ANDROID MALWARE

CONTENTS

INTRODUCTION

- The development of the mobile operating systems, such as IOS and Android made the operating systems popular.
- High demand of great number of the operating system security holes and a lack of antivirus protection aroused cybercriminals' interest in the development or purchasing of the mobile platforms' malware.
- As a result, it set up new tasks for digital forensic experts.

GENERAL CHARACTERIZATION OF THE ANDROID MALWARE

- Most of the applications on Android operating system are developed using Java programming language.
- The programs are executed in the system via *AndroidRuntime,*starting from version 4.4

FORENSICS
In order to make the applications' analysis, an expert has to understand its format.

THINGS TO KNOW

- to analyze, applications are stored in the mobile device memory in the APK format.
- an application is compiled first and only then it is archived into the APK file with all its parts.
- The file is a **ZIP-archive** that contains **bytecodes, resources, certificates, and manifest-file.**
- After installation, the APK file is copied to a specific location in the system
- *For system apps: /system/appand*
- *For user-installed apps: /data/app.*

FORENSICS ON ANDROID

- an APK file contains three main important parts: ***signature***, ***bytecode*** **and *resources*.**
- The **signature** contains a hash-sum of the APK file, which may help the expert to find out if the application is damaged.
- Moreover, the expert can collect the signatures of applications, including the signatures of malware, in order to speed up a process of malware identification in the memory of an examined device.

Bytecode

- The executable part of the application is stored in the file classes.dex that is in the APK file, and it contains all the compiled classes in the form of **bytecodes**.
- bytecode is converted to the instructions for the ***AndroidRuntime*** virtual machine, as this virtual machine is register-based unlike Java virtual machine. Also, APK file may contain compiled code (catalog lib).

- **<u>Resources</u>**
- Resources are non-executable parts of the application
- Ex: components of user's interface.
- **AndroidManifest.xml :** This file contains information about permissions that the application requires at the installation step.
- Some applications request the permission to use messages, contacts etc., in order to get an access to the protected API files of the Android operating system.
- The analysis of the examined file is the most important step of the malware detection.

Android malware detection;

- Hash can be used in order to detect a malware.
- Hash database can be collected from the data that is represented, for example, in Google Play.
- **HOW** If hash sum of an application does not match any hash from the database , is may be an indication that this application is a malware.

- **Suspicious permission request** is one of the main features of a malware.

- Many users ignore the permission requirement and as a result, the malware arrives.

ANTI-FORENSIC TECHNIQUES AND COUNTERMEASURES;

- *Obfuscation*
- *strings encryption*
- *Decompilation resistance*
- *environment verification.*

OBFUSCATION

- Obfuscation is a technique that allows the developers to safe the functions of an application but the code of it will be changed in the way that it will be hard to make its analysis and to understand its algorithms.

- The expert has to make a decompilation of the malware before the deobfuscation of the code.

- ApkTool : to extract bytecode (in .dex format) from APK.
- A combination of Dex2Jar (decompilation) and JD-CUI (analysis), can be used for decompilation of a bytecode into Java source code.

- The decompiled Java source code has to be edited: there may be a need to remove empty classes, correct errors, rename techniques, classes, objects etc.

STRING ENCRYPTION

- Encryption of Strings of a malware
- developers use both simple techniques, such as XOR, Base64, ROT13 (including its variations, for example, ROT15), and sophisticated ones, such as DES and AES.
- Such applications check not only the system properties, but also international mobile subscriber identity (IMSI).

Malware Analysis

There are many ways to study a program's behavior

- Static
- Dynamic
- Post-Mortem

Analysis of malicious activity traces

There are two types of malware analysis:

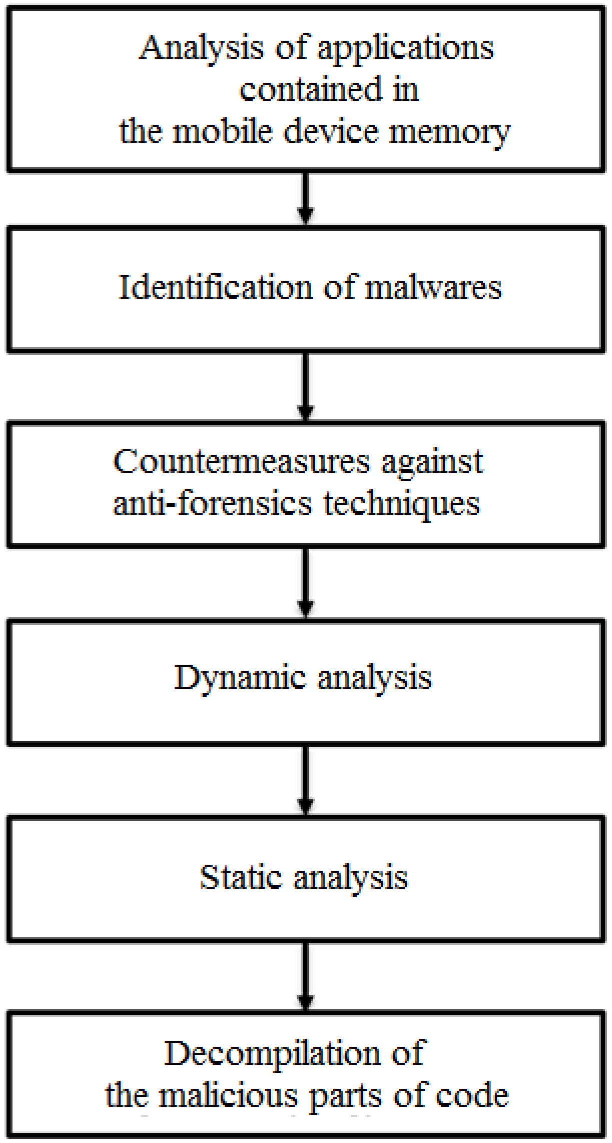
- **dynamic and static.**

DYNAMIC/BEHAVIORIAL ANALYSIS

- In this experts deal with the behavioral features, including the information about the interaction with the system, what kind of data it collects, what kind network connection it setups etc.

Malware Analysis- Dynamic

- Study a program as it executes
- Tools of the trade:
  - Debuggers
  - Function call tracers
  - Machine emulators
  - Logic analyzers
  - Network sniffers.
- Advantage of dynamic analysis is that it can be quick
- Disadvantage of dynamic is what you see what you get.


Analysis of applications contained in the mobile device memory
Identification of malwares
Countermeasures against anti-forensics techniques
Dynamic analysis
Static analysis
Decompilation of the malicious parts of code

cimal-representation-of-a-postgresql-database-table/

- *****Steps of detection and analysis of the malicious activity traces in the Android operating system.**

Static analysis

- No execution
- **Tools of the trade:**
  - Disassemblers
  - Decompilers
  - Source code analyzers
  - Basic utilities as strings and grep
- Reveals how a program would behave under unusual conditions,
- Examine parts of a program that normally do not execute.
- It is impossible to fully predict the behavior of all but the smallest programs

Static analysis

- It makes analysis of its code. The main task is to identify the part of code that executes the malicious activity.
- There are two widely used ways of making static analysis:
- 1) via **ApkTool**
- 2) via the combination of **Dex2Jar and JD-GUI**.

APK TOOL

- ApkTool allows to disassemble a malware.

The analysis include files like:

- **Android Manifest** file, which contains the information about permissions requested by the malware and the information about entry points
- A **res catalog**, which contains XML-files that describe application’s template, and required image files for the application etc.;
- A **smali catalog**, which contains .smali files (operational code) that can be analyzed via text editor with syntax highlighting feature, for example Notepad++.

Malware Analysis- Post-Mortem

- Study of the program behavior by looking at the after effects of execution
- Data analyzed includes:
  - Local or remote logging
  - Changes to file contents or to file access time patterns
  - Deleted file information
  - Data that was written to swap space
  - Data that still lingers on in memory
  - Information that was recorded outside the machine.