Source Analysis of Security Smells in Android Apps

BSc Thesis - Presentation #1

Patrick Frischknecht

28.11.2017
Agenda

1) Relevance of mobile security
2) Android security smells
3) Bachelors thesis
   - Familiarization with state-of-the-art
   - Implementation of detectors
   - Evaluation
Relevance of Android Security

➢ 85% of the smartphone market [1]
➢ More than three million apps [2]

Issues emerge:

➢ Privacy
➢ Data sensitivity
  ▪ E-commerce
  ▪ E-banking
  ▪ Healthcare
Android Security Smells

Symptoms in the code that indicate the prospect of security and privacy vulnerabilities

[03]
Android Security Smells #2

➢ Exposed Persistent Data
  using MODE_WORLD_READABLE/WRITEABLE

➢ Dynamic Code Loading
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
I will NOT use insecure and outdated APIs any longer!
Ambitions

1) Creation of an easy to use IDE tool

2) Detection of a security smells subset

3) Evaluation on a set of Android projects
Project Plan

- Existing tools evaluation
  - Android Lint
  - Findbugs Security
  - Amandroid
  - ...

- Selection and extension of a tool

- Automated analysis on a larger set of Android projects
Tool: Findbugs Security
Tool: Android Lint

- Static source code analysis *tool for Android*
- Integrated in *Android Studio*
- Provides *a lot of built in security checks*
  - 14 different security smells checks built-in
  - Especially manifest smells are well covered
Android Lint: In Action

- Code highlighting
- Tooltips
- Quickfixes
Android Lint: Extension

- Creation of a new detector class
  - XML or JavaDetector

- Registration in a registry class

- JAR repackaging and copying to Lint folder
Android Lint: Example
Weak Hash Function Detector

➢ Smell: Weak Crypto Algorithm

➢ Finds usages of MD5 hash function

*MD5 is vulnerable to collision attack*
New JavaScanner looking for `MessageDigest.getInstance("MD5")`
Android Lint: Example
Weak Hash Function Detector

private static void checkRequestedHashFunction(@NonNull JavaContext context,
                                               @NonNull MethodInvocation call, @NonNull Node node) {
    if (WEAK_HASH_FUNCTIONS.contains(value)) {
        String message = """+value+" is considered a weak hash function and should
context.report(ISSUE, call, context.getLocation(node), message);""";
    }
}

➢ Adds a report
Android Lint: Example
Weak Hash Function Detector

```java
java.security.MessageDigest md5Digest = null;
try {
    md5Digest = java.security.MessageDigest.getInstance("MD5");
} catch (NoSuchAlgorithmException e) {
    // MD5 is considered a weak hash function and should not be used in a security critical context
}
```

- Visible in Android Studio
- ... and in the HTML report
Android Lint: Evaluation

- Execution on a subset of Android open-source projects
- Manual execution
- Fast & Scalable!
Roadmap

➢ Investigation of the relation between security smells and required detectors

➢ Implementation of more detectors

➢ Expansion of the Android app test set
References

https://www.idc.com/promo/smartphone-market-share/os

https://www.appbrain.com/stats/number-of-android-apps

Android Lint: Issues

➢ Only plain static analysis
  ▪ no taint analysis
  ▪ no dynamic analysis

➢ Android Lint API is unstable

➢ Projects must be compiled
  (some detectors require class files)

➢ Executes on source code, thus we need
  ▪ Open-source projects
  ▪ ... or decompiled byte-code
Android Lint: Preliminary Results

![Bar chart showing instances of issues in Android apps]

- InsufficientRSAKeySize
- WeakHashFunction
- HardwareIds
- TrustAllX509TrustManager

Source Analysis of Android Apps