Malware Analysis III

Francesco Mercaldo University of Molise, IIT-CNR francesco.mercaldo@unimol.it









Consiglio Nazionale delle Ricerche

Formal Methods for Secure Systems, University of Pisa - 07/05/2021

Target of mobile attack



MALWARE VARIANTS FLOW

220

148

Q3

The reason why



Consiglio Naxionale delle Ricerche - Pisa
 Informatica e Telematica

Malicious Behaviors

- Steal privacy sensitive data
 - Contacts
 - Text messages
- Steal user's money
 - Send text message
 - Register to premium services
 - Try to intercept bank transactions
- Show undesired advertisements (spam)
- Take control of the mobile device

Native Android Security Mechanisms

- Sandboxing (Isolation)
 - Virtual Machine
 - Linux Kernel
- Access Control
 - Permission System
- Storage separation
 - Possible for internal memory (ext3)
 - Not possible for SDCard (fat32)

Sandboxing

- Dalvik Virtual Machine (or ART environment) act as a sandbox for Android applications.
- Each application can perform all of its operations inside the virtual machine.
- Each application operates behaves like if there are no other applications running on the device.
- Application cannot communicate directly.

Isolation

- Every Android application has a different Linux User ID.
- Different storage space: an application cannot modify files of other applications.
- Application execution is interference-free.
- This should avoid the privilege escalation attacks.
- Android applications are normal Linux user without root privilege: an application cannot modify system files.



Eonsiglio Naxionale delle Ricerche - Pisa
 Intro Istituto di Informatica e Telematica

Access Control

- An Android application that will access a critical resource, or will perform a protected operation, have to ask the permission to do so.
- Permissions can be seen like a declaration of intent.
- The application developer declares that the application want to perform a critical operation.

Permissions in Manifest

• Permissions are declared by developer in the manifest file, using a specific XML tag:

<uses-permission android:<u>name</u>="string" />

• Android defines 150+ permissions, identified by the name: android.permission.Permission

Permission Checker

- The permission checker is the component that verifies at runtime, if an application that is going to perform a critical operation, has declared the related permissions.
- If the permission has been declared the operation is allowed, otherwise the operation is denied.

Permission Verification



Static Permission VS Dynamic Verification

- Permissions are declared statically in manifest files. Verification is performed dynamically.
- It is possible that a developer call in the Java code a critical function without asking for the permission in the manifest file.
 - Programming error. No warning are raised! When including a potentially critical function control the API documentation to see the required permissions.

Kind of attacks

- To infect mobile users, malicious apps typically lure users into downloading and installing them.
- Repackaging: downloading popular benign apps, repackaging them with additional malicious payloads, and then uploading repackaged ones to various Android marketplaces.
- Update attack : the malicious payloads are disguised as the "updated" version of legitimate apps.
- Drive-by download: redirect users to download malware, e.g., by using aggressive in-app advertisement or malicious QR code.

Google Bouncer

- Virtual Environment to check if app is malicious
- Runs the app in a phone like environment for around 5 mins before publishing
- Detects most of the known malware...
- Can be bypassed easily

Android application

• APKs file

| META-INF | Cartella di fil |
|-----------------------|-----------------|
| res | Cartella di fil |
| 💥 AndroidManifest.xml | File XML |
| classes.dex | File DEX |
| resources.arsc | File ARSC |

.

APP representations

Difficult to understand but we are able to rebuild the app



ApkTool



- It can decode resources to nearly original form and rebuild them after making *some* modifications
 - In most cases...

```
C:\Users\Seven\Desktop\ToolChain\apktool>apktool d Viber_2.1.6.632.apk
I: Baksmaling...
I: Loading resource table...
I: Loaded.
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Seven\apktool\framework\1.apk
I: Loaded.
I: Regular manifest package...
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Done.
I: Copying assets and libs...
```

Consiglio Naxionale delle Ricerche - Pisa Istituto di Informatica e Telematica

• Java source code/ Java Bytecode visualizer

Bytecode Viewer 2.9.16 - https://bytecodeviewer.com | https://the.bytecode.club - @Konloch

Bytecode Viewer

File View Settings Plugins

| Files | Work Space | | |
|---------------------------------------|---|----------|---|
| i location | android/support/v4/graphics/drawable/DrawableCompat\$DrawableImpl.class x com/baidu/location/aa.class | × | |
| i i i i i i i i i i i i i i i i i i i | | - | |
| | ± ± | ± | T |
| a st class | Procyon Decompiler - Editable: false | В | lytecode Decompiler - Editable: false |
| aa\$2.class | 1 package com.baidu.location; | 18 | class com/baidu/location/aa implements com/baidu/location/au, andro 🛆 |
| aa\$3.class | 2 | 2 | <classversion=50></classversion=50> |
| aa\$a.class | 3 □ import android.os.*; | 3 | |
| aa\$b.class | 4 import java.io.*; | 4 | <pre>private static com.baidu.location.aa j2;</pre> |
| aa.class | 5 import android.hardware.*; | 5 | private boolean j0; |
| | 6 | б | <pre>private android.os.Handler jl;</pre> |
| | 7 class aa implements au, SensorEventListener, b | 7 | private boolean j3; |
| | 8 🖂 (| 8 | private int j4; |
| | 9 private static aa j2; | 9 | <pre>private java.lang.StringBuffer j5;</pre> |
| | 10 private boolean j0; | 10 | <pre>private android.hardware.SensorManager j6;</pre> |
| < > | 11 private Handler j1; | 11 | <pre>private java.lang.StringBuffer j7;</pre> |
| Quick file search (no file extension) | 12 private boolean j3; | 12 | <pre>private java.lang.Runnable j8;</pre> |
| Evart - + | 13 private int j4; | 13 | private final int j9; |
| | 14 private StringBuffer j5; | 14 | <pre>private java.lang.Runnable jS;</pre> |
| Search | 15 private SensorManager j6; | 15 | <pre>private android.hardware.Sensor jT;</pre> |
| Jean | 16 private StringBuffer j7; | 16 | private final int jU; |
| Search from All_Classes v | 17 private Runnable j8; | 17 | private boolean jV; |
| Strings 🗸 🗸 | 18 private final int j9; | 18 | private boolean jW; |
| Search String | 19 private Runnable jS; | 19 | private int jX; |
| Starth String. | 20 private Sensor jT; | 20 | private boolean jY; |
| Exact | 21 private final int jU; | 21 | <pre>private android.hardware.Sensor jZ;</pre> |
| Search | 22 private boolean jV; | 22 | |
| t. Deaths | 23 private boolean jW; | 23 E | <pre>private aa() { // <init> //()V</init></pre> |
| Results | 24 private int jX; | 24 | TryCatch: L1 to L2 handled by L3: java/lang/Exception |
| | 25 private boolean jY; | 25 | aloadO // reference to self |
| | 26 private Sensor jZ; | 26 | invokespecial java/lang/Object. <init>()V</init> |
| | 27 | 27 | aloadO // reference to self |
| | 28 private aa() { | 28 | iconst_1 |
| | 29 this.j9 = 1; | 29 | putfield com/baidu/location/aa.j9:int |
| | 30 this.jU = 2; | 30 | aloadO // reference to self |
| | 21 thin in - false: | 21 | ianat 2 |
| | | | • |

java -jar Bytecode-Viewer-2.9.16.jar

– 0 ×

Refresh