American Journal of Interdisciplinary Research and Development
ISSN Online: 2771-8948
Website: www.ajird.journalspark.org
Volume 10, Nov., 2022

A FORENSIC ANALYSIS OF ANDROID MALWARE

Abdumuminov Abdurafiq Abdurashidovich
Republican center for management of telecommunications networks of Uzbekistan.

Ibragimov Jalaliddin Obidjon o'g'li
Teacher of the Department, "Systematic and Practical Programming", Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi, Uzbekistan

Shoraimov Khusanboy Uktamboyevich
Teacher of the Department, "Systematic and Practical Programming", Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi, Uzbekistan

ABSTRACT
We consider in this paper the analysis of a large set of malware and benign applications from the Android ecosystem. Although a large body of research work has dealt with Android malware over the last years, none has addressed it from a forensic point of view.
After collecting over 500 000 applications from user markets and research repositories, we perform an analysis that yields precious insights on the writing process of Android malware. This study also explores some strange artifacts in the datasets, and the divergent capabilities of state-of-the-art antivirus to recognize/define malware. We further highlight some major weak usage and misunderstanding of Android security by the criminal community and show some patterns in their operational flow. Finally, using insights from this analysis, we build a naive malware detection scheme that could complement existing anti-virus software.

Keywords: Android Security, Digital Forensics, Malware Analysis, Malware development

I. INTRODUCTION

Android has progressively grown to become in a few years the most widely used smartphone operating system . With more and more users relying on Android-enabled handheld device, and able to install third party applications from official and alternative markets, the security of both devices and the underlying network becomes an essential concern for both the end user and his service provider. In recent years, practitioners and researchers have witnessed the emergence of a variety of Android

malware. The associated threats range from simple user tracking and disclosure of personal information to advanced fraud and premium-rate SMS services subscription, or even unwarranted involvement in botnets. Although most users are nowadays aware that personal computers can and will be attacked by malware, very few realize that their smartphone is prone to an equally dangerous threat.

To assess the threat of software downloaded from the internet, discerning users rely on scan results yielded by antivirus products. Unfortunately, each antivirus vendor has its secret recipe on how/why it decides to assign a malware label to a given application. Thus, distinct antivirus products, leading to damaging confusions, can differently appreciate an application. Indeed, both practitioners and researchers heavily rely on antivirus, whether to trust apps or to build the ground truths for assessment tasks.

For the purpose of our study, we have collected a large and up-to-date dataset of hundreds of thousands of Android applications from markets and repositories. We have then scanned each of these applications using about 45 antivirus products generously hosted by VirusTotal to assess whether they are labelled as malware or not. This effort was made to obtain a clear view of the business of malware writing and some insights in the evolution of malware and its detection by antivirus products. We also take this opportunity to investigate, indirectly, how skilled malware creators are.

Several research studies have investigated Android malware. Most of these academic works are however about using advanced code analysis and data mining techniques to study applications. Thus, there are scarce reports on the actual artefacts that a typical incident responder would rely on in practice. Our study aims at filling this gap by performing such an analysis and reporting our findings based on a large dataset.

II. Preliminaries

An investigation into the business of malware requires a significant dataset representing real-world applications. We have built our dataset by collecting applications from markets, i.e., the online stores where developers distribute their applications to end-users. Indeed, although Google -the main developer of the Android software stack- operates an official market named Google Play, the policy of Google makes it possible for Android users to download and install Apps from any other alternative market.

Alternative markets are often created to distribute specific selection of applications. For example, some of these markets may focus on a specific geographical area, e.g., Russia or China, providing users with Apps in their local languages. Other markets focus exclusively on free software, and at least one market is known to be dedicated to adult content. Users may also directly share Apps, either in close circles, or with application bundles released through Bit Torrent. Such apps are often distributed by other users who have paid for them in non-free markets. Finally, we have included in our datasets, apps that have been collected by others to construct research

repositories.

In the remainder of this section, we provide details on the different sources of our dataset, on the scanning process that were used to label each application as malware or benign, and on the artifacts that we have extracted from application packages to perform our study.

A. Dataset sources

We have developed specialized crawlers for several market places to automatically browse their content, find Android applications that could be retrieved for free, and download them into our repository. In this step we have found that several market owners took various steps in order to prevent their market to be automatically mined. Thus, for two of such markets, we cannot assure that we have retrieved their whole content. However, to the best of our knowledge, the total number of apps that we have collected constitutes the largest dataset of Android apps ever used in research studies.

Google Play[1]: The official market of Android is a web-site that allows users to browse its content through a web browser. Applications cannot however be downloaded through the web browser as any other file would be. Instead, Google provides an Android application[2] that uses a proprietary protocol to communicate with Google Play servers. Furthermore, no application can be downloaded from Google Play without a valid Google account - not even free Apps. Both issues thus outlined were overcome using open-source implementations of the proprietary protocol and by creating free Google accounts. The remaining constraint was time, as Google also enforces a strict account-level rate-limiting. Indeed, one given account is not allowed to

Marketplace	# of Android apps	Percentage
Google Play	325 214	54.73%
appchina	125 248	21.13%
anzhi	76414	12.86%
lmobile	57 506	9.68%
slideme	27 274	4.68%
torrents	5 294	0.89%
freewarelovers	4145	0.70%
proandroid	3 683	0.62%
fdroid	2023	0.34%
genome	1 247	0.21%
apk_bang	363	0.06%
Total	**594 000 Unique apps**	

B. Artifacts of study

To perform our study we have mined information from the application packages focusing on two artifact metadata in Android package files.

Packaging dates: An Android application is distributed as an .apk file which is actually a ZIP archive containing all the resources an application needs to run, such as the application binary code and images. An interesting side-effect of this package format is that all the files that makes an application go from the developer's computer to end-users' devices without any modification. In particular, all metadata of the files contained in the .apk package, such as the last modification date, are preserved.

All bytecode, representing the application binary code, is assembled into a classes.dex file that is produced at packaging-time. Thus the last modification date of this file represents the packaging time. In the remainder of this paper, packaging date will refer to this date.

Certificate Metadata: In the Android platform, a first security measure was made mandatory to guarantee that the authenticity of each application can be traced back to its creator. Thus, all Android applications must be signed with a cryptographic certificate. Certificates are included in the app package to allow end-users to verify the package's signature. For each application from our dataset, we have collected the certificates and analyzed their attributes, including owner and issuer, as described by the X.509 standard .

C. Malware Labelling

over the course of several months, while we collect the dataset, we have undertaken to analyze them with anti virus products actually used in the software market. For our study, we have relied on VirusTotal[1], a web portal that hosts about 40 products from renown anti virus vendors, including McAfee®, Symantec® or Avast®. We have sent all applications from our dataset to VirusTotal and collected the scan results for analysis and correlation studies.

D. Test of Statistical Significance

our forensics analysis is based on a sample of Android applications. Although, to the best of our knowledge, no related study involving Android malware has ever exploited that many applications, there is a need to ensure, for some of our findings, that they are significant. To this end, we resort to the common metric of the Mann-Whitney-Wilcoxon (MWW) test.

The MWW test is a non-parametric statistical hypothesis test that assesses the statistical significance of the difference between the distributions in two datasets [19]. We adopt this test as it does not assume any specific distribution, a suitable property for our experimental setting. once the Mann-Whitney U value is computed it is used to determine the p-value. Given a significance level a = 0.001, if p — value < a, then the test rejects the null hypothesis, implying that the two datasets have different distributions at the significance level of a = 0.001: there is one chance in a thousand that this is due to a coincidence.

III. Analysis

In this section, we describe and interpret the results of our findings on how malware are written, in comparison with benign applications, and how anti virus products perform in their detection.

A. Malware identification by anti virus products

Malware identification by anti virus products is critical to practitioners and researchers alike. Indeed, anti virus products remain the most trusted means to flag an application as malware. Traditionally, the common detection scheme of anti virus is signature-based. Thus, to identify malware statically, antivirus software compares the contents of application files to their secret dictionary of virus signatures. This approach can be very effective, but can only help identify malware for which samples have already been obtained and associated signatures created. Some antivirus products add heuristics to their process in order to identify new malware or variants of known malware.

In Figure 1, we see that most of our data sources contain Android applications that are flagged as malware by at least 1 anti virus product hosted by VirusTotal. Even Google Play, where each application goes through the Bouncer[12], shows a malware-rate of 22%. These malware are often in the form of adware, i.e., applications that continuously display undesired advertisement during use. Anzhi and AppChina include the largest share of flagged applications. Each of all the malware samples from the Genome dataset are indeed flagged by at least one anti virus software.

IV. Discussion

The forensic analysis that we have performed and whose results were outlined in the previous section has yielded a number of insights for the research and practice of malware detection. In this section, we summarize these insights and discuss how this empirical study could be instrumented in our work on malware detection.

A. Summary of findings

On Anti virus software: Our large-scale analysis of hundreds of thousands of Android applications with over 40 anti virus products have revealed that most malware are not simultaneously identified by several anti virus. Only a small subset of common malware is detected by most anti virus software. This finding actually supports the idea that there is a need to invest in alternative tools for malware detection such as machine-learning based approaches which are promising to flag more malware variants.

On malware business: We have presented empirical evidence that malware were mass produced. This raises a number of questions leading to hypothesis on how malware developers manage to remain productive. The first hypothesis would be that, malware is not written from scratch, thus providing an opportunity to detect malware by discovering the piece of code that was grafted to existing, potentially popular, apps.

B. Insights

Building a naive anti virus software: Exploring the rate of shared certificates within malware, we were able to devise a naive malware detection mechanism based on the appearance of a tagged certificate. In its simplest form, the scheme consists in tagging any application as malicious if the signing key has been already observed for a confirmed malicious application.

To assess this naive approach we have considered that in a first phase we have manually discovered all malware packaged before 01/Jan/2013 in our dataset. We consider for this step only malware that are detected by at least half of the anti virus products. Then based on the certificates recorded for the found malware, we arbitrarily tag as malicious all applications packaged after 01/Jan/2013 and that are signed with any of the flagged certificates. Table V provides the results for this experiment. We were able to build a malware detector with a Precision of 84% (2,166 false positives out of 2,166 + 11,460 tagged). While we succeed in flagging almost 1 actual malware out of 10, we only wrongly tag as malicious about 1 benign app in 100.

	Benign apps tagged	**Malware tagged**
Number	2166	11460
Percentage	1.19%	8.82%

Table V PERFORMANCE OF A NAIVE ANTI VIRUS SOFTWARE BASED ON CERTIFICATES

At the minimum, the obtained results show that our naive approach could be used by anti virus vendors to improve their recall, by being suspicious of more apps, and improve precision by trusting apps signed with certificates that have been used in a large number of benign apps.

Localizing malware: Our findings on the potential mass production of malware could be leveraged in an approach of malware localization. Indeed, simultaneous development and packaging of malware suggests a redundant insertion of malware code in all applications. Thus, a similarity measure of the bytecode could allow to isolate this code and then locate it in other malware samples.

V. Related Work

In this section, we enumerate a number of related work to emphasize on the importance of understanding the development of malware in order to devise efficient techniques for their detection. These related work span from empirical studies on on datasets of malware, to malware detection schemes.

A. Malicious datasets analysis

Researchers have already shown interest in malicious application datasets analysis.

Felt et al. have analyzed several instances of malware deployed on various mobile platforms such as iOS, Android and Symbian. They detail the wide range of incentives for malware writing, such as users' personal information and credentials exfiltration, ransom attack and the easiest way to profit from smartphone malware, premium-rate SMS services. Their study is however qualitative, while we have focused on a quantitative study to draw generalizable findings on common patterns.

The Genome dataset, our source of well-established malware, was built as part of a study by Zhou et al. They expose in details features and incentives of the current malware threat on Android. They also suggest that existing anti virus software still need improvements. Our analysis also comes to this conclusion when we demonstrate that most malware cannot be found by all anti virus products.

Opposite to our lightweight forensic analysis approach, Enck et al. did an in-depth analysis of Android applications by using advanced static analysis methods. Doing so, they were able to discover some risky features of the android framework that could be used by malicious applications. However, our approach allowed to highlight interesting patterns that are could be leveraged more easily.

In recent work , Allix et al. have devised a sophisticated Feature set to use in a machine learning-based malware detection for Android. This approach has however proved to be resource-intensive, suggesting further investigations into more straightforward features. The study detailed in this paper is part of the roadmaps we have devised for our investigations.

B. Dynamic analysis

Various solutions have been proposed to detect malicious Android applications. Crowdroid, presented by Burguera et al., performs dynamic analysis of Android applications by first collecting system calls patterns of applications, and then applying clustering algorithms to discriminate benign from suspicious behaviors. Crowdroid strongly rely on crowd sourcing for system calls patterns collection.

Vidas and Christin has investigated applications from alternative markets and compared them to applications from the official market They have found that certain alternative markets almost exclusively distribute repackaged applications containing malware. They have proceeded to propose AppIntegrity to strengthen the authentication properties offered in application marketplaces. Our findings are in line with those, when we note that malware seem to be mass produced, and that the same certificates overlap between malware and benign applications.

VI. Conclusion

The recent and steady rise of Android malware over the past four years has lead to a rapidly growing automation in the malware creation process. Due to the specific nature of development of Android applications, important artifacts leak out and can provide some insights about their creators. We have analyzed the available data

through this perspective. For our large-scale study, we have considered over 500,000 Android applications, which included both benign applications and malware.

Packaging dates show substantial time localization behavior. Waves of packaging can be observed thus shedding a new light on the malware creation process. Digital certificates, albeit self-signed also provide valuable pieces of information. We have observed huge quantities of malware sharing the same private key and thus proving that either keys have been stolen, or those malware have the same origin. On the other hand massive copy/paste coding, relying on directly copying code from popular tutorials and blogs, shows that the malware programming is done at a fast pace by developers lacking elementary cryptography knowledge.

It's surprising to see that most malware writers do not use digital certificates properly and that many of the current mitigation techniques did not check them. However, more troubling is the extent to which private keys seem to have been compromised and that both benign applications and malware share the same certificates.

In the future, we plan to leverage the insights discussed in Section IV. Furthermore, we plan to extend this work by considering also the automated analysis of the bytecode. Some preliminary work have been done and the results are promising.

REFERENCES

1. K. Allix, T. F. Bissyande, Q. Jerome, J. Klein, R. State, and Y. Le Traon, "Large-scale machine learning-based malware detection: Confronting the "10-fold cross validation scheme" with reality," in CODASPY '14, 2019.
2. S. Arzt, S. Rasthofer, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in Conference on Programming Language Design and Implementation (PLDI), 2020.
3. A. Bartel, J. Klein, M. Monperrus, K. Allix, and Y. Le Traon, "Improving privacy on android smartphones through in-vivo bytecode instrumentation," Technical Report, May 2017.
4. A. Bartel, J. Klein, M. Monperrus, and Y. Le Traon, "Dexpler: Converting Android Dalvik Bytecode to Jimple for Static Analysis with Soot," in ACM Sigplan Workshop on the State Of The Art in Java Program Analysis (SOAP), 2018.
5. J. Bickford, R. O'Hare, A. Baliga, V. Ganapathy, and L. Iftode, "Rootkits on smart phones: attacks, implications and opportunities," in HotMobile '10, Maryland, 2016.