011 smartphones surpassed traditional desktop computers in terms of units sold and the smartphone market grew by 13% in 2015 alone. It should be noted, however, that 82,8% of the total is made by Android devices with only a 13% of IOS and a circa 5% of other Operating Systems (IDC, 2015).

Android is an Operating System for mobile devices based on the Linux Kernel and is currently developed by Google. It is aimed mainly at touch screen systems (smartphones and tablets) and it has the characteristics of an almost totally *Free and Open Source Software*, apart from some proprietary drivers included by device's manufacturers.

Apache 2.0 license permits to modify and redistribute the source code and thanks to that a vast community, made up of developers and supporters, has grown up around it to become the most popular OS in mobile environment, in addition to the fact that it is best choice for mobile manufacturers thanks to low production costs, high customization and lightness of the platform itself.

A growing distribution of smartphones and of the Android OS itself has brought an entirely new world of threats and malicious activities linked with this relatively new and young system. Moreover, accessing the so called "black market" it is possible to share malware and exploits easily with other users and the only trend that seems to contrast it, is to pay customers and developers to advise unknown bugs and vulnerabilities instead of spreading them on the web. Even if specific programs have been developed to use *fuzzing* techniques to find or prevent big bugs to be spread, it is still very

* Corresponding author.
E-mail address: w.buchanan@napier.ac.uk (W.J. Buchanan).

https://doi.org/10.1016/j.diin.2017.10.002
1742-2876/

likely that a large number of exploits can be detected each year in Android OS as shown in the last CVE survey for 2015 (CVE, 2015).

While the early days of Android mobile malware were just a proof of concept, after many years of studies in mobile phone security, specific categories have been created based mainly on malware behaviour and targets. Even if rooting techniques (used to get access to the file system) are becoming more sophisticated, developers take advantage of web communities, which publish root exploits to increase smartphone customization. Using them, a malware developer can often root the target device and execute malicious code on it without user permission (Felt et al., 2011).

The increasing amount of vulnerabilities discovered weekly, along with the considerable number of new variants that are being revealed recently, are setting up a dangerous scenario which risks to transform Android in a totally unsecure platform. Those potentially alarming situations are well represented in another survey performed by Symantec in 2016, where the last three years are compared in order to show how the increasing amount of Android users is changing its security level (Symantec Corporation, 2016).

Android became popular thanks also to its way of sharing *very easy to code* applications. It uses the "Google Play Store" as the official marketplace to download apps but periodically new malware and threats are discovered inside the Play Store despite Google increased security through automated controls like Google Bouncer (Faruki et al., 2015).

To cite an example to help readers to understand actual risks and dangers linked with Android OS, a very recent research made by Trend Micro revealed a new malware named *Godless* which uses mainly publicly available root exploits along with a malicious payload in order to infect as many users as possible. Potentially 90% of all devices worldwide are contaminated with this malware (Trend Micro, 2016). High risk malware are being discovered frequently especially during 2016, when not only third party marketplaces have been found hosting potentially malicious content, but also the Google Bouncer has been often too easily avoided, resulting in many dangerous applications stored in the Google Play Store.

As a proof of how easily new malware are spreading and how much more complex they are becoming in terms of coding, at writing time, a new very powerful malicious APK has been found by Check Point Research Team on the 30/11/2016. They named it *Gooligan* and it is infecting circa 13,000 devices per day (Check Point Research Team, 2016).

Related literature

Market stores

With the growing number of Android applications, Google had to make its own marketplace, called "Google Play Store", as robust and trustable as possible. But, for various reasons, people often need or want to download applications from different sources from the main one and so during the last years Third Party Marketplaces started to grow throughout the internet. A potential reason for customers to prefer other Marketplaces instead of the Play Store is the excessive power that Google has over each Google Account. It has been shown various times that Google has used a backdoor to remotely uninstall applications downloaded from the Play Store, which have been found to be malicious. Many security experts and the "Free Software Foundation" have commented on the actions taken, saying that no one should have such considerable power over its customers (Keizer, 2011).

Google Play Store

Google Play Store is the way Google forces people to download applications through its market. Developers have to upload APKs signed with their own Google key to the marketplace if they want them to be easily accessed by customers. From a user point of view the Play Store provides access to a huge database of applications and their details and statistics (Google Inc, 2016). Unfortunately, currently not much information has been revealed about Google Play Store's backend system, but a recent project developed by Computer Science Department of Columbia University built an application called "PlayDrone" used to crawl inside Play Store so to explore as much as possible about it. Researchers found out that Play Store contains more than 1,100,000 APKs with 25% of them being duplicates of other ones. They also discovered a lot of badly managed applications that could put customer's sensitive data at high risk (Viennot et al., 2015).

One of the main selling points of the Google Play Store is that it is very useful both for customers and publishers. Customers can either search easily through an always updated and well managed application for a specific app or use Google Top rankings charts to see which applications could fit better for them. From a publisher point of view the Play Store guarantees 70% of applications revenue paying just 25$ fee for the developer account once (Google, 2016).

Third party marketplaces

New Android marketplaces regularly appear on the internet, however due to IP and geo blocking restrictions, it can be difficult to categorise them all. In late 2013 the Politecnico di Milano University tried to characterise alternative marketplaces finding 89 distinct webstores with a tool called *AndroCrawl* (Fig. 1). Other studies have been performed in the same field but without a specific tool and with minor results (Sisto, 2013).

It is interesting to note how China in particular is one of the countries with the highest number of marketplaces. The reasons for it resides on the fact that China is the only country in the world where Google Play Store is not permitted as Fig. 2 shows (Google Inc, 2016).

Security

Google Bouncer

As previously explained, the main approach used by Android (and other OS) to protect users is to create a trusted pool from where to safely download new applications, the so called marketplaces. However, trying to make it a safe store seems to be a very hard challenge, especially considering the huge number of new applications uploaded daily. Along with them an impressive number of malicious applications are linked. According to one of the last security reports made by "G Data", 8240 new Android malware threats have been discovered in just one day during last quarter of 2015 (IDC, 2015).

These numbers make it impossible to use manual verification for each application like the Apple Store does so on 2nd February 2012 Google released Google Bouncer, an app validator announced with the following description:

> "The service performs a set of analyses on new applications, applications already in Android Market, and developer accounts. Here's how it works: once an application is uploaded, the service immediately starts analyzing it for known malware, spyware and trojans. It also looks for behaviors that indicate an application might be misbehaving, and compares it against previously analyzed apps to detect possible red flags. We actually run every application on Google's cloud infrastructure and simulate how it will run on an Android device to look for hidden, malicious behaviour. We also analyze new developer accounts to help prevent malicious and repeat offending developers from coming back" (Google Mobile Blog, 2012).

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002

Marketplace	URL	Language
AndroidBlip	http://www.androidblip.com/	English
AppsLib	http://www.appslib.com/	English
F-Droid	http://www.f-droid.org	English
GetJar	http://m.getjar.com/	English
AppMarket	http://indiroid.com/	Turkish
Soc.io	http://soc.io/	English
Andiim3	http://andiim3.com	English
Android	http://androidis.ru	Russian
AndroidPhones.ru	http://android-phones.ru/category/files/	Russian
Anzhi	http://www.anzhi.com	Chinese
Aptoide	http://aptoide.com	English
Amazon AppStore	http://www.amazon.it/mobile-apps/b?node=1661660031	English
Hiapk	http://hiapk.com	Chinese
Opera Mobile Store	http://apps.opera.com/	English
Mikandi	http://mikandi.com	English
Myandroid.su	http://myandroid.su/index.php/catprog	Russian
Wandojia	http://wandoujia.com/	Chinese
Androidpit	http://www.androidpit.com	English
1Mobile	http://www.1mobile.com	English
92apk	http://www.92apk.com	Chinese
Android online	http://www.androidonline.net	Chinese
Appchina.com	http://www.appchina.com	Chinese
Appitalism	http://www.appitalism.com	English
Eomarket.com	http://www.eoemarket.com	Chinese
Insyde Market	http://www.insydemarket.com	English
Nduoa	http://www.nduoa.com	Chinese
SlideMe	http://slideme.org	English
SjApk	http://www.sjapk.com	Chinese
Xda Developers	http://forum.xda-developers.com	English
4PDA	http://4pda.ru/forum/index.php?showforum=281	Russian
Softportal	http://www.softportal.com/dlcategory-1649.html	Russian
AppBrain	http://www.appbrain.com/	English
Apk gfan	http://apk.gfan.com/	Chinese
ProAndroid.net	http://www.proandroid.net/	Russian
Andapponline	https://www.andapponline.com/	English
AppsZoom	http://www.appszoom.com/	English
AndroLib	http://www.androlib.com/	English
Camangi	http://www.camangimarket.com/	English
ESDN	http://www.esdn.ws/mobile-applications-market	English
T-app	http://tapp.ru/	Russian
Jimi168	http://www.jimi168.com/	Chinese
Android MyApp	http://android.myapp.com	Chinese
D.cn	http://android.d.cn/	Chinese
Hami apps	http://hamiapps.emome.net/	Taiwanese
Lenovo	http://3g.lenovomm.com/	Chinese
Mobango	http://www.mobango.com	English
Nexva.com	http://nexva.com	English
Panda app	http://download.pandaapp.com	English
T-store	http://www.tstore.co.kr	Korean
Taobao	http://app.taobao.com	Chinese
Yandex	http://store.yandex.com/	Russian
BrotherSoft	http://android.brothersoft.com/	English
Mobo Market	http://store.moborobo.com	English
AppZil	http://www.appzil.com/	Korean
Freeware Lovers	http://www.freewarelovers.com/	English
Android Downloadz	http://www.androiddownloadz.com/	English
CoolApk	http://www.coolapk.com	Chinese
APK8	http://www.apk8.com/	Chinese
APK	http://apk.imobile.com.cn/	Chinese
Gooyo	http://www.gooyo.com/	Chinese
Aibala	http://www.aibala.com/	Chinese
AppVN	http://appstore.vn/android/	English
AppTomato	http://apptomato.com/	English
Mobogenie	http://www.mobogenie.com/	English
GetBazaar	http://getbazaar.com/en/	Arabic
RepoDroid	http://repodroid.com/	English
GetApk	http://getapk.co/	English
Samsung Galaxy App	http://samsungapps.sina.cn/main/getMain.as	English
189store	http://www.189store.com	Chinese
Baidu Mobile	http://as.baidu.com	Chinese
Sogou	http://app.sogou.com	Chinese
Zhushou 360	http://zhushou.360.cn	Chinese
25PP	http://apps.uc.cn	Chinese
CNMO	http://app.cnmo.com	Chinese
Removed Apps	http://www.removedapps.com	English
Appdh.com	http://www.appdh.com/	Chinese
Apps Apk	http://www.appsapk.com	English
Mobile Apk World	http://mobileapkworld.com	English
AndroidPit	https://www.androidpit.com/	English

Fig. 1. All marketplaces found in September 2015.

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002

Bulgaria	Yes	Yes	No	No	No	No	No	Yes	Yes
Burkina Faso	Yes	No	No	No	No	Yes	No	No	No
Cambodia	Yes	No	No	No	No	Yes	No	No	No
Cameroon	Yes	No	No	No	No	No	No	No	No
Canada	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Cape Verde	Yes	No	No	No	No	Yes	No	No	No
Chile	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes
China	No	Yes	No	No	No	No	No	No	No
Colombia	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes
Costa Rica	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes
Croatia	Yes	No	No	No	No	Yes	No	Yes	Yes
Cyprus	Yes	Yes	No	No	No	Yes	No	Yes	Yes
Czech Republic	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes
Denmark	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes
Dominican Republic	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes
Ecuador	Yes	No	No	No	Yes	Yes	No	Yes	Yes
Egypt	Yes	Yes	No	Yes	Yes	No	No	No	No

Fig. 2. List of countries with relative Play Store features (Google Inc, 2016).

Initially in 2012 Bouncer was nothing more than a list of emulated devices on which try to install and launch applications trying to trigger malicious events. All the emulated apps are analysed in a dynamic and behavioural way in order to see potential malicious activities. Google chose dynamic analysis instead of the cheaper and usually more efficient static one in order to prevent obfuscation and repackaging techniques that are very popular and easy to perform. Dynamic analysis is more robust against code level evading methods and can detect new threats easily, despite it still cannot guarantee a total protection from code evasion (Diao, 2016).

Not much technical information has been revealed about Bouncer but an in depth analysis made during *SummerCon* 2012 found out some IP addresses used by Google emulated devices and also some used by manual operators. They discovered that an unverified application run for 5 min on Emulated Android environment (QEMU) on Google's infrastructure which have a permission for external network access. Using this access the researchers fingerprinted the exact QEMU version, the emulated device ID and the owner name. The way Bouncer explores applications is via emulated UI input, which are easy predictable and many methods can be used to exploit this type of code verification (Oberheide and Miller, 2012).

It is interesting to notice how two of the most famous and widespread malware families, Dendroid and the more recent GM Bot (also known as Mazar), used simple approaches to avoid Bouncer check. Dendroid used a hard coded flag to determine if the application is running in Google Bouncer. It simply checks DEVICE, PRODUCT, CPU, MODEL and BRAND of the system to see if they match with an emulator one (Blue Coat Labs, 2014).

GM Bot and other spyware and adware software use newer ways to pass undetected. An interesting method used by *AdDisplay.Cheastorm* is to obtain the IP address of the device and check it with WHOIS record. If it returns a string with the word *Google* inside it assumes to be run inside Bouncer and so basically it decides to postpone or directly not run the malicious code (Stefanko, 2015).

As it is possible to evaluate from the two examples described above, Bouncer can still be bypassed with customized techniques. A properly skilled malware developer can find a way with more sophisticated tricks and get also more information directly from Google Bouncer even if according to Oberheide and Miller Google is very severe and restrictive when it comes to uploading malicious applications, with a direct ban of the IP address used by the application owner (Oberheide and Miller, 2012).

As far as researchers know, Google Bouncer can be avoided mainly through two attack types:

- **Delayed Attacks**: an application can contain malicious code when it is checked but its behaviour results not malicious inside Bouncer execution. Malicious code is executed only once application is installed on a user device.
- **Update Attack**: no malicious code is included within the application. Once it passes Bouncer check and it is installed on a user device, the application downloads malicious code from a server or tries to connect to a C&C server to send user's sensitive data.

Both attacks are based on *dynamic code loading*, which is used primarily to download and execute code from a live application. Analysing these assumptions and considering recent facts, it seems that Google Play Store is still not safe and that it works better in a post reaction way, doing an immediate ban of a malicious application from the market and, thanks to the way Android OS itself is built, eventually making a remote deletion inside infected devices.

Third party marketplaces

Third party marketplaces are evolving fast because of the high quantity of competitors and currently most of them present external metadata along with each application. Often they show package name, developer name, update contents, upload date and reviews to convince users to download from their webstore. Most of them have also added internal metadata like MD5 hashes, developer fingerprint, Android permissions, size and application version. A real problem however is that there are often little or no security policies for third party marketplaces and just a few of them can guarantee an AV check for applications. For that reason it is very easy for them to spread malware or grayware.

Yi Ying Ng and his team demonstrated well the general diffidence of people about unofficial marketplaces through a software called *TransRank*, which has shown (focusing on China) more than 36% of analysed applications as highly suspicious. Developers knows that publishing and deleting applications continuously or hopping between unofficial marketplaces is a key strategy to stay unchecked. In addition to that it is known that code control and dynamic analysis are computationally intensive if they have to be done for thousands of applications. For this reason marketplace owners prefer to avoid them (Ng et al., 2014).

Malware databases

As of 2016 not so many Malware Datasets have been collected specifically for Android. The first and probably most well known is

called the Genome Project, created and used by Zhou et al., in 2012, it was the first overview of the Android malware world (Zhou, 2012). It is a dataset, shared only academically, of 1260 samples categorized in 49 different families. They analysed every aspect of each malware such as the infection method, the malicious aim, the AV evasion techniques, and the triggering action.

The analysis made by Zhou et al. was the first one to reveal that the BOOT_COMPLETED event was being used by malware developers to wait for the right moment to attack. It revealed also that some malware hijacks the main activity or the handler of the UI components. Even if it is a complete research paper, often it omits the exact conditions used to launch the malicious part of the application, the result of which makes it difficult to replicate parts of the experiment. A good example is DroidKungFu, where from the article it is possible to know that it can be triggered at boot time but it is not specified that there is a time bomb which launches the malicious part only after 240 min waiting. Without this information it is difficult to analyse it properly with other tools (Zhou et al., 2012).

A second and more recent dataset was presented in 2014 in Artz et al. paper. They created *FlowDroid*, a static taint analysis tool used to monitor Android apps. With static analysis they detect data leaks, but in order to test and validate it Artz et al. developed also *DroidBench1,* which is a database of apps implementing different types of data leakages. They classified applications through methods used to leak data with a description of the performed leak. It is interesting to note that the source code for every leak is given and it is useful for evaluation and comparisons. The database contains 120 APKs along with minimal source code. Even if it is a very thorough project and database, samples are not real malware and often are too easy to be spread in "real world". Plus, samples are not mixed with benign code as is usual in malware (Arzt et al., 2013).

A third well known malware dataset is the *Contagio Mobile Minidump* project which is basically a blog used to store malware of any type along with their descriptions. The idea was to offer an online repository to share malware samples, encrypted via password protected zip files, in order to manage who could download them. Even if it is easily accessible, at the time of writing, it still only contains a few samples and relative hashes (Contagio, 2016).

Methodology and design

In order to test the application nine marketplaces will be examined. While Zhou Y. et al. used as base pool a total of 204,040 APKs among 4 marketplaces in 2011, because of time constraints, only 1000 APKs will be download per marketplaces gathering a total of 9000 applications to analyse. It is important also to get the same number of APKs from each marketplace so to be as fair as possible for statistical results. To permit discussions and comparisons with older projects, the chosen regions for the marketplace locations follow the only similar location based research, made by W. Zhou et al., which is performed on Western Countries, East Europe, and Chinese markets (Zhou et al., 2012). Unfortunately, due to instability of third party marketplaces, it is impossible to adopt the same unofficial market list used in other studies. Furthermore, some marketplaces have changed over the years improving their security policies and a few of them have built anti download checks that could block completely the analysis of the specific market.

For the reasons listed above the resulted marketplace schema used as input for the application will be: BAIDU (China); WANDOUJIA (China); XIAOMI (China); FDROID (Europe); FREEWARELOVERS (Europe); ONEMOBILE (Europe); SOFTPORTAL (Russia); TEGRAMARKET (Russia); and UPTODOWN (Russia).

A sub criteria chosen to apply to each market is linked to which kind of APKs should be downloaded. In Google Play Store and usually also in third party marketplaces Android APKs are split in categories so to help users to choose what to download. Following the schema used by W. Enck et al. if the marketplace has a list of top downloaded/popular applications, those ones will have priority over the others. In case there is no top list, APK downloads will be equally spread through all the categories so to obtain an equal distribution of APK type and size (Enck et al., 2011).

Before continuing with the in depth analysis it has to be specified that because of the quantity and variety of AVs used by VirusTotal, a high number of false positive results could be returned. To compensate this problem, it has been chosen to only define an APK as malicious if 6 or more AVs out of an average of 55 detect it as a threat. Vice versa APKs with 5 or less detection are considered genuine for the aim of this project.

Crawler

The first section to design is the way to catch URLs once a marketplace is given. It is possible to find many crawling projects on GitHub used for different purposes and situations but just few of them are designed to catch APKs specifically from marketplaces. In particular *apkcrawler* developed in the "Open GApps" GitHub channel uses a main Google API interface to interact with Google Play Store easily, plus another eight marketplaces. It is becoming popular as the main APK crawling application but it is developed only to download main GApps. GApps are APKs directly developed by Google and usually preinstalled on Android smartphones (GitHub, 2016). Because of the criteria specified above the crawler has to be per market and per category customizable so to easily select a range of application to catch. After a deep analysis of the marketplace websites the resulting schema in Fig. 3 represents which design is considered as the best solution.

As Fig. 3 shows the crawler is designed to get marketplace's details and capture through a script all the URLs. After that, a per link cycle checks for the existence of an identical file and in case of success it goes through a multithread download engine, built with the aim to save time in download and avoid single data failures. As an output, the entire system will produce a log and print a row to the shell in order to manage the current situation properly.

Scanner

The second relevant section to be designed is the Scanner part. It is the core of the application and it will take files from the crawler output to check them and mark them as malicious one or good one. To do this it is fundamental to rely on a malware database considered as a trustable source. From the literature review, only VirusTotal results to have a complete and updated malware database system and for this reason it will be used as scanner engine during this project.

At the same time metadata information of each specific APK are extracted from the package and then details from both the metadata and the VirusTotal report are collected and stored. From VirusTotal it is possible to return three results:

- OK result, which indicates a clean APK.
- Malicious result, which indicates the APK detected as malicious by one or more AVs.
- Not Found, which indicates that the hash of the specific APK is not stored in VirusTotal Database.

VirusTotal works through a personal account analysing the MD5 hash, in this scenario, and matching it with 59 AVs. If an AV finds the hash positive it will send back also the category of the malware (VirusTotal, 2016). All the information captured from both

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002

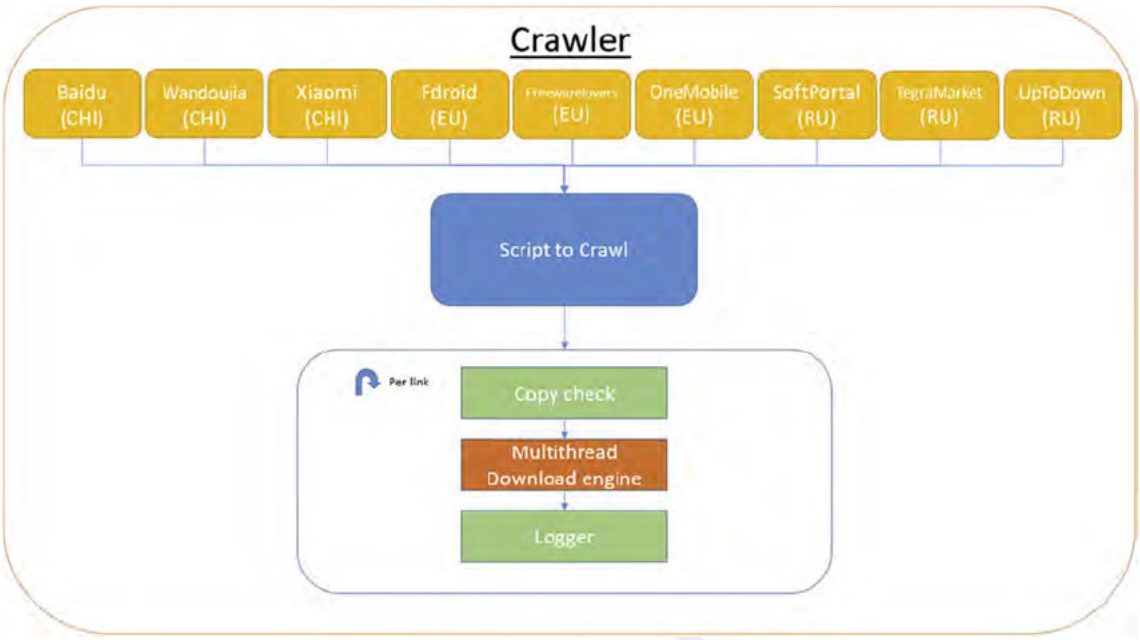

Fig. 3. Design of the crawler.

metadata and VirusTotal are stored in per marketplace tables to be used later in statistical analysis.

In Fig. 4 it is possible to see the entire Scanner process in which, for each file downloaded during crawling phase, metadata with MD5 are extracted and sent to VirusTotal for a malware check. In detail, the relevant stored data are:

- The APK package unique name used as identifier (from metadata).
- The version number of the APK (from metadata).
- The MD5 hash signature (from metadata).
- The total number of AVs used (from VT).
- The number of positive detections (from VT).
- The list of Android Permission requested (from metadata).

New file scanner

The third part of this tool is tasked with scanning "never before scanned" files and storing the results within a specific Exceptions Table. In Fig. 5, instead of sending an MD5 verification, the picked up files go through a three step process. The first step is a proper upload and analysis on VirusTotal, the second one is a main permission based check with Google Play Store to see if some permissions have been altered, and the third one is an upload to a

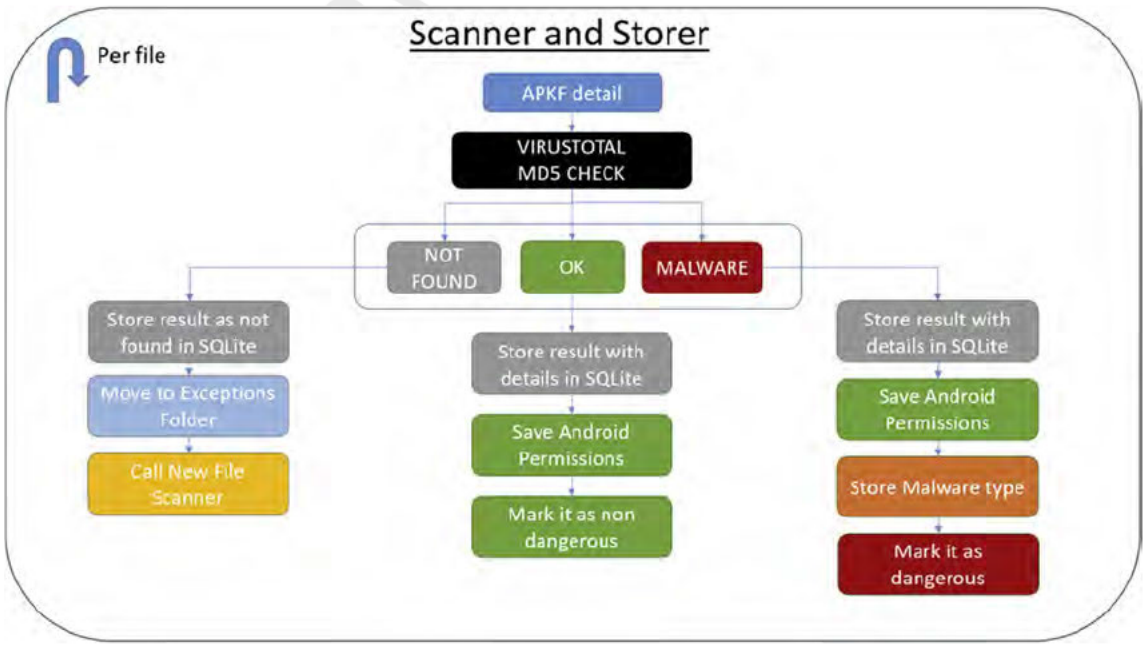

Fig. 4. Design of the file scanner.

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002

dynamic data leak online scanner called AppAudit. Results from these three steps will be stored to define later if the specific APK is malicious or not.

Database design

A key point for the application and for further statistical analysis is a structured local Database system used to store details for each analysed APK. Taking considerations of the aims analysed in the literature review, part of the database has been built on a per marketplace structure as outlined in Fig. 6. Each table inside Mar kets Database follows a precise structure which represent the de tails of the applications stored inside it. This structure can be summarised as follows (Fig. 7).

Results

This section of the dissertation is used to test the designed and implemented software in the real scenario and analyse the results. Due to the large amount of information extracted from the exper iment it will be possible to build different graphs and discuss re sults and their trustiness critically.

Data collection overall

Using the tool designed and implemented in previous sections it was possible to obtain detection data from 9000 APKs. As specified at the beginning of the project three main regions have been cho sen (China, Europe, Russia) as information sources. To reach trust iness and equity unofficial marketplaces chosen use no AV automated control over their APKs and permit crawling through automated scripts.

To begin to analyse the paper's results, an overall view of the downloaded and scanned APKs is plotted to a pie chart, so to show a realistic proportion of detected applications. As can be deduced from Fig. 8, analysing a total of nine marketplaces spread through Europe, China and Russia, 64% of the downloaded APKs scanned via VirusTotal result as Negative (or genuine). Only 5% of the total APKs

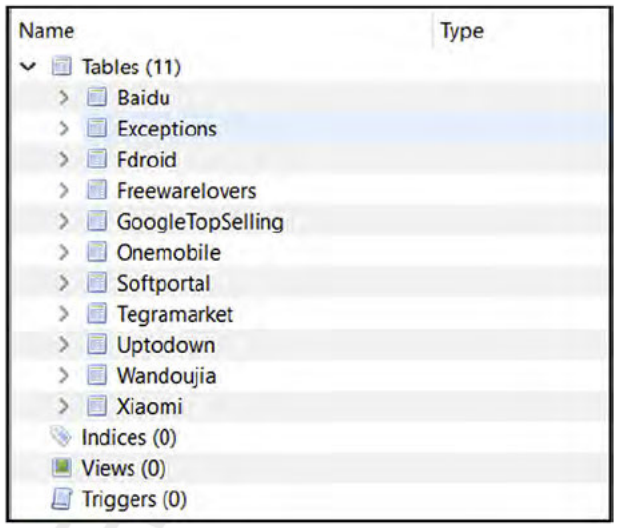

Fig. 6. Database structure used for the software.

have been detected from more than 5 AVs but it is still circa 450 malicious APKs. Furthermore, 31% of the total downloaded APKs are still unknown because they have never been uploaded to VirusTotal.

At this point, it is important to remember that these statistics are based on AV detections and for this reason, it means that it is not guarantee that the 64% of the applications are completely safe, although it is more likely that they do not contain malicious code (or at least currently known malicious code). At the same time, the 31% of new applications to be stored, reflects an Android world in continuous growth, with many new or repackaged applications built every day, and for this reason, difficult to manage in terms of security.

A second chart pictured in Fig. 9 represents more in depth the division of detected malicious activities throughout the three main regions adopted for this project (Russia in green, China in blue,

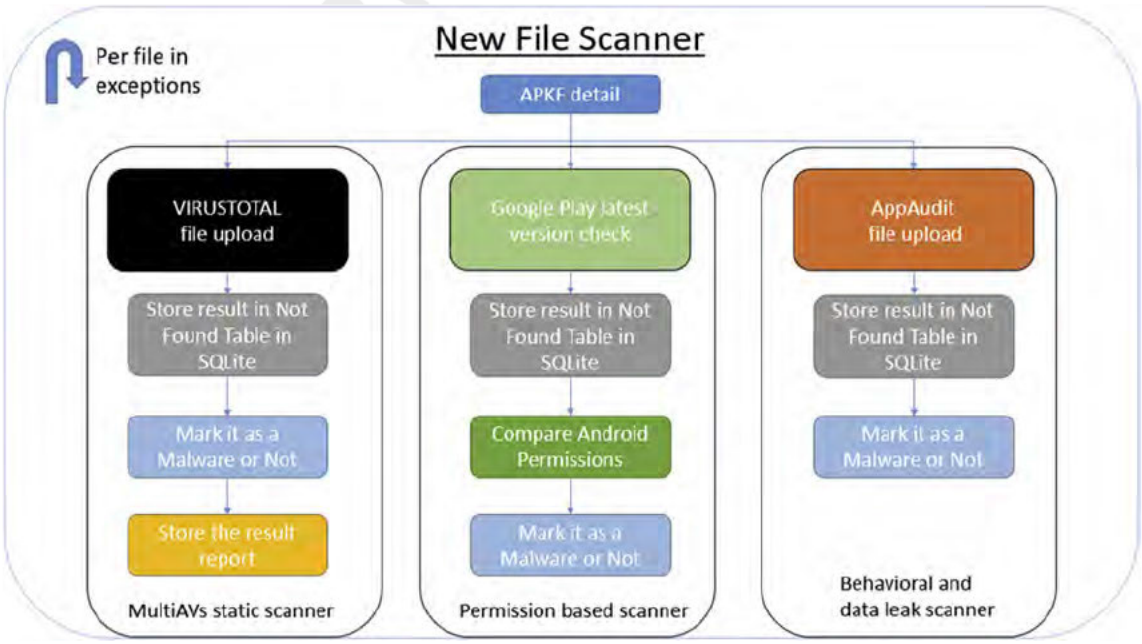

Fig. 5. Design of the new file scanner.

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002

Package	TEXT	`Package` TEXT
Version	INT	`Version` INT
Name	TEXT	`Name` TEXT
MD5	TEXT	`MD5` TEXT
GoogleCheck	INT	`GoogleCheck` INT
VirusTotal	INT	`VirusTotal` INT
TotalAVs	INTEGER	`TotalAVs` INTEGER
PositiveAVs	INT	`PositiveAVs` INT
VirusType	TEXT	`VirusType` TEXT
Permissions	TEXT	`Permissions` TEXT
LastUpdate	DATE	`LastUpdate` DATE

Fig. 7. Table structure used for each marketplace.

Fig. 8. Overall results of analysed unofficial marketplaces.

Europe in red). As it would be expected from the literature review, because Chinese does not have a legal Google Play Store active at the moment, the number of positive detected APKs is considerably higher than the other two regions (248 vs 81 and 90). It is also interesting to note how China has a remarkable difference in terms of "never scanned" APKs.

Data collection malware

The next step that has to be analysed to help to define an average geo localized malware spreading detection scenario, is a list of detected threats. Using a customised Python script it was possible to extract all uniquely detected malware types from each region. Due to the way that AVs work it almost always happened that an APK was detected as malicious, but with a different category name from each AV brand. For that reason it was decided not to specifically check an APK malware type manually, but to plot only the result of detected malware types for files with more than five positive AVs detections. Because the Python script returned a list of hundreds of malware types it was decided to only show the top 15 detected malicious applications.

Fig. 10 defines the Top 15 detected malware samples given by the sum of three analysed Russian marketplaces (Softportal, Tegramarket, Uptodown). As it can be seen, at least 5 of the most detected threats are *AirPush* adware. AirPush is a mobile ad network based on push notifications and instead of using in app advertisements, they add notifications inside the Android tray bar (LaCouvee, 2011).

The first and second detected malware threats, with a number of 51 and 48 respectively, are instead generic detection for Trojan, which usually could refer to adware and spyware. The second histogram, shown in Fig. 11 reveals more or less the same scenario found for the Russian marketplaces, with just an increased number of *Artemis* threats. Artemis was detected for the first time by McAfee and is considered a potentially unwanted program. It usually works as an adware and it can also hijack the main browser.

Fig. 12 reflects the Chinese average marketplace scenario and it is undoubtedly more dangerous than the previous two. Even though the highest number of detections are still generic malware it shows how riskware such as *SMSPay* or the similar *SMSReg* are spread throughout the Chinese web. SMSPay is a malware used to add an SMS activation fee to fake legitimate applications (Wontok, 2015).

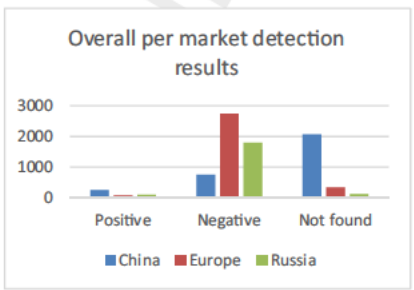

Fig. 9. Overall results divided into Europe, China and Russia.

Fig. 10. Russian marketplaces malware detection histogram.

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002

Fig. 11. European marketplaces malware detection histogram.

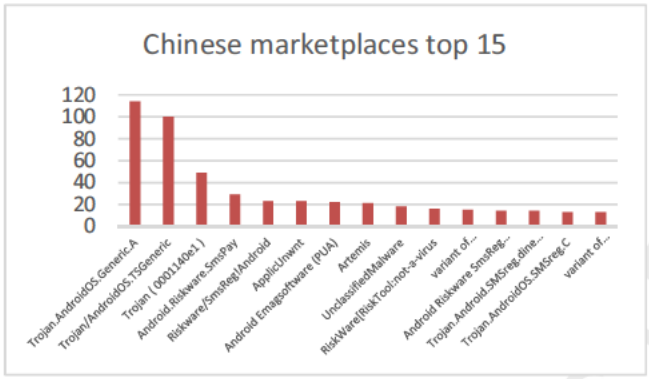

Fig. 12. Chinese marketplaces malware detection histogram.

Exceptions

A last important analysis that has to be done and represents one of the fulcra of the entire project, regards files moved in the exception folders. The New File Scanner used the three detection techniques described in the design section to examine 1622 files, which were effectively APKs that have never been detected (while the rest of the excluded data were damaged or other extensions files). VirusTotal and AppAudit combined revealed potential malware threats in 221 of the 1622 analysed APKs (with a detection rate > 5). With a percentage of 13.62% of po tential malware throughout the never scanned APKs it was revealed how much new or repackaged applications are likely to contain malicious code, especially the ones from Chinese marketplaces.

A second interesting scenario has been outlined via the Google Comparison Permission script, which firstly has revealed that only circa half of the analysed applications were also stored inside the Google official market, but more interesting it revealed how some detected malware threats are probably repackaged version of legitimate one with a higher plethora of permission requested compared with the original one.

Fig. 13 highlights an example of this category were usually games are targeted (in this scenario 'com.ea.game.pvz2_row') in order to make a malicious copy of them. The box on the right of the image represents permission not currently requested in the closest version of the same APK found in the Google Play Store (in that case the Play Store version was 210). It is easy to note how permissions like 'GET_TASKS', 'RECORD_AUDIO', and 'WRITE_SETTINGS', which are potentially dangerous permissions, have been added in the unofficial version.

Another interesting result that could require further analysis has been found in not more than 10 APKs generated by the Exception folder where the exact version of the Play Store one has been detected but a completely different list of permissions was requested. Fig. 14 outlines this scenario for the package 'cn.wps_moffice_eng', which is a similar office program (Fig. 15), where 'Version' and 'PlayStoreVersion' were the same but the field 'PermissionDifference' revealed some suspicious permissions requested. It is interesting to note how VirusTotal did not detect any malicious activity even if those permissions were added.

Comparison

Focusing on projects analysed during the literature review, only one of them performed a geo localized evaluation of the repack aged market inside third party marketplaces. W. Zhou et al. used DroidMOSS, and later a manual verification, to check repackaging rate over 200 randomly chosen APKs from Europe, China and East Europe. In the paper they found Softportal marketplace with 5% of repackaged apps and Freewarelovers with 6% of them. It is possible to note indeed, that statistics evaluated from this results section are aligned with 2011 Zhou results (with all the limitations that could come with it) (Zhou et al., 2012).

A second comparison that has to be made is with Y. Zhou et al. As illustrated in the literature review section they built Droi dRanger, a tool to detect, both in static and behavioural way, mal ware inside Android APKs. They crawled 51,038 applications from five different marketplaces finding 179 malicious activities including some zero day malware. Comparing their results with the ones retrieved from this project it is possible to see that they found circa 3% of malicious applications inside 51,038 APKs against 5% of this research. Even though the results are close, it should be noted that they did not consider Adware inside their research (Zhou et al., 2011).

Significant results

Even though the per market graph shows a clear preponderance of generic malware and adware, searching through the entire list of detected malicious applications some well known and high risk threats have been found and will be listed below, along with a description of their main functionalities:

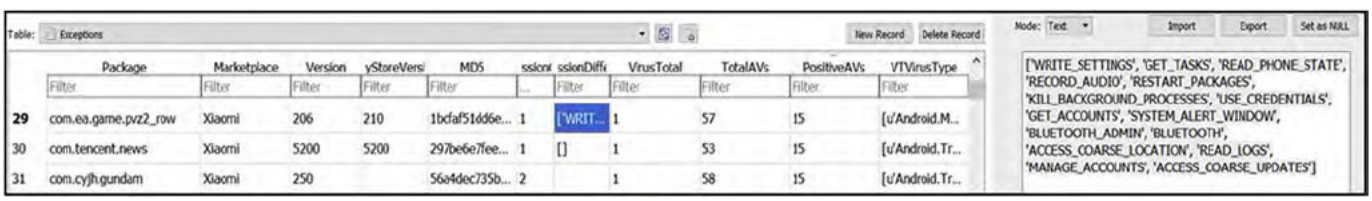

Fig. 13. Repackaged application in Xiaomi marketplace.

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002

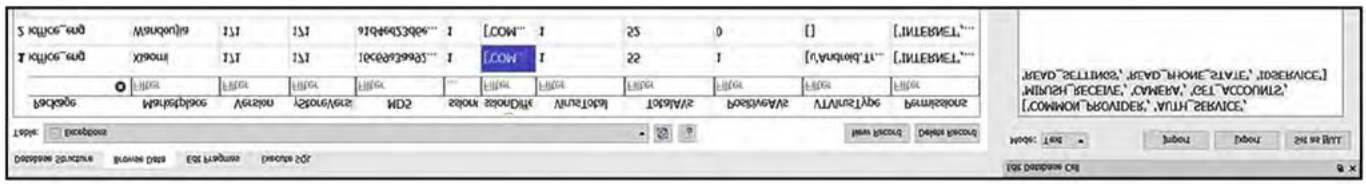

Fig. 14. Same version possible malware detection.

Fig. 15. Example of the original APK in Play Store.

- *GinMaster*, is a note trojanized APK distributed in Chinese alternative Android marketplaces which was discovered for the first time in 2011 and since then has changed its source code through three generations. It uses a malicious service to root the infected device, and then tries to escalate privileges so to send confidential information to a C&C server. It is evolving using a polymorphic structure in order to avoid AV detection (Yu, 2013).
- *Plankton*, is another well known Trojan which is used to forward as much information as possible about the device and user's details. It uses an update attack via an URL sent by a remote server, with a link to a JAR file in it, in order to avoid AV detection (F Secure Corporation, 2016).
- *GoldenEagle*, is instead an older Trojan from 2011 which creates two services mainly to record and store user calls and SMS. It is interesting to note that it is SMS controlled and uses a service to intercept received SMS messages and hide them from users. At boot time it sends an SMS to a specific number which contains the words "A host online, attention please!" (Symantec, 2011).
- *Fusob*, is probably the most interesting of the detected malware. It started between 2014 and 2015 from a Russian developer and along with *Small* is part of the ransomware family in Android. Ransomware are specific malware used to encrypt the user's device and then used to ask for a payment in order to decrypt it. It works by displaying a fake screen which accuses victims of a misdemeanour. With a threat of opening a criminal case, they force users to pay a fine. A peculiarity of this ransomware is that they suggest a payment via iTunes gift cards (Kaspersky Lab, 2016).

Conclusions

As it was analysed in the result section a first overall view of the current situation on the average third party marketplace reveals that 5% of the total APKs have a high likelihood of containing malicious code, 64% are likely to be clean and 31% of them have never been scanned by VirusTotal. These percentages can be considered only as a guideline, because VirusTotal's AVs cannot guarantee to uncover malicious code inside a specific APK. Furthermore, every one of the AVs used by Virus Total uses its own database and method, which brings to a plethora of different malware types detected. For these reasons a lower cap of 6 positive detections was set during this paper to define an APK as malicious. On account of that, two previously analysed studies, DroidMOSS and DroidRanger, have been compared revealing a similar malware and repackaging percentage in similar Android marketplace scenarios.

Geo localized over three main regions (China, Europe, and Russia) has been plotted showing a majority of malware detections focused on China, with 248 probable malicious APKs found. Furthermore, in China a vast number of application was found as never detected by VirusTotal, if compared with the other two regions, confirming that China is a continuously changing scenario with always either new or repackaged applications growing wildly because of the absence of an official Play Store.

Subsequently a proper malware detection comparison has been analysed throughout the three main regions. While Russia and Europe had similar results, with a preponderance of generic malware detection, adware, and Airpush (advertisement mobile network for push notifications) and only a few cases of the most dangerous applications, China had again a completely different scenario, revealing a highly detected number of riskware like SMSPay, which are used to trick money from users.

Finally, searching through all the detected malware types, a couple of noteworthy Trojans have been found like *GoldenEagle* and *Plankton*. Also *GinMaster*, a trojanized application strictly geo localized in Chinese marketplaces was detected numerous times. Furthermore, *Fusob*, a mobile ransomware discovered recently, was also detected throughout the dataset.

As it would be expected, this paper has shown how a high percentage of dangerous APK can still be found over the web through untrusted marketplaces, especially Chinese ones. However, most interesting, it has to be noted how many applications are distributed along the internet without any security check as VirusTotal database revealed.

Uncited reference

Apache, 2016. Q2

References

Apache, 2016. http://www.apache.org/licenses/LICENSE-2.0.
Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., McDaniel, P., 2013. FlowDroid. Proc. 35th ACM SIGPLAN Conf. Program. Lang. Des. Implement. - PLDI '14 259 269.
Blue Coat Labs, 2014. Dendroid under the Hood a Look inside an Android RAT Kit.
Check Point Research Team, 2016. More than 1 Million Google Accounts Breached by Gooligan | Check Point Blog.
Contagio, 2016. Minidump Mobile.
CVE, 2015. Vulnerability Trends over Time.
Diao, W., 2016. Evading Android Runtime Analysis through Detecting Programmed Interactions.
Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S., 2011. A study of android application security. USENIX Secur. 39. August, pp. 21 21.
Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M.S., Conti, M., Rajarajan, M., 2015. Android security: a survey of issues, malware penetration, and defenses. IEEE Commun. Surv. Tutorials 17 (2), 998 1022.
Felt, A.P., Finifter, M., Chin, E., Hanna, S., Wagner, D., 2011. A survey of mobile malware in the wild, pp. 3 14.
F-Secure Corporation, 2016. Trojan:Android/Plankton Description | F-secure Labs.
GitHub, 2016. opengapps/apkcrawler.
Google Inc, 2016. Google Play Help.
Google Mobile Blog, 2012. Android and Security.

Google, 2016. https://developer.android.com/about/dashboards/index.html.
IDC, 2015. Smartphone OS Market Share, 2015 Q2.
Kaspersky Lab, 2016. Mobile ransomware: Major Threats and Best Means of Protection.
Keizer, G., 2011. Google Throws 'kill Switch' on Android Phones | Computerworld.
LaCouvee, D., 2011. Interview with Airpush - the Future of Mobile Advertising? - Android Authority.
Ng, Y.Y., Zhou, H., Ji, Z., Luo, H., Dong, Y., 2014. Which Android App Store Can Be Trusted in China?.
Oberheide, J., Miller, C., 2012. Dissecting the android bouncer. Summercon 2012.
Sisto, A., 2013. AndroCrawl : Studying Alternative Android Marketplaces.
Stefanko, L., 2015. Android AdDisplay Using Anti-bouncer Technique.
Symantec Corporation, 2016. Internet Security Threat Report.
Symantec, 2011. Android. Goldeneagle Technical Details | Symantec.
Trend Micro, 2016. GODLESS' Mobile Malware Uses Multiple Exploits to Root Devices.
Viennot, N., Garcia, E., Nieh, J., 2015. A Measurement Study of Google Play.
VirusTotal, 2016. Advanced Features & Tools.
Wontok, 2015. Wontok Lab Tests Android RiskWare SMSPay Striking APAC - Wontok.
Yu, R., 2013. Ginmaster : a case study in android malware. Virus Bull. 92 104. October.
Zhou, Y., Wang, Z., Zhou, W., Jiang, X., 2011. Hey, You, Get off of My Market : Detecting Malicious Apps in Official and Alternative Android Markets, vol. 2.
Zhou, W., Zhou, Y., Jiang, X., Ning, P., 2012. Detecting repackaged smartphone applications in third-party android marketplaces. Proc. Second ACM Conf. Data Appl. Secur. Priv. - CODASKY '12 317 326.
Zhou, Y., 2012. Dissecting Android Malware : Characterization and Evolution, vol. 4.

Please cite this article in press as: Buchanan, et al., A methodology for the security evaluation within third party Android Marketplaces, Digital Investigation (2017), https://doi.org/10.1016/j.diin.2017.10.002