Challenges in Android Malware Analysis

Valérie Viet Triem Tong, Jean-François Lalande, Mourad Leslous

Challenges in Android Malware Analysis

Valérie Viet Triem Tong[1], Jean-François Lalande[2], and Mourad Leslous[1]

[1] EPI CIDRE
CentraleSupelec, Inria Université de Rennes 1 CNRS, IRISA UMR 6074
F-35065 Rennes, France
valerie.viettriemtong@centralesupelec.fr
[2] INSA Centre Val de Loire
Univ. Orléans LIFO EA 4022,
F-18020 Bourges, France

The best protection against malware is to execute it: a security paradox.

Android has become the world's most popular mobile operating system, and consequently the most popular target for unscrupulous developers. These developers seek to make money by taking advantage of Android users who customise their devices with various applications, which are the main malware infection vector.

Indeed, the most likely way a user will execute a repackaged application is by downloading a seemingly harmless application from a store and executing it. Such an application will have been modified by an attacker in order to add malicious pieces of code. Consequently, a user will have to deal with one of the many different types of malware, such as aggressive adware that constantly display ads making the device unusable, ransomware that encrypt user's data and require a ransom to be paid to decrypt it or remote administration tools (RAT) that take control of the device and allow the attacker to use it as his own.

To fight repackaged applications containing malicious code, most official application marketplaces have implemented security analysis tools that try to detect and remove malware. In this battle between application stores and malware developers, the latter are a step ahead. Malware developers have imagined a lot of countermeasures to defeat security analysis. These countermeasures can be divided into two main approaches: avoiding static analysis and avoiding dynamic analysis.

A static analysis of an application consists of analysing its code and its resources without executing it. For instance, this can simply amount to listing all the permissions required by the application, while checking that these permissions cannot be used maliciously. Static analysis is also used to evaluate the similarity between the application under review and well-known malicious code. Thus to avoid static analysis detection, a developer needs only create unreachable or unintelligible malicious code. Obfuscation techniques, encryption and dynamic loading of code are used to achieve this and disqualify static analysers.

Conversely, dynamic analysis stands for any kind of analysis that requires executing the application in order to observe its actions. Dynamic analysis grants understanding of how an application interacts with other objects in its environment. This kind of analysis is not influenced by the nature of the code executed, whether obfuscated, encrypted or protected by any other means against static analysis. As a matter of fact, the main protection against dynamic analysis coined by malware developers is merely to delay execution of the malicious code. This can be done by waiting for a special event, such as a command sent by a remote server before triggering the malicious code. Moreover, if the malicious code is hard to trigger, dynamic analysis will most likely detect nothing.

The Kharon project goes a step further from classical dynamic analysis of malware. Founded by the Labex CominLabs and involving partners of CentraleSupélec, Inria and INSA Centre Val de Loire, this project aims to capture a compact and comprehensive representation of malware. To achieve such a goal we have developed tools [1] to monitor operating systems' information flows induced by the execution of a marked application.

Figure 1 illustrates an example of such a representation. The studied application's code (.dex file) has been marked and monitored. The graph explains how this piece of code has infected the operating system: which files, sockets and processes have been created or modified. Finally, if the malicious code has been executed, the graph summarises the attack. For example, the execution that has led to the graph represented in

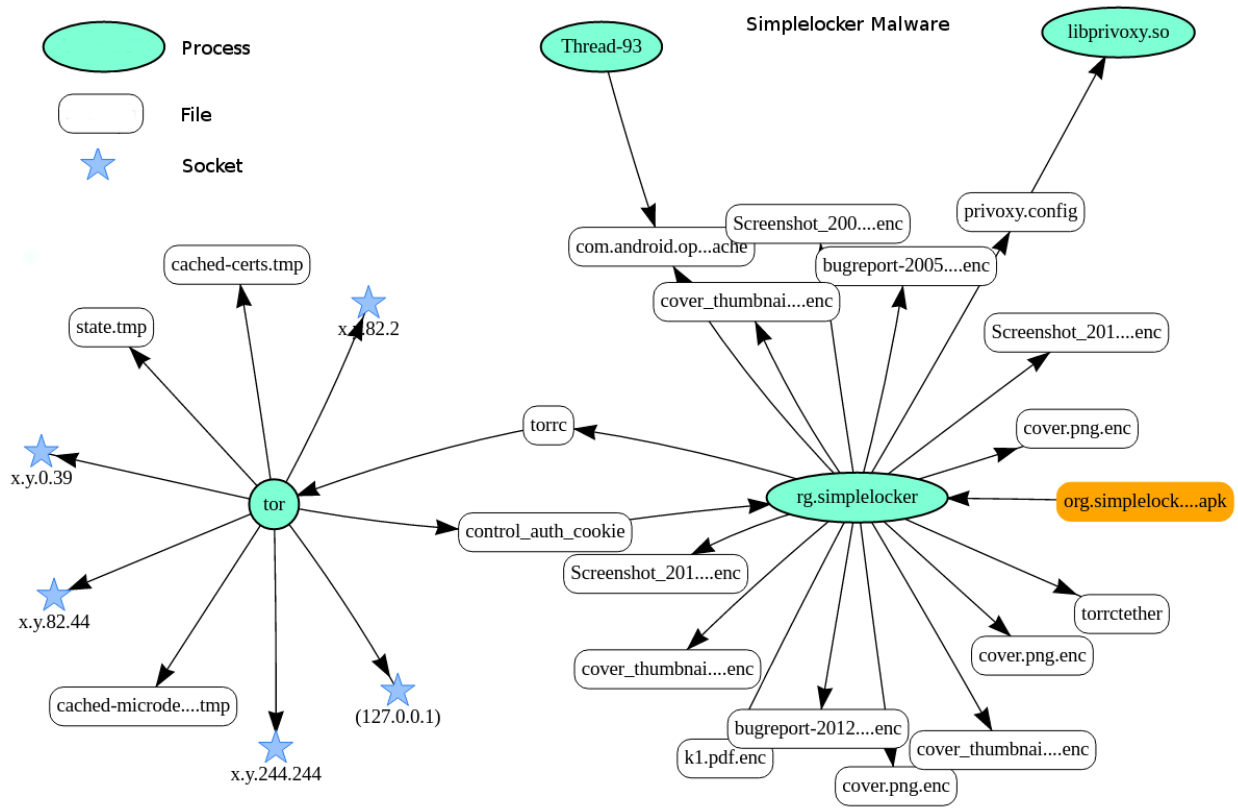

Fig. 1: Comprehensive representation of malware

Figure 1 has been computed from the observation of the execution of a ransomware : SimpleLocker [2]. The graph shows that the malware starts by encrypting a user's files (*.enc files) and then initiates a remote communication through the TOR anonymous network to check if the user has paid the ransom.

In the Kharon project, we support the idea that the best way to understand malware impact is to observe it in its normal execution environment i.e., a real smartphone. Additionally, the main challenge is to be able to trigger malicious behaviours even if the malware tries to escape dynamic analysis.

In this context, we have developed an original solution that mainly consists of 'helping the malware to execute'. In other words we slightly modify the bytecode of the infected application in order to defeat the protection against dynamic analysis and we execute the suspicious code in its most favourable execution conditions. Thus, our software helps us understand malware's objectives and the consequences on the health of a user's device.

Based on these observations, our main research direction and challenge is to develop new and original protections against malicious applications that try to defeat classical dynamic analysis.

References

[1] A. Abraham et al.: "GroddDroid: a gorilla for triggering malicious behaviors", in 10th International Conference on Malicious and Unwanted Software, MALCON 2015, IEEE Computer Society, pp. 119-127, 2015.

[2] N. Kiss, J.-F. Lalande, M. Leslous, V. Viet Triem Tong : "Kharon dataset: Android malware under a microscope", in Learning from Authoritative Security Experiment Results, IEEE Symposium on Security and Privacy Workshop, 2016.