Kharon dataset: Android malware under a microscope

Nicolas Kiss  Jean-François Lalande  Mourad Leslous  Valérie Viet Triem Tong

The LASER Workshop 2016
Learning from Authoritative Security Experiment Results

May 26th 2016
Lessons learned

Android malware findings

- malware hide themselves from dynamic analysis
- triggering malware is not obvious

Methodology:

- manual reverse engineering of 7 malware
- manual triggering (not obvious)
- execution and information flow capture

By Con-struct + replicant community [CC BY-SA 3.0]
Lessons learned

Android malware findings

- malware hide themselves from dynamic analysis
- triggering malware is not obvious

Methodology:

- manual reverse engineering of 7 malware
- manual triggering (not obvious)
- execution and information flow capture

By Con-struct + replicant community [CC BY-SA 3.0]
Why building such a dataset?

Papers with **Android malware experiments**:

- use **extracts of reference datasets**:
  - The Genome project (stopped !) [Zhou et al. 12]
  - Contagio mobile dataset [Mila Parkour]
  - Hand crafted malicious apps (DroidBench [Artz et al. 14])
  - Some Security Challenges’ apps
- need to be **significant**:
  - Tons of apps (e.g. 1.3 million for PhaLibs [Chen et al. 16])
  - Some apps (e.g. 11 for TriggerScope [Fratantonio et al. 16])

- A well documented dataset does not exist !
- Online services give **poor information** !

---

**contagio mobile**  
**virus total**
Why building such a dataset?

Papers with Android malware experiments:

- use extracts of reference datasets:
  - The Genome project (stopped!) [Zhou et al. 12]
  - Contagio mobile dataset [Mila Parkour]
  - Hand crafted malicious apps (DroidBench [Artz et al. 14])
  - Some Security Challenges’ apps
- need to be significant:
  - Tons of apps (e.g. 1.3 million for PhaLibs [Chen et al. 16])
  - Some apps (e.g. 11 for TriggerScope [Fratantonio et al. 16])

- A well documented dataset does not exist!
- Online services give poor information!
Analyzing malware

Main analysis methods are:

- **static analysis**:  
  ⇒ try to recognize known characteristics of malware in the code/ressources of studied applications

- **dynamic analysis**:  
  ⇒ try to execute the malware
Analyzing malware

Main analysis methods are:

- **static analysis**: try to recognize known characteristics of malware in code/resources.
- **dynamic analysis**: try to execute the malware.

**Countermeasures:**
- Reflection, obfuscation, dynamic loading, encryption
- Logic bomb, time bomb, remote server
Methodology

APK → Manual / Automatic Execution → Monitoring Informations Flows → AndroBlare

Manual decompilation → ~ Source code review
Kharon dataset: 7 malware:

http://kharon.gforge.inria.fr/dataset

- DroidKungFu, BadNews (2011, 2013)
- WipeLocker (2014)
- MobiDash (2015)
- SaveMe, Cajino (2015)
- SimpleLocker (2014)

Approved by Inria’s Operational Legal and Ethical Risk Assessment Committee: We warn the readers that these samples have to be used for research purpose only. We also advise to carefully check the SHA256 hash of the studied malware samples and to manipulate them in a sandboxed environment. In particular, the manipulation of these malware impose to follow safety rules of your Institutional Review Boards.
Remote admin Tools

Install malicious apps:

- **Badnews**: Obeys to a remote server + delays attack
  Triggering: Patch the bytecode + Build a fake server

- **DroidKungFu1** (well known): Delays attack
  Triggering: Modify 'start' to 1 in `sstimestamp.xml` and reboot the device
Wipes of the SD card and block social apps:

- **WipeLocker**: Delayed Attack
  Triggering: Launch the app and reboot the device
Displays adds after some days:

- **MobiDash**: Delayed Attack
  Triggering: Launch the application, reboot the device and modify `com.cardgame.durak_preferences.xml`
Spyware

Steals contacts, sms, IMEI, . . .

- **SaveMe**: Verifies the Internet access
  Triggering: Enable Internet access and launch the app

- **Cajino**: Obeys a Baidu remote server
  Triggering: Simulate a server command with an Intent
Ransomware

Encrypts user’s files and asks for paying:

**SimpleLocker**

- Waits the reboot of the device
  Triggering: send a `BOOT_COMPLETED` intent

More details about SimpleLocker...
Example: SimpleLocker

The main malicious functions:

```java
org.simplelocker.MainService.onCreate()
org.simplelocker.MainService$4.run()
org.simplelocker.TorSender.sendCheck(final Context context)
org.simplelocker.FilesEncryptor.encrypt()
org.simplelocker.AesCrypt.AesCrypt(final String s)
```

The encryption loop:

```java
final AesCrypt aesCrypt = new AesCrypt("jndlasf074hr");

for (final String s : this.filesToEncrypt) {
    aesCrypt.encrypt(s, String.valueOf(s) + ".enc");
    new File(s).delete();
}
```

The System Flow Graph:
Let’s discuss :)

Discussion

Kharon dataset: Android malware under a microscope
## Dataset overview

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Protection against dynamic Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAT</td>
<td>Badnews</td>
<td>→ Remediation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>← Obeys to a remote server and delays the attack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Modify the apk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Build a fake server</td>
</tr>
<tr>
<td>Ransomware</td>
<td>SimpleLocker</td>
<td>← Waits the reboot of the device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ send a BOOT_COMPLETED intent</td>
</tr>
<tr>
<td>RAT</td>
<td>DroidKungFu</td>
<td>← Delayed Attack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Modify the value <code>start</code> to <code>1</code> in <code>sstimestamp.xml</code></td>
</tr>
<tr>
<td>Adware</td>
<td>MobiDash</td>
<td>← Delayed Attack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Launch the infected application, reboot the device and modify <code>com.cardgame.durak_preferences.xml</code></td>
</tr>
<tr>
<td>Spyware</td>
<td>SaveMe</td>
<td>← Delayed Attack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Enable Internet access and launch the application</td>
</tr>
<tr>
<td>Eraser+LK</td>
<td>WipeLocker</td>
<td>← Delayed Attack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Press the icon launcher and reboot the device</td>
</tr>
<tr>
<td>Spyware</td>
<td>Cajino</td>
<td>← Obeys to a remote server</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Simulate the remote server by sending an intent</td>
</tr>
</tbody>
</table>
You are now able to execute the malicious code in a real environment and conduct precise experiments.

Kharon dataset is online!

- descriptions and code extracts
- malicious method names
- Graph representation:
  ⇒ replay the malware!

http://kharon.gforge.inria.fr/dataset