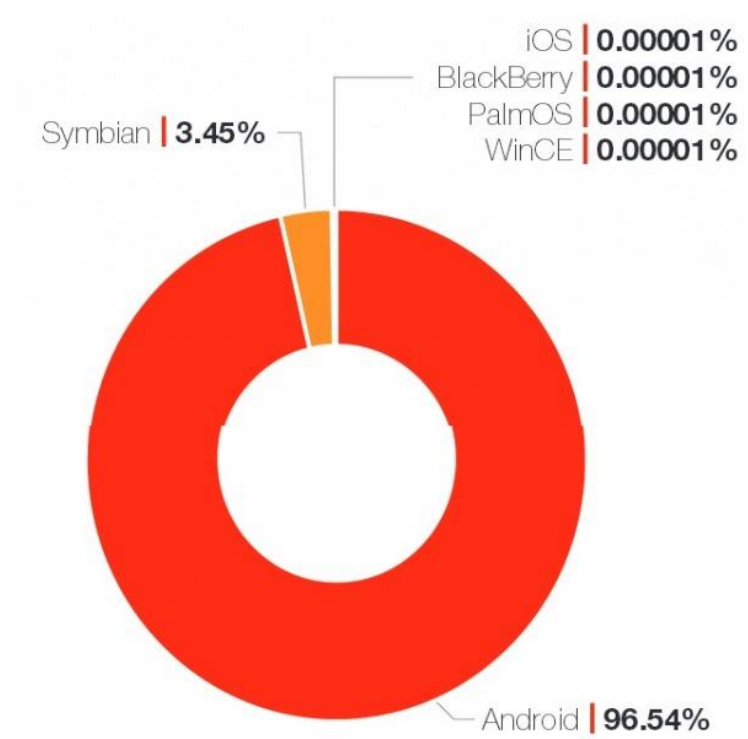
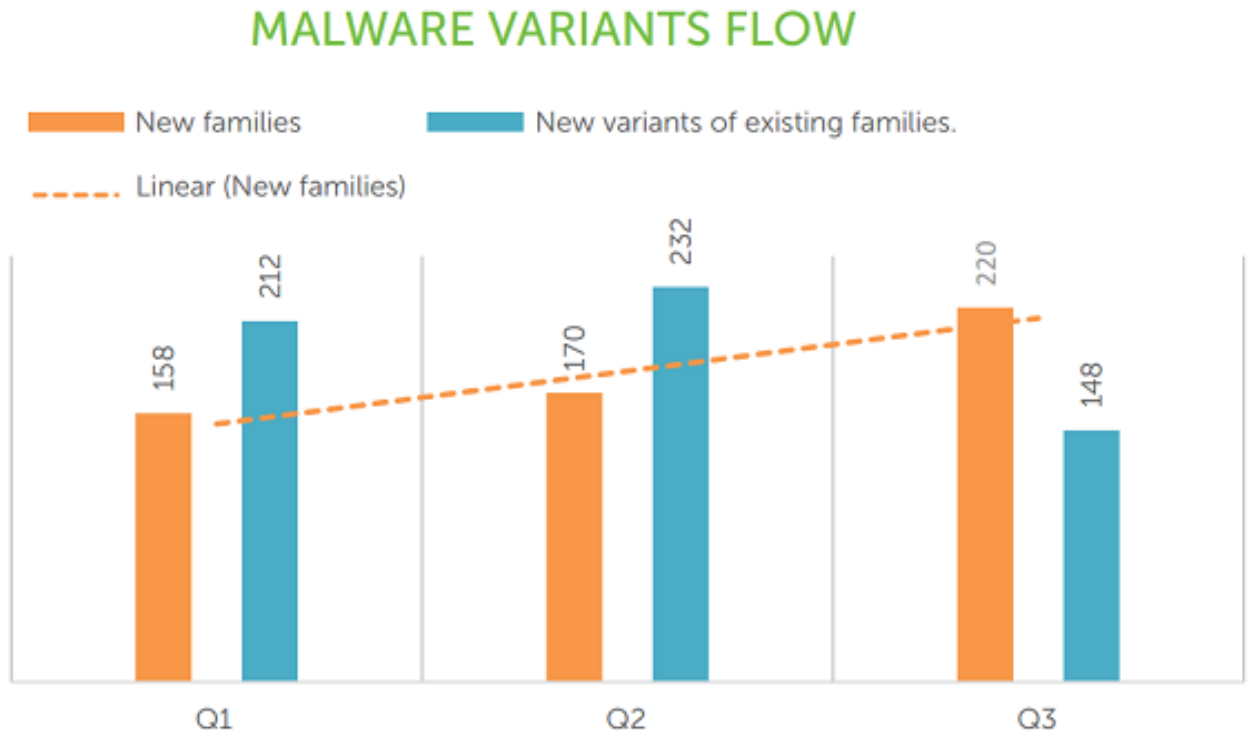
Malware Analysis III

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

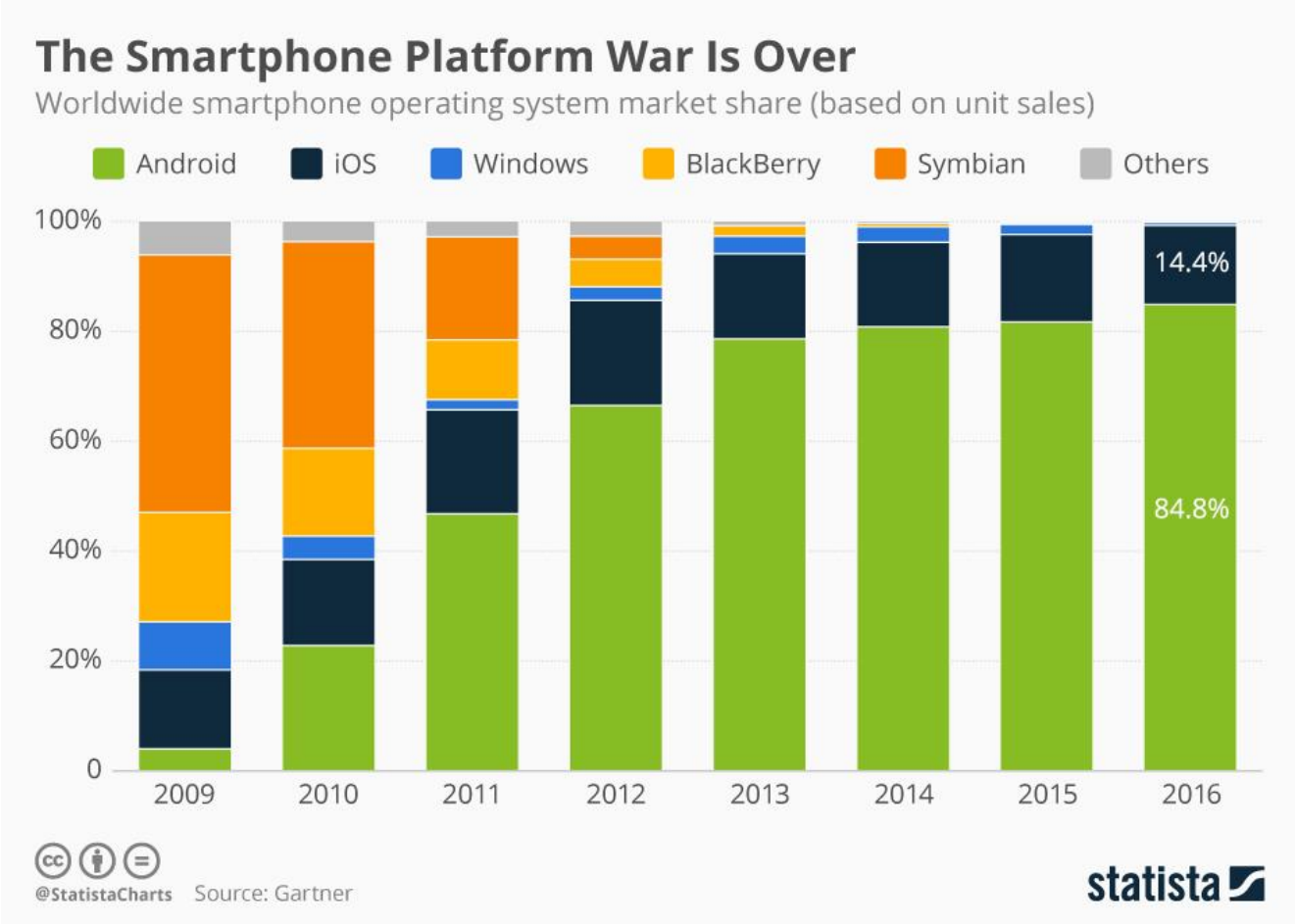
Formal Methods for Secure Systems, University of Pisa

Target of mobile attack





The reason why



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

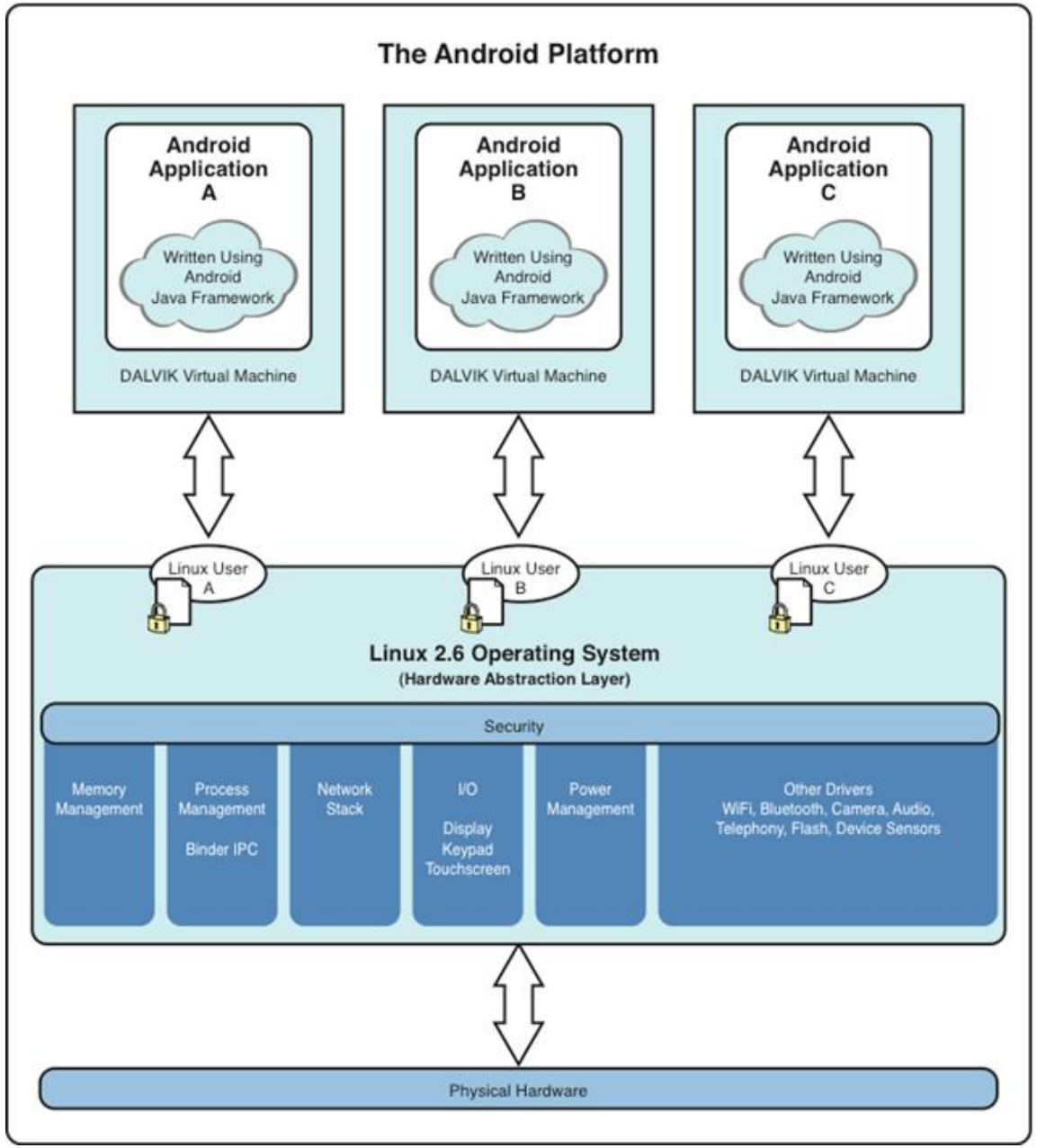
- Sandboxing (Isolation)
  - Virtual Machine
  - Linux Kernel
- Access Control
  - Permission System
- Storage separation
  - Possible for internal memory (ext3)
  - Not possible for SDCard (fat32)

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.


The Android Platform
Android Application A
Written Using Android Java Framework
DALVIK Virtual Machine
Android Application B
Written Using Android Java Framework
DALVIK Virtual Machine
Android Application C
Written Using Android Java Framework
DALVIK Virtual Machine
Linux User A
Linux User B
Linux User C
Linux 2.6 Operating System
(Hardware Abstraction Layer)
Security
Memory Management
Process Management
Binder IPC
Network Stack
I/O
Display
Keypad
Touchscreen
Power Management
Other Drivers
WiFi, Bluetooth, Camera, Audio, Telephony, Flash, Device Sensors
Physical Hardware

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:

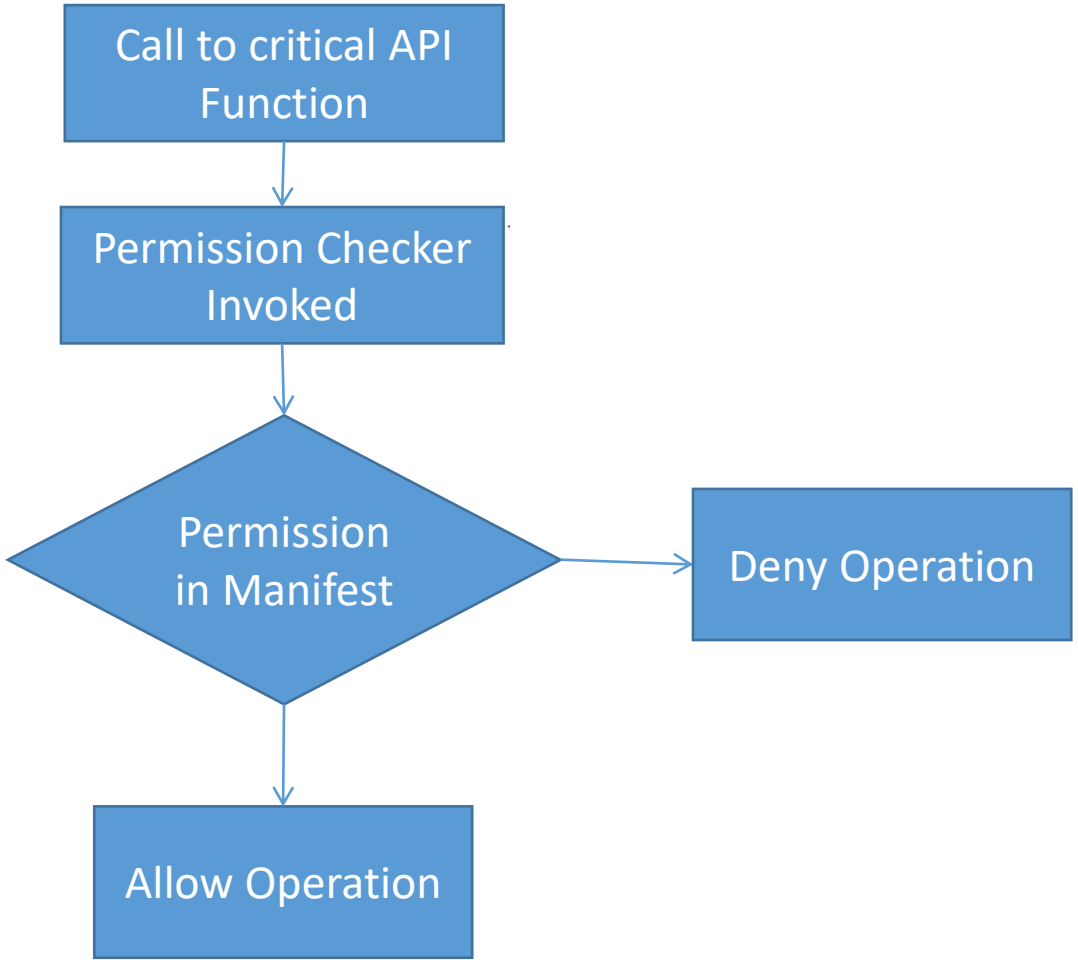
<uses-permission android:name="*string*" />

- 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

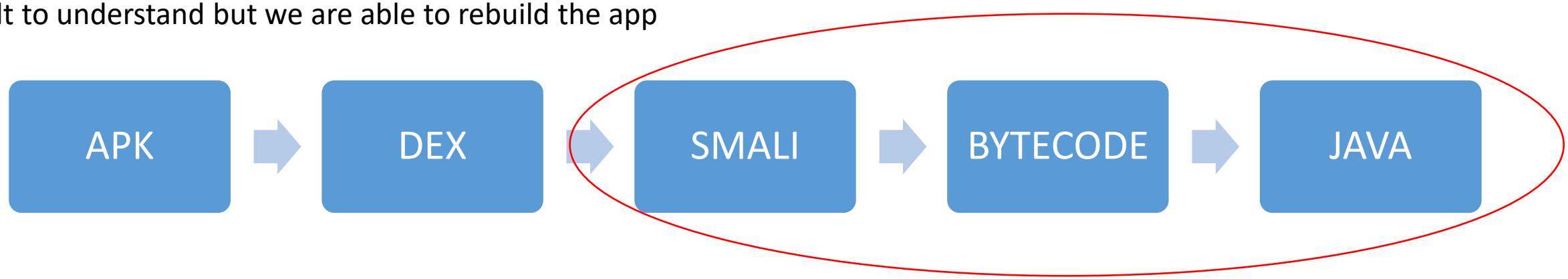
- Virtual Environment to check if app is malicious
- Runs the app in a phone like environment for around 5 mins before publishing
- Detects most of the known malware…
- Can be bypassed easily

Android application

- APKs file

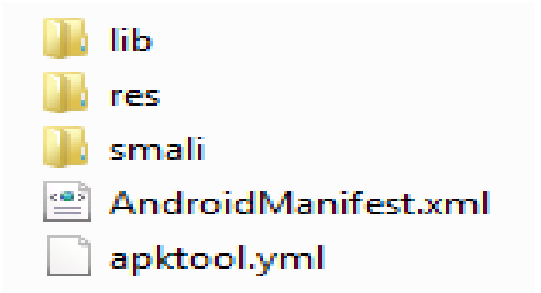
APP representations

Difficult to understand but we are able to rebuild the app



Easy to understand but we are not 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: Decoding file-resources...
I: Decoding values */* XMLs...
I: Done.
I: Copying assets and libs...
```

Bytecode Viewer

- Java source code/ Java Bytecode visualizer

```
java -jar Bytecode-Viewer-2.9.16.jar
```