UWS UNIVERSITY OF THE WEST of SCOTLAND

How can Android Packed malware be detected and prevented?

Student:
Marius-Lucian Olariu

Supervisor:
Dr. Zeeshan Pervez

Paisley
2019

Contents

List of Figures

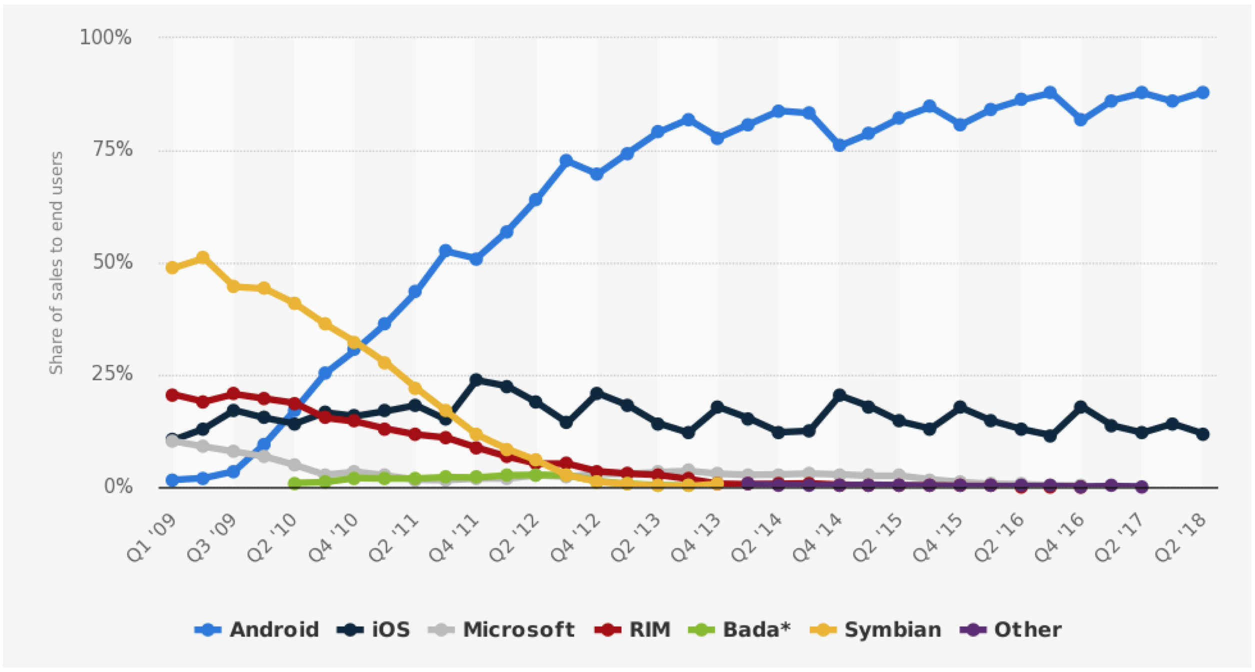
List of Tables

Introduction

(5p - 500w)

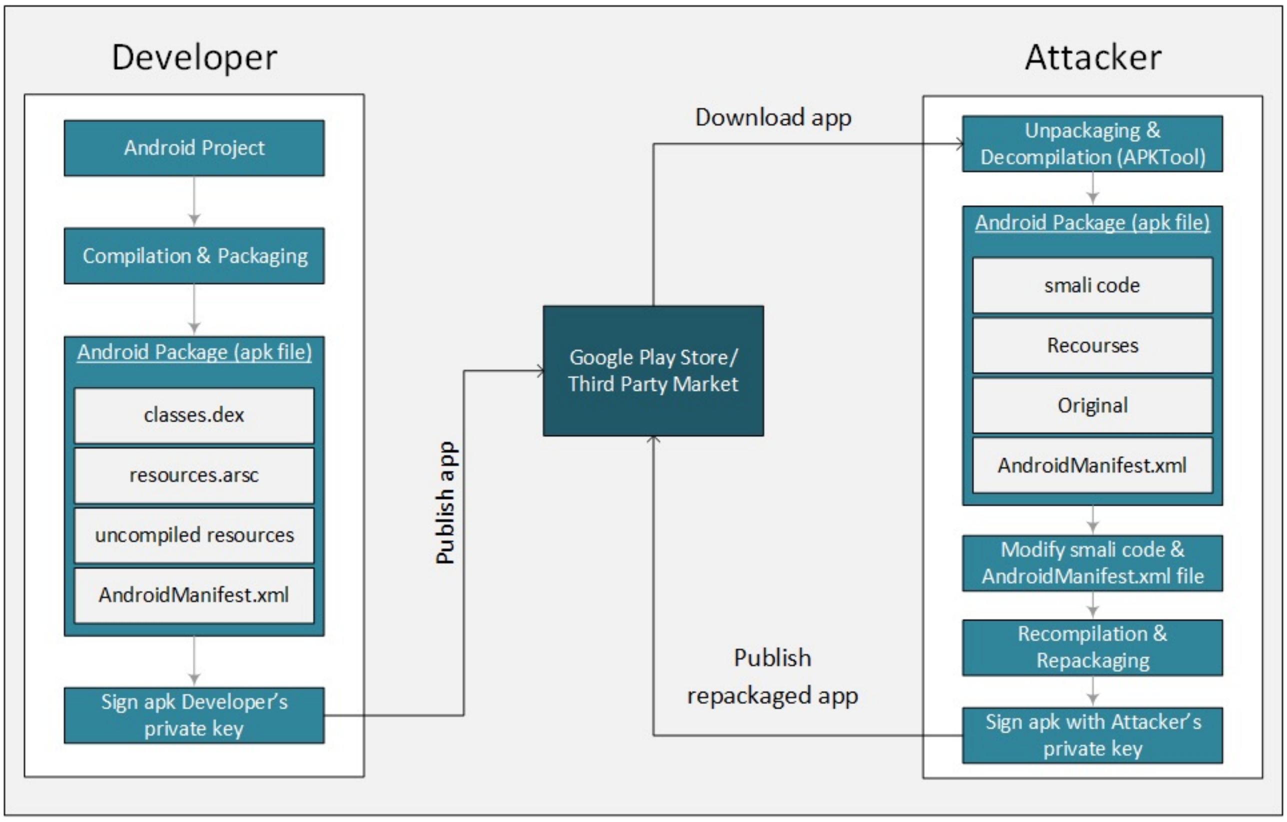
Android is an **O**perating **S**ystem (OS) designed for mobile devices like tablets and smartphones and is currently the market leader of mobile operating systems since of 2016 (Mobile OS Market Share 2018, 2019). Android has been the leading mobile OS in phones sold since 2011 as can be seen in Figure 1 (Mobile OS Market Share 2018, 2019). The CEO of Google, Sundar Pichai, declared at Google I/O 2017 that there are over two billion smartphones running Android OS (Google I/O 2017, n.d). The success of Android is due to the fact that is very efficient (uses a modified version of the Linux kernel) and open-source. The fact that Android is open-source and the large number of users it has attracts a lot of attackers to develop malware (huge material gains and prestige). The problem of Android malware is similar to the one Windows OS faces, in other words, because most of the personal computers run Windows most of the malware targets Windows systems and not Linux or macOS systems. Another fact that makes Android so appealing to malware authors is the fact that is very easy to install applications from unknown sources and there are many third-party markets for distribution of Android applications.

Figure 1: Global market share sales by mobile OSs from 2009 to 2018



Android applications are distributed and installed using the **A**ndroid **P**ac**k**age (.apk) format. The apk format is essentially an archive (was extended from JAR and ZIP) that contains the application compiled code, resources, certificates and the manifest file. Because an apk file it is just an archive it can be easily unarchived, thus it a fairly straight process to get to the complied code of a certain application once you have the apk file. For Android the compiled code is in **D**alvik **Ex**ecutable (DEX) format and because executable code is hard to understand for a human it needs to be further converted. DEX files can be converted to a human-readable format known as *smali* using open-source tools like *Apktool* (Tumbleson, n.d.). Moreover, *Apktool* recovers the xml files and Android manifest file to a form close to the orignal ones. Typically, a smali file coresponds to a Java class code. In order to add malicious code the attacker can add smali code in the app files or add new smali files containg malicious code. For instance one could write a broadcast receiver that reads and sends the phone contacts of the users to a webserver and is triggered when the user restarts his phone, this does not need too much effort from an attacker. Once the malicious code is added to the application, the attacker reasembles everything again into an apk file (representing the malicious Android app). The next step is to sign the newly created apk file which can be done either using an certificate authority or a private key generated by the attacker of the app, obviously the attacker will pick the later variant. The signing of an apk file should help Google identify the author of an application, however, non-official markets are not really interested in finding the author of an app. At this point the attacker has ”cloned” a genuine app and is free to redistribute it. In short, the above described process is known as the *repackaging attack*, please see Figure 2 for a visual depiction of the attack (Wenliang, 2015) .

Figure 2: Overview of the repackaging attack


Developer
Attacker
Android Project
Compilation & Packaging
Android Package (apk file)
classes.dex
resources.arsc
uncompiled resources
AndroidManifest.xml
Sign apk Developer's private key
Publish app
Google Play Store/ Third Party Market
Download app
Unpacking & Decompilation (APKTool)
Android Package (apk file)
smali code
Recourses
Original
AndroidManifest.xml
Modify smali code & AndroidManifest.xml file
Recompilation & Repackaging
Sign apk with Attacker's private key
Publish repackaged app

In order to overcome the above mentioned attack there have been developed Android Packers that can enhance Android applications by providing anti-decompliation, anti-runtime injection and anti-debug techniques. The Android Packers are available only through where the developer can upload his apk file and get it in *packed* form which is safer for distribution on application markets. However, recently the Android Packers have started to be used not only by developers to protect their intelectual property but also by hackers to better hide malware in applications (Yang et al, 2015). A packed application poses a real challenge for cybersecurity specialists because it is really hard to be analyzed to detect if is bening or malicious.

Literature review

(10p - 1000w)

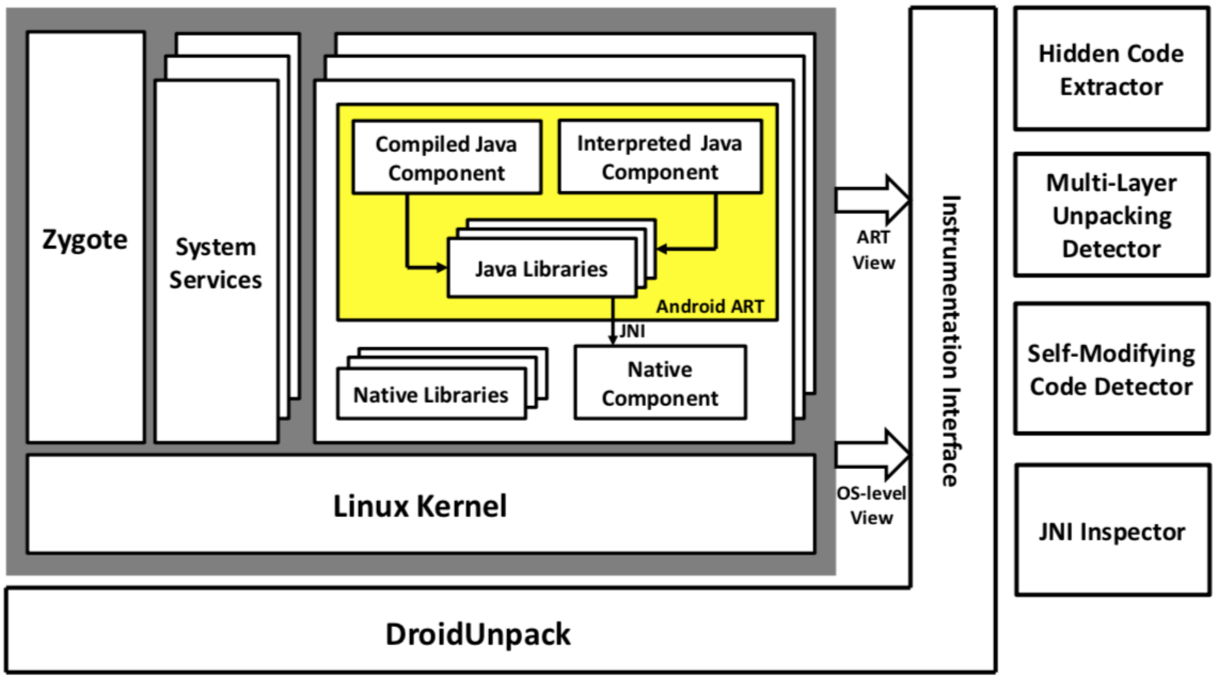
DroidUnpack

DroidUnpack is a framework developed for detection and analysis of Android applications with packed malware and it can monitor app execution at the lowest level and also reconstruct Java level execution (Duan et al., 2018). In other words, DroidUnpack is an emulator that can run the Android OS with the app to be analysed and monitors the memory writes performed by the application. The framework has been tested against 93910 Android applications containing malware, 6 major commercial packers (Ali, Baidu, Bangle, Ijami, Qihoo and Tencent) and 3 unpackers (DexHunter, AppSpear and Kisskiss) over a large time frame (2010 - 2015) and the authors want to make it public in the future.
DroidUnpack makes use of the following tools to analysis tools to investigate a packed application:

- The Hidden Code Extractor
- The Multi-layer Unpacking Detector
- The Self-Modifying Code Detector
- The JNI inspector

For a general view of the framework please see Figure 3.

Figure 3: Overview of the DroidUnpack framework



DexHunter

Security and privacy advancement and glitches

(10p - 1000w)

Real-world scenarios, use cases

(8p - 800w)

Conclusion

(5p - 500w)

References

Defining Malware: FAQ. (2019) [Online] Available: https://docs.microsoft.com/en-us/previous-versions/tn-archive/dd632948(v=technet.10) [Accessed: 6 March 2019].

Duan, Y., Zhang, M., Bhaskar, A.V., Yin, H., Pan, X., Li, T., Wang, X. and Wang, X. (2018)
Things you may not know about android (un) packers: a systematic study based on whole-syster
In 25th Annual Network and Distributed System Security Symposium, NDSS (pp. 18-21).

Google I/O 2017. (n.d.) [Online] Available: https://bit.ly/2CvBOjd [Accessed: 12 March 2019].

Mobile OS Market Share 2018. (2019) [Online] Available: https://bit.ly/2Fwii8f [Accessed: 24 March 2019].

Ruiz, F. (2016) Obfuscated Malware Discovered On Google Play [Online] Available: https://bit.ly/2UcWx5y [Accessed: 12 March 2019].

Tumbleson, C. (n.d.) Apktool. [Online] Available: https://ibotpeaches.github.io/Apktool/ [Accessed: 24 March 2019].

Wenliang, D. (2015) Android Repackaging Attack Lab [Online]
Available: https://bit.ly/2urlfRa [Accessed: 24 March 2019]

Yang, W., Zhang, Y., Li, J., Shu, J., Li, B., Hu, W. and Gu, D. (2015)
Appspear: Bytecode decrypting and dex reassembling for packed android malware. In International Workshop on Recent Advances in Intrusion Detection (pp. 359-381). Springer, Cham.