Android Application Pentesting Workshop 1

Jakub Kaluzny, SecTalks – 13.06.2017

whoami

@j_kaluzny: Security Consultant at The Missing Link Security, Sydney

Previously testing AppSec in Europe: payments systems, fraud detection, stocks, enterprise software

Sometimes I have interesting projects:

- Bypassing JS-based malware detection (MiTB)
- RE of proprietary network protocols
- Enterprise solutions: Hadoop, HPaaS
- Voice biometrics

Tools

- Android SDK (emulator), API 15
- Apktool
- D2j
- jd-gui
- Android MobSF
- QARK
- sqlite3
- Local proxy (e.g. ZAP, BurpSuite)

Intro

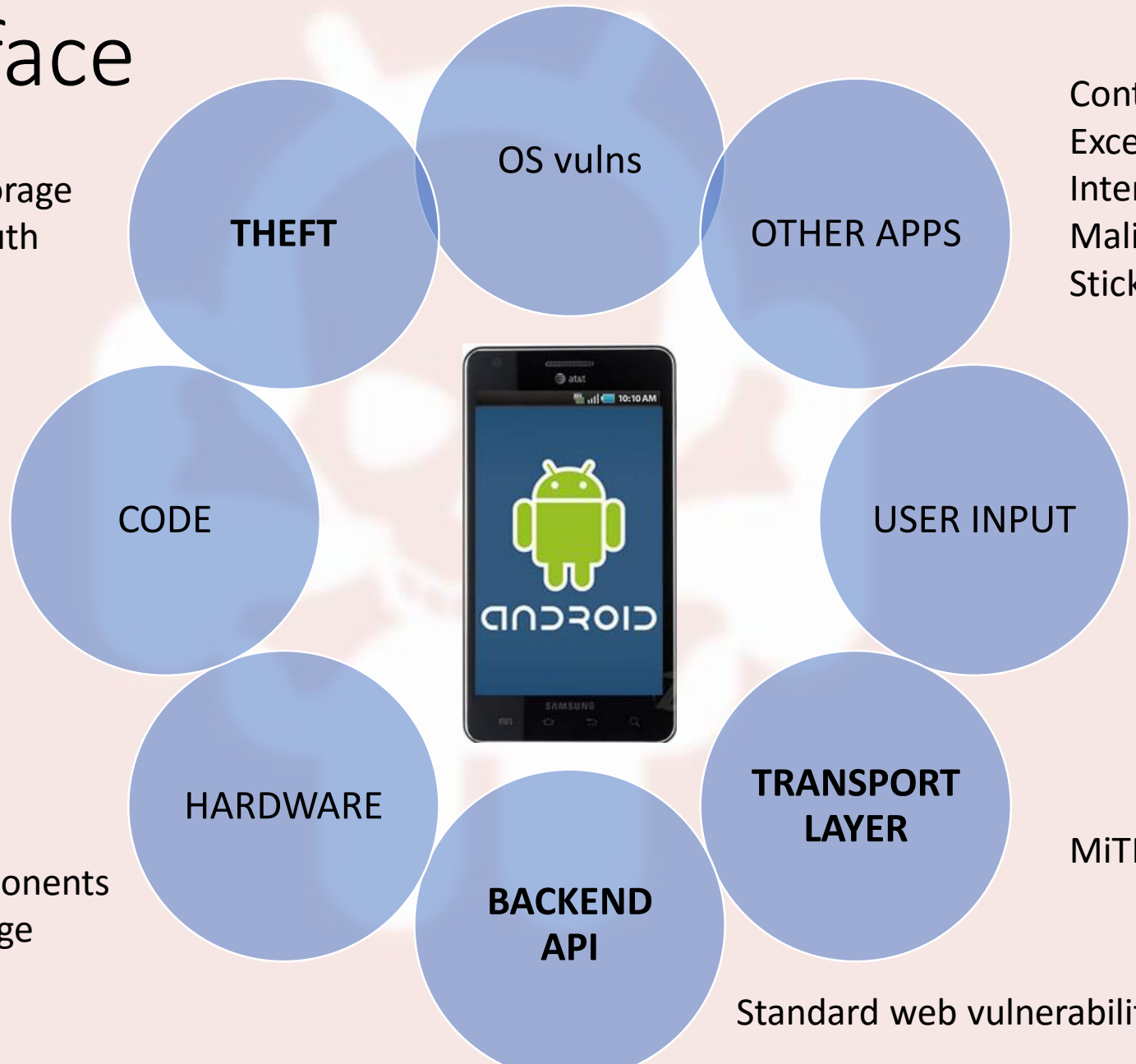
- 2 bln mobile banking users
- 4 mln mobile apps, 25% of them are malicious

- Pentesting mobile apps requires to set up a lab
- A lot of low hanging fruits
- Mobile app developers are not familiar with attack vectors

So, how to attack mobile apps?

Agenda

Attack surface



Insecure storage
Local PIN auth

Container access
Excessive intents
Intent spoofing
Malicious intent receipt
Sticky broadcast

Excessive information
Token leakage

Local SQL Injection
XSS in WebView
CAPTCHA bypass

Relying on 3rd party components
Fingerprints, secure storage

MiTM

Standard web vulnerabilities

OS vulnerabilities

- Multiple root techniques in the wild
- Mostly via malicious application
- But: *Over The Air: Exploiting Broadcom's Wi-Fi Stack*
- KingRoot - https://kingroot.net/
- Each of those is worth > $100k

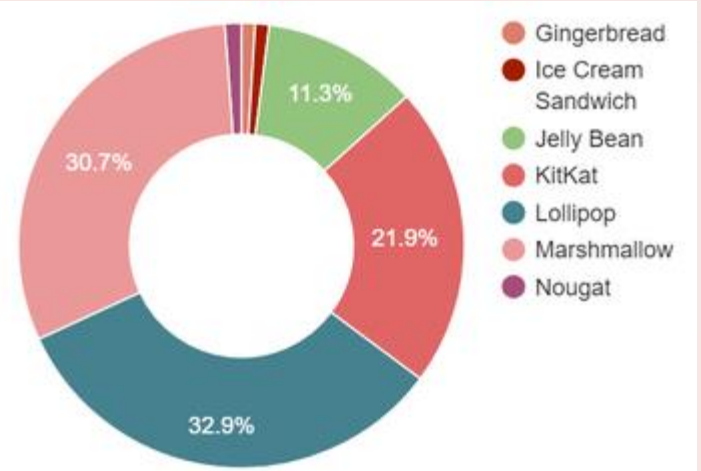
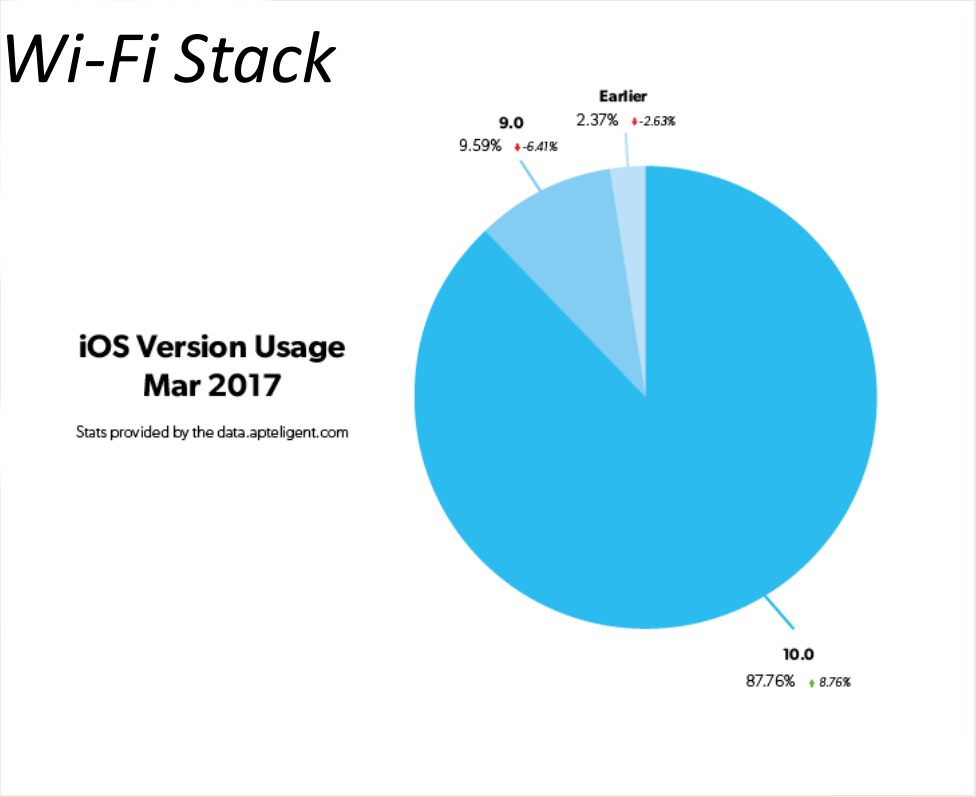


Image Source: Global Android Version Distribution (February 2017), Wikipedia



ADB

- Root the device
- Enable USB debugging
- Enable APK installation from “untrusted sources”
- Connect with ADB

- Some apps will detect root
- HideMyRoot, RootCloak

ADB with Android emulator

- Download Android SDK
- Create AVD
- Run AVD
- Connect with ADB

DEMO

- Some apps will detect emulator

Static analysis

Quick look at APK

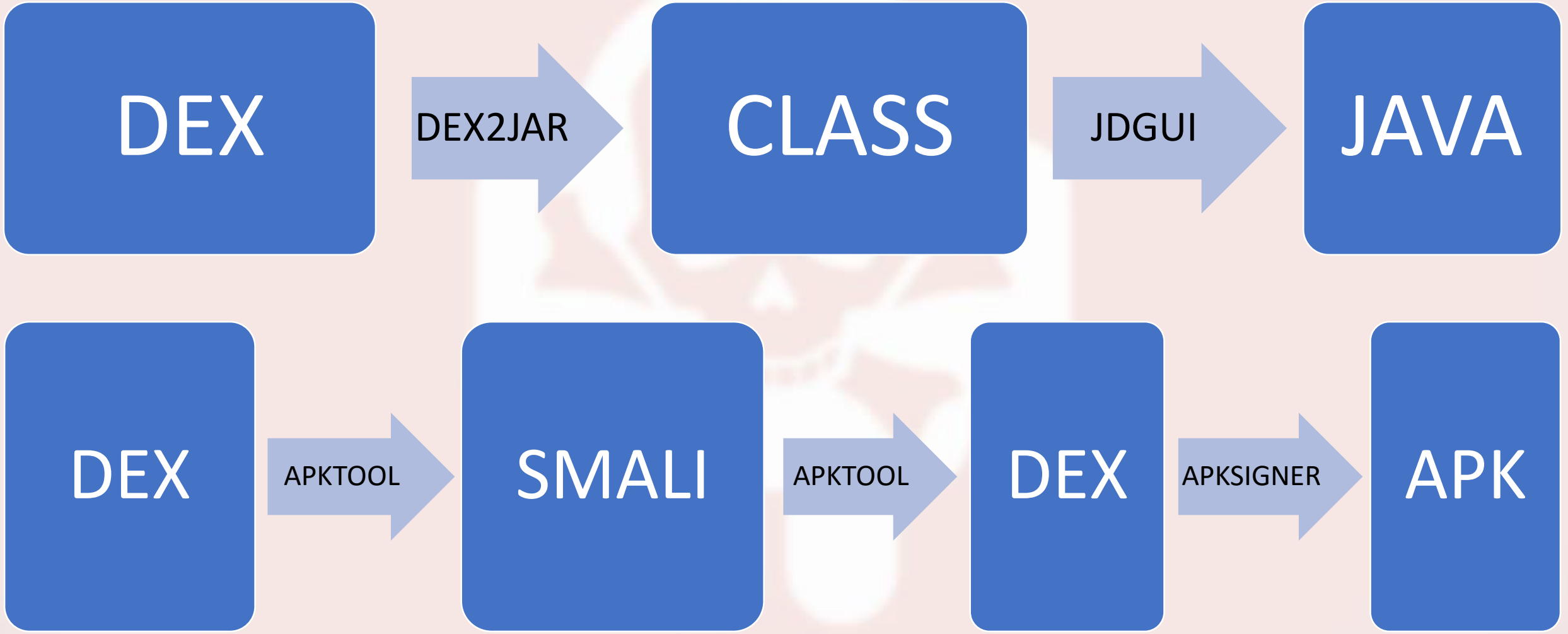
- It can be pulled from a device :

```
adb shell pm list packages
adb pull /data/app/com.example.someapp-2.apk
```

- There are online services to download APK, e.g. APKpure.com (beware of malware)
- You can sniff the traffic to Google Play and collect it from local proxy
- It’s a ZIP file!

DEMO

What to do with DEX?



Demo – what can you find in APK?

~~`unzip file.apk`~~

```
apktool decode file.apk # manifest, resources,
views, templates and smali code
```

DEMO

Decompiling DEX to Java

Convert to JAR and decompile

```
d2j-dex2jar file.apk # output is .jar
jd-gui file.jar
```

DEMO

Sometimes the source is obfuscated

DEMO

Recompiling APK

```
apktool d -f file.apk -o smali
# change smali code
apktool b -f smali/ -o unaligned.apk
# sign the apk
jarsigner -verbose -sigalg MD5withRSA -digestalg
SHA1 -keystore ~/.android/debug.keystore  -storepass
android  unaligned.apk  androiddebugkey
zipalign -v 4 unaligned.apk smali.apk # if necessary
```

DEMO

Automated static analysis

- OWASP: Android Mobile Security Framework
- QARK: Quick Android Review Kit

DEMO

What should you look for?

- OWASP MASVS – Mobile Application Security Verification Standard
- OWASP Mobile Top 10

- Any vulnerability that can cause damage

DEMO

Static analysis is not enough

Dynamic analysis

- Logs:

```
adb logcat
```

DEMO

- File diff:

```
adb pull /data/data/package/
```

DEMO

- Databases:

```
sqlite3 cache.db
```

DEMO

Traffic proxy

- Wireshark + socat if non-HTTP
- HTTP local proxy (ZAP, BurpSuite):

```
emulator -avd myavd -http-proxy http://168.192.1.2:8080
Real device: connect it to the same Wi-Fi
```

DEMO

- Install SSL Certificate

```
adb push cert.crt /sdcard/
```

DEMO

SSL MiTM

- Accepts any certificate
- Accepts a trusted certificate issued to other domain
- Weak pinning – e.g. only chain length and CN fields are checked

DEMO

SSL pinning

- Certificate may be hardcoded in APK resources / code
- Simply change it and recompile the app for the purpose of testing

Proxying Android 7.0

res/xml/network_security_config.xml

```
<network-security-config>
      <base-config>
            <trust-anchors>
                <!-- Trust preinstalled CAs -->
                <certificates src="system" />
                <!-- Additionally trust user added CAs -->
                <certificates src="user" />
           </trust-anchors>
      </base-config>
 </network-security-config>
```

Thank you

@j_kaluzny