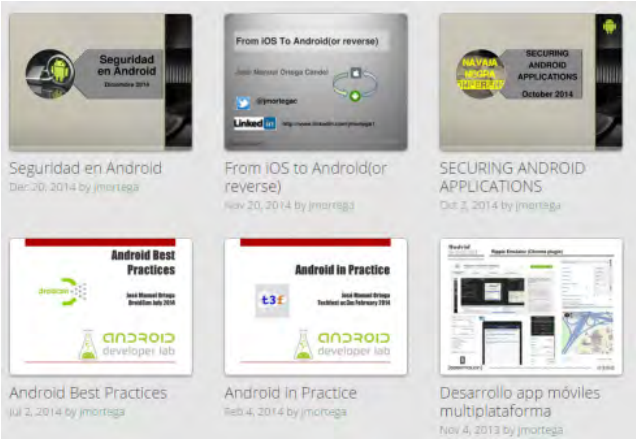
Security testing in mobile applications

José Manuel Ortega Candel

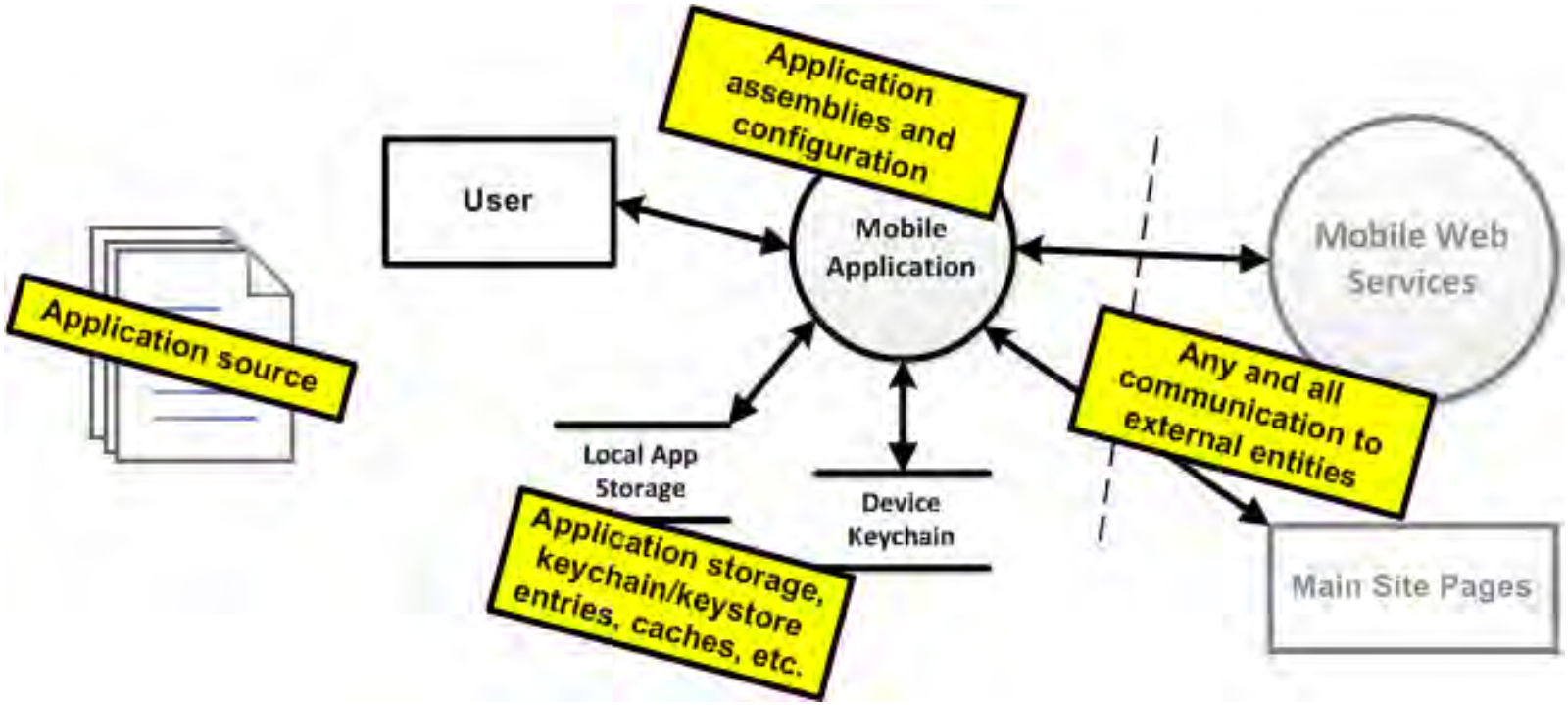
About me

- Centers Technician at Everis
- Computer engineer by Alicante University
- Frontend and backend developer in Java/J2EE
- Speaker site with some presentations in mobile
  - https://speakerdeck.com/jmortega



Index

Security Testing for Mobile Apps



Security Testing for Mobile Apps

White Box Testing
Static Analysis Code

Black Box Testing
Dinamyc Analysis at Runtime

Static Application Security Testing

Source code Review

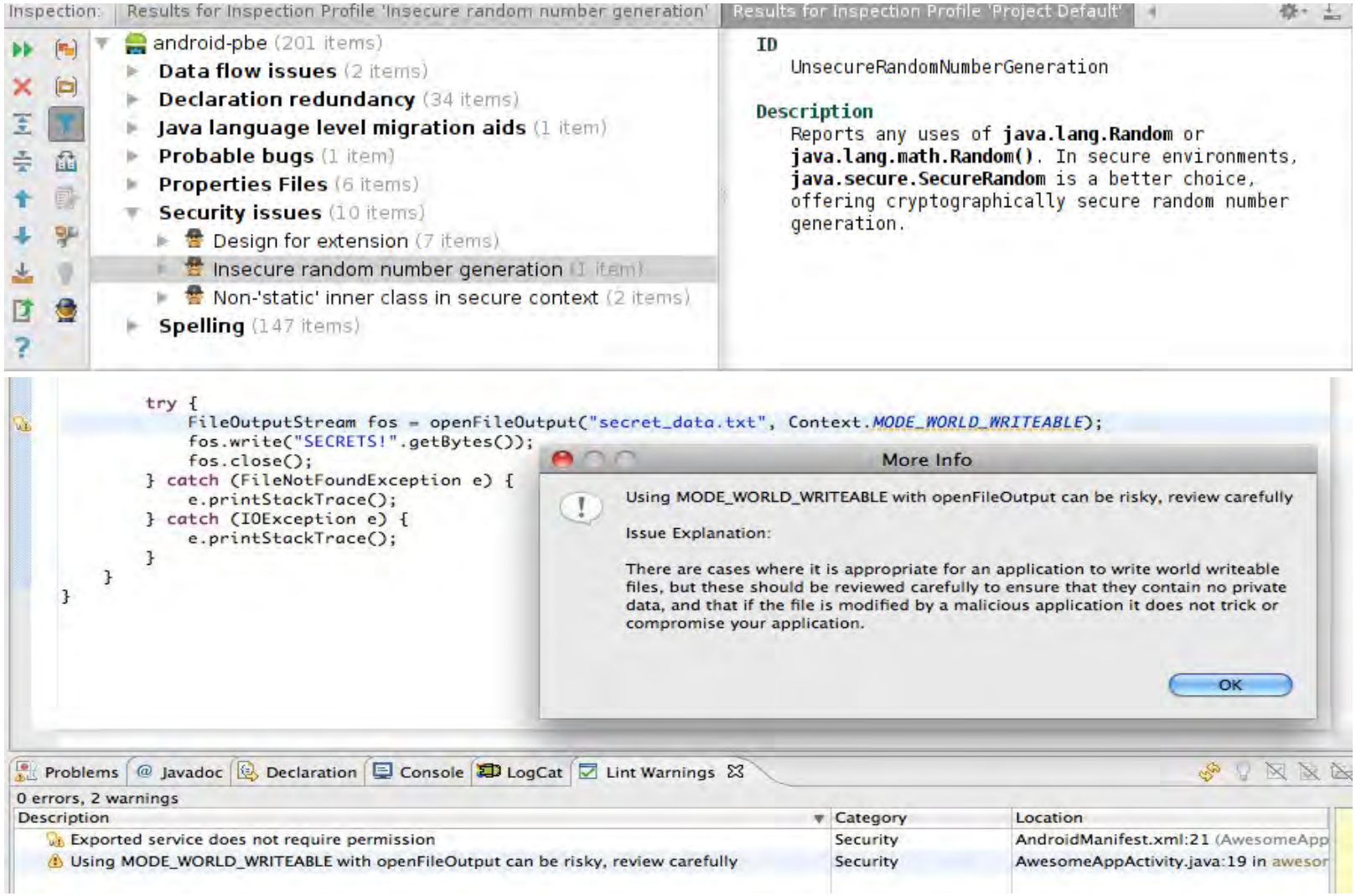
Reverse engineering

Lint for Android Studio

Sonnar Plugins for mobile

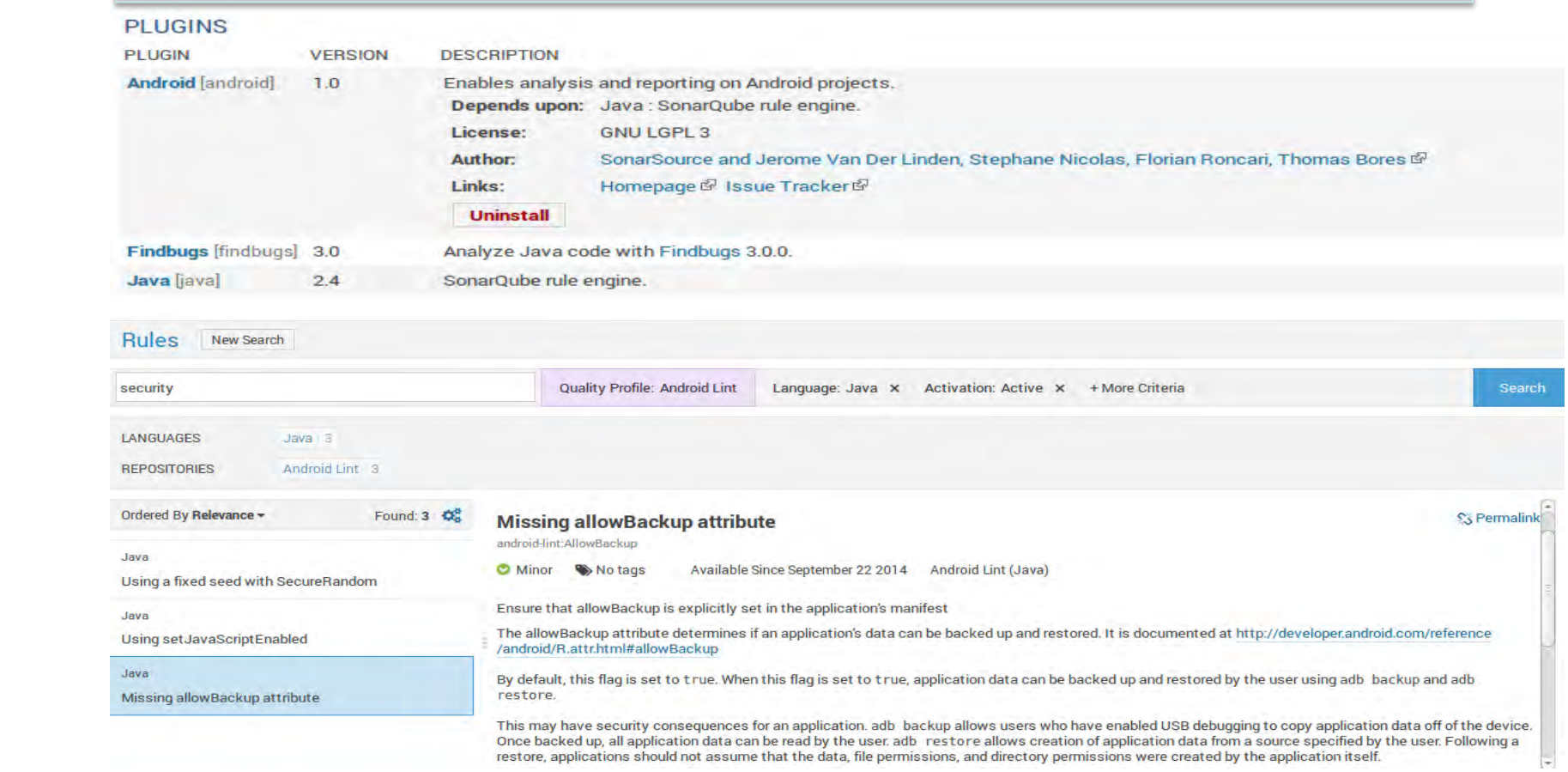
http:// www.sonarqube.org

Static Application Security Testing



Static Application Security Testing

https://github.com/SonarCommunity/sonar-android



Static Application Security Testing

http://sourceforge.net/projects/agnitiotool

You can decompile APKs and search calls dangerous functions.

Num...	Principle/s	Questions	Yes	No	N/A
62	Auditing and Logging	Are logs correctly sanitised to ensure that sensitive data is not logged in clear text by the application?	☐	☐	☐
63	Auditing and Logging	Are all application errors logged to a central log server?	☐	☐	☐
64	Auditing and Logging	Does the design identify the level of auditing & logging needed and identify key parameters to be logged & audited?	☑	☐	☐
65	Auditing and Logging	Does the design ensure that successful and unsuccessful attempts to execute privileged actions are logged?	☑	☐	☐
66	Auditing and Logging	Does the design ensure that the content of audit logs meet the requirements of company/regulatory standards?	☐	☐	☐
78	Auditing and Logging	Does the application have non-repudiable logs for users access to paid resources?	☐	☑	☐
15	Authenticatior	Are application trust boundaries identified on the data flow diagrams in the design document?	☐	☑	☐
16	Authenticatior	Does the design identify the identities that are used to access resources across all trust boundaries?	☑	☐	☐
17	Authenticatior	Does the application require authentication for all resource access attempts except for publicly accessible resources?	☑	☐	☐

ation profile | View application profiles | Reporting | The Principles | Checklist guidance | Static analysis

Your code will be shown below

```
    // Encrypt
    SecretKey secret = new SecretKeySpec(tmp.getEncoded(), "AES");
    Cipher encryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM);
    IvParameterSpec ivspec = new IvParameterSpec(initVector);
    encryptionCipher.init(Cipher.ENCRYPT_MODE, secret, ivspec);
    encryptedText = encryptionCipher.doFinal(cleartext.getBytes());

    // Encode encrypted bytes to base64 text to save in text file
    result = base64.encodeToString(encryptedText, base64.DEFAULT);

  } catch (Exception e) {
    e.printStackTrace();
  }

  return result;
}

// Password based encryption
// Decode the provided base64 string, then decrypt based on the provided password
// Return decrypted cleartext string
public String decrypt(String encryptedText, String password) {
  String result = "";

  try {

      // Decode base64 text back to encrypted bytes
        byte[] toDecrypt = base64.decode(encryptedText, base64.DEFAULT);
```

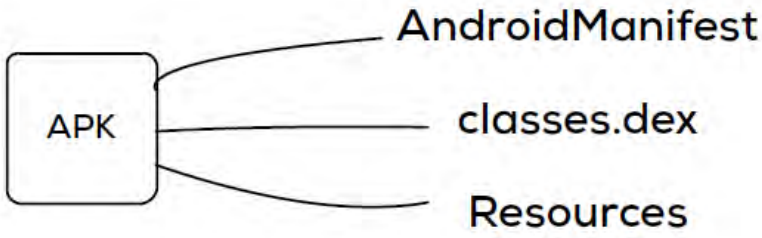
Findings and linked checklist items

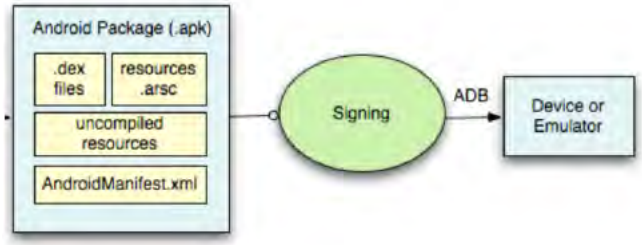
If cryptography is used then a strong cipher should be used such as AES, 3DES or SHA-256.

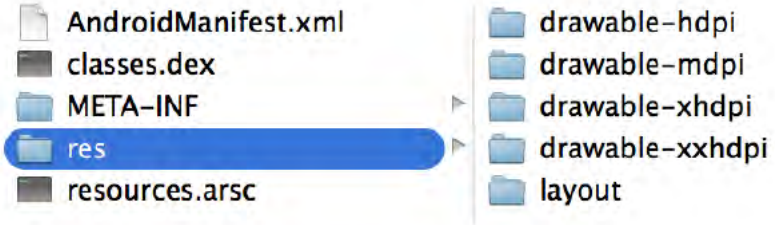
You must also determine what size key is used, the larger the better. Where is hashing performed? Are password that are being persisted hashed? How are random numbers used in cryptographic methods generated? Is the PRNG sufficiently random?

Questions
Does the application secure secrets that shouldn't be visible to the users with server side functions?
Does the application use salts when hashing passwords?
Does the design identify how encryption keys and salts will be securely stored?
Does the design identify the key rotation policy for the application?
Does the design specify how the encryption keys for this application are stored securely?

Reverse engineering on Android







APK Tool /Dex2jar/Java Decompiler

```
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /Users/filip/Library/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...
```

Static analysis on iOS

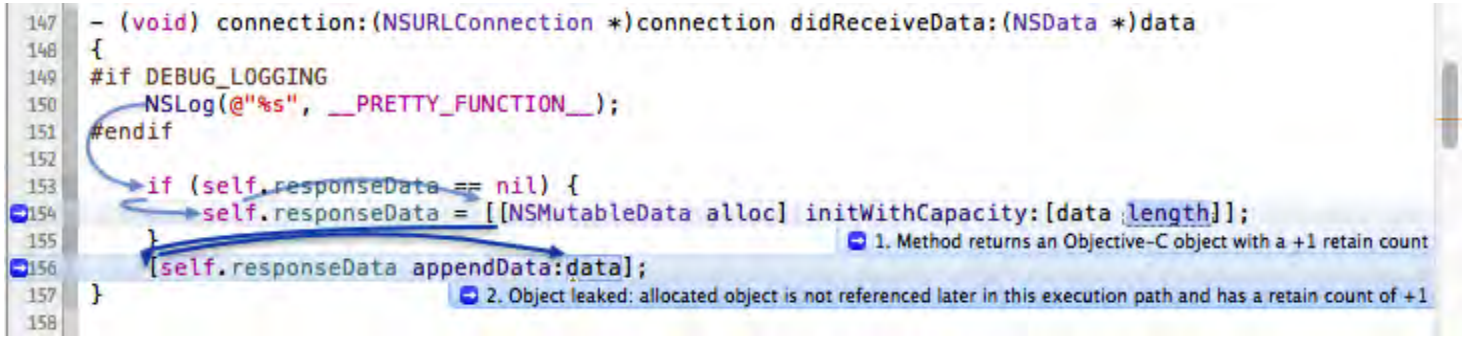
Xcode Development environment

Apple Developer Tools

'otool' command provided by XCode can be used to get information from iOS application binaries and can be used to support security analysis.

Static analysis on iOS

Detect memory leaks with XCode



Clang Static Analyzer
http://clang-analyzer.llvm.org

Flawfinder(Security weakness in C/C++)
http://www.dwheeler.com/flawfinder

Dinamyc Application Security Testing

Analyze Network Traffic at runtime

Analyze Remote Services

DroidBox for Android Instruments for iOS

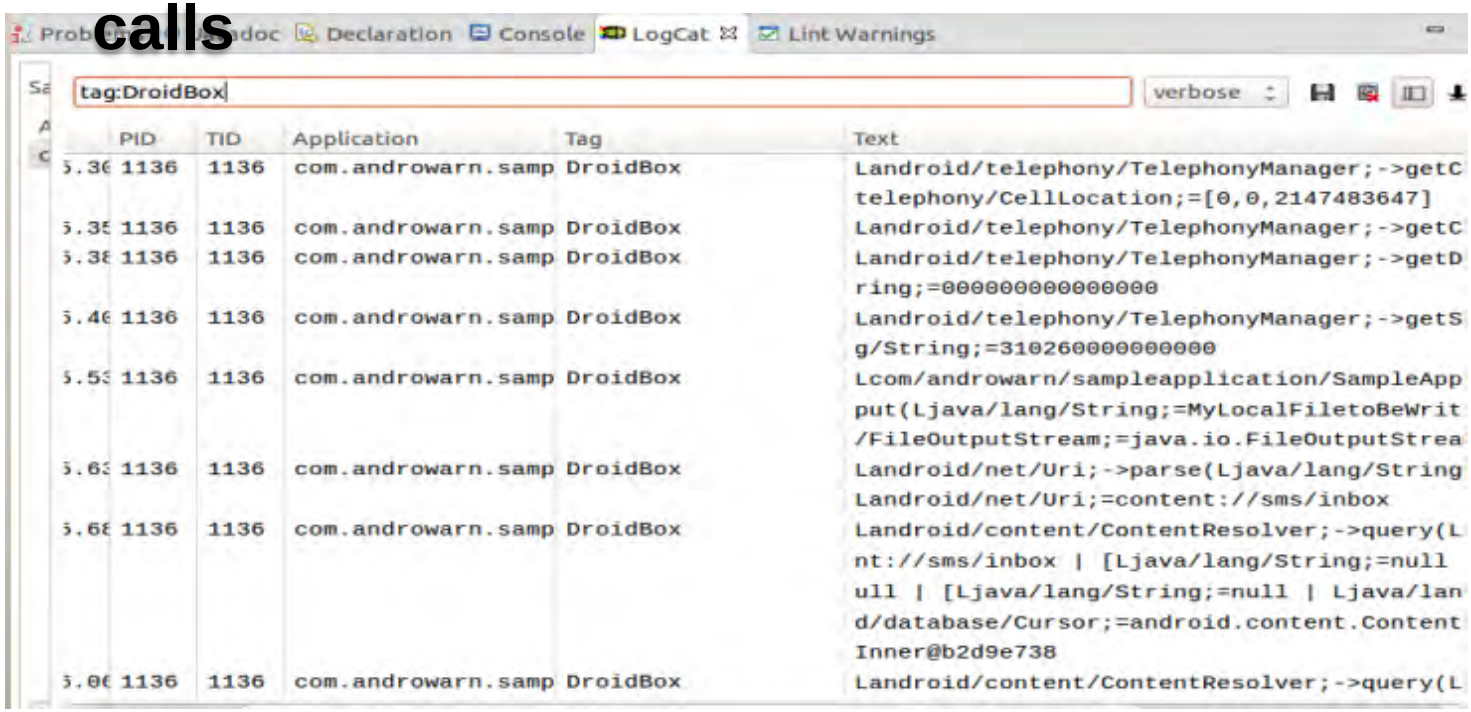
Discovering logical vulnerabilities

Dinamyc Application Security Testing

- https://code.google.com/p/droidbox

Monitoring Actions

- ❑ Information leaks — Network IO and File IO
- ❑ Cryptography operations — SMS and Phone calls



Dinamyc Application Security Testing

```
[File activities]
----------------
   [Read operations]
   ----------------
      [22.9400451183]     Path: /data/data/droidbox.tests/files/myfilename.txt^A
              Data: Write a line
      [24.2107310295]     Path: /data/data/droidbox.tests/files/myfilename.txt
              Data:
      [25.997330904]    Path: /data/data/droidbox.tests/files/output.txt
              Data: null
      [26.781430006]    Path: /data/data/droidbox.tests/files/output.txt
              Data:
   [Write operations]
   -----------------
      [21.3330090046]     Path: /data/data/droidbox.tests/files/myfilename.txt^A
              Data: Write a line
      [21.3614990711]     Path: /data/data/droidbox.tests/files/output.txt
              Data: null
[Crypto API activities]
----------------------
      [26.8029410839]     Key:{0, 42, 2, 54, 4, 45, 6, 7, 65, 9, 54, 11, 12, 13, 60, 15} Algorithm: AES
      [26.811686039]    Operation:{encryption} Algorithm: AES
              Data:{357242043237517}
      [26.818600893]    Key:{0, 42, 2, 54, 4, 45, 6, 7, 65, 9, 54, 11, 12, 13, 60, 15} Algorithm: AES
      [26.8250999451]     Operation:{decryption} Algorithm: AES
              Data:{357242043237517}
      [26.8305909634]     Key:{0, 42, 2, 54, 4, 45, 6, 8} Algorithm: DES
      [26.8399989605]     Operation:{encryption} Algorithm: DES
              Data:{357242043237517}
      [26.8453080654]     Key:{0, 42, 2, 54, 4, 45, 6, 8} Algorithm: DES
      [26.853967905]    Operation:{decryption} Algorithm: DES
              Data:{357242043237517}
[Network activity]
-----------------
```

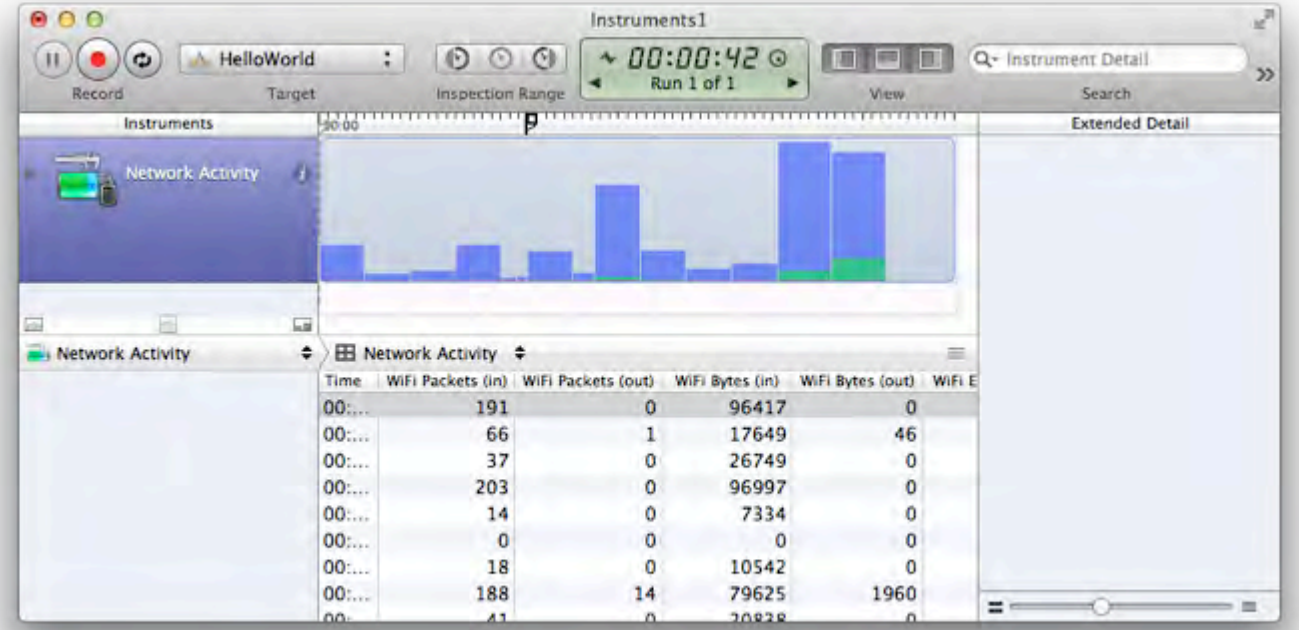
Instruments for iOS applications

File Activity Monitoring

Network Monitoring

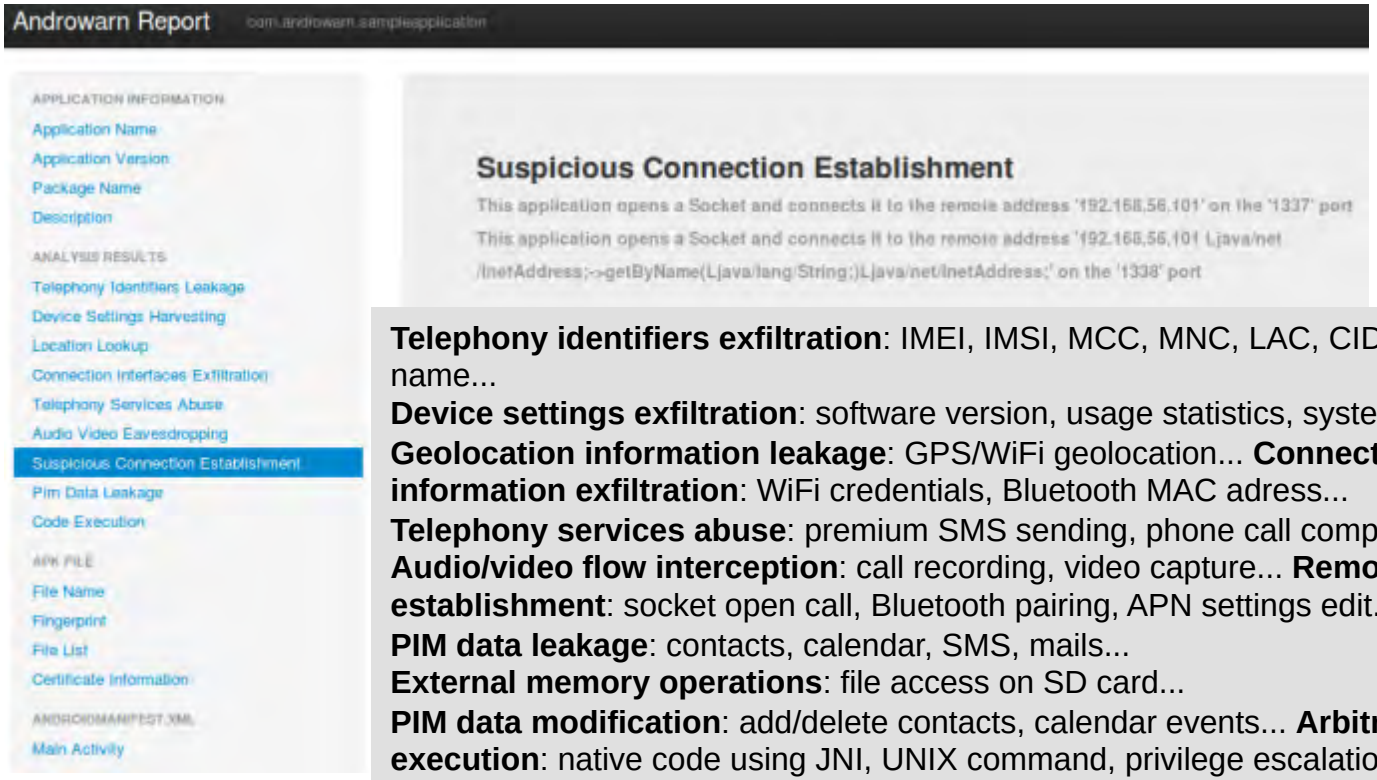
Memory Monitoring

Process Monitoring



Application Security Analyser

python androwarn.py -i my_apk.apk -r html -v 3



Telephony identifiers exfiltration: IMEI, IMSI, MCC, MNC, LAC, CID, operator's name...
Device settings exfiltration: software version, usage statistics, system settings, logs...
Geolocation information leakage: GPS/WiFi geolocation... **Connection interfaces information exfiltration**: WiFi credentials, Bluetooth MAC adress...
Telephony services abuse: premium SMS sending, phone call composition...
Audio/video flow interception: call recording, video capture... **Remote connection establishment**: socket open call, Bluetooth pairing, APN settings edit...
PIM data leakage: contacts, calendar, SMS, mails...
External memory operations: file access on SD card...
PIM data modification: add/delete contacts, calendar events... **Arbitrary code execution**: native code using JNI, UNIX command, privilege escalation...
Denial of Service: event notification deactivation, file deletion, process killing, virtual keyboard disable, terminal shutdown/reboot...

Risk Management

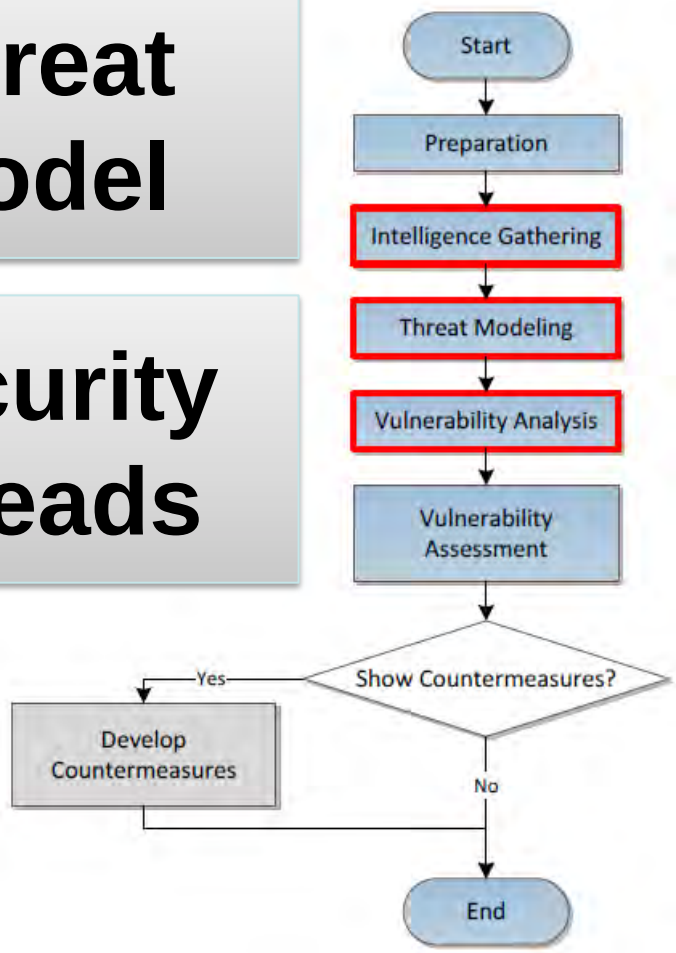
Intelligence Gathering

Threat model

Vulnerabilities

Security threads

OWASP Security risks



Intelligence Gathering

Environmental Analysis

Analyze internal processes and structures

Architectural Analysis

App [network interfaces, used data, communication with other resources, session management]

Runtime environment [MDM, jailbreak/rooting, OS version]

Backend services [application server, databases]

Intelligence Gathering

What type of device is it?

Determine Operating System version

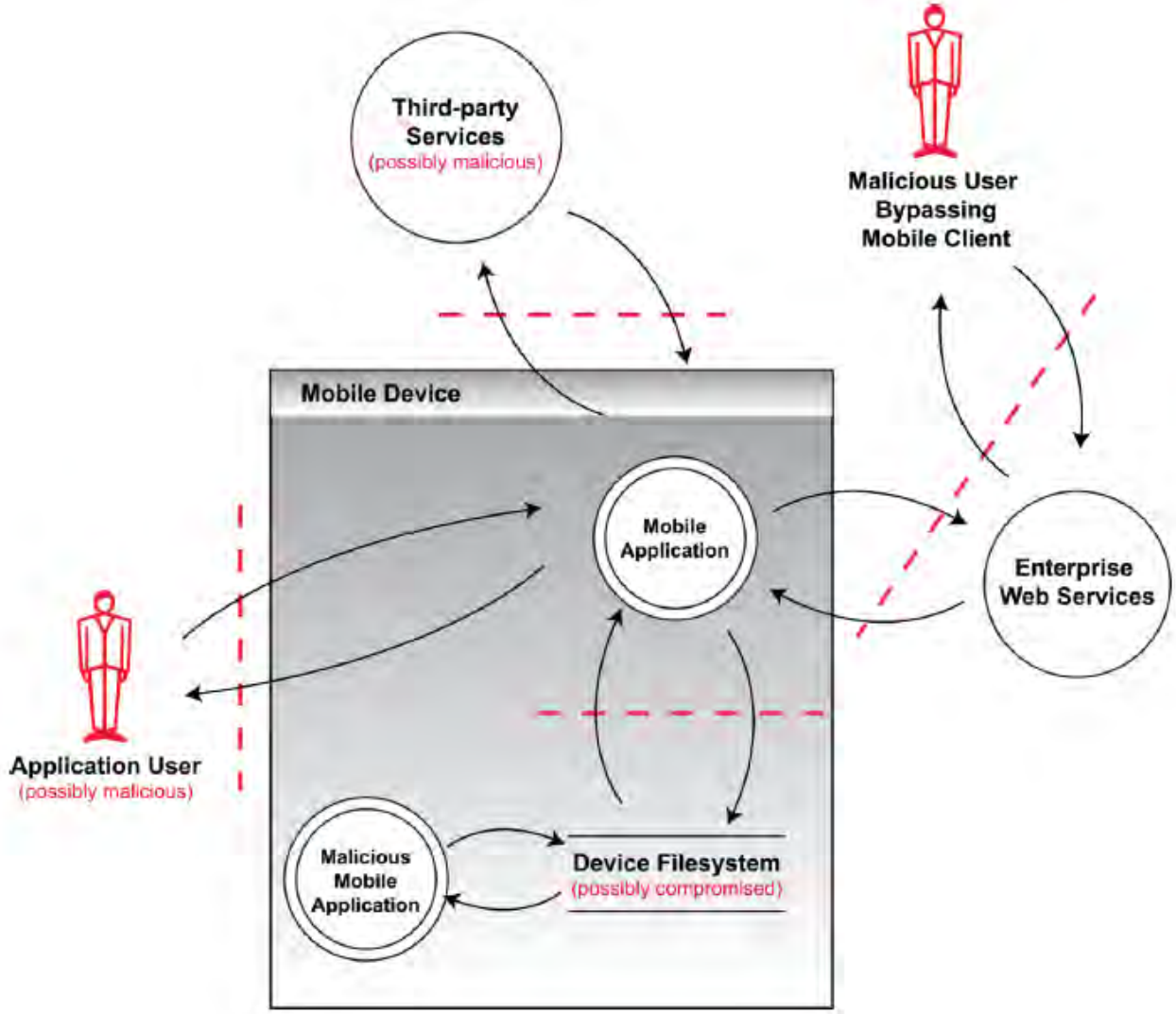
Is the device already rooted?

Is the device passcode enabled?

What key applications are installed?

Is the device connected to network?

Thread model in Mobile applications



Functional Security threads

- Authentication
- Cryptography
- Data Protection
- Input Validation
- Access Control
- Communication Security
- Session Management
- Error Handling and Logging

Security issues

Malicious Applications
- Rooting Exploits
- SMS Fraud
- Rapid Malware Production

Dynamic Analysis
- Sandbox
- Real-time Monitoring
- Mobile Specific Features

Static Analysis
- Permissions
- Data Flow
- Control Flow

Browser Attacks
- Phishing
- Click Through

Mobile Botnets
- Epidemic Spread
- Attacking Network Services
- Tracking Uninfected Devices

User Education
- Ignoring Permissions
- Phishing
- Improperly Rooting Devices
- Alternative Markets

Vulnerabilities

Third party libraries

Multiplaftorm libraries for hybrid apps

Components

WebView [JavaScript+Cache]

SQLite DataBase

Shared Preferences

Vulnerabilities

1. Activity monitoring and data retrieval
2. Unauthorized dialing, SMS, and payments
3. Unauthorized network connectivity (exfiltration or command & control)
4. UI Impersonation
5. System modification (rootkit, APN proxy config)
6. Logic or Time bomb
7. Sensitive data leakage (inadvertent or side channel)
8. Unsafe sensitive data storage
9. Unsafe sensitive data transmission
10. Hardcoded password/ keys

Vulnerabilities Analysis

Static methods	**Automatic and manual source code analysis**
	Reverse Engineering
Dynamic methods	**Network monitoring and trafic analyzing**
	Runtime analysis
Forensic methods	**Log analysis**
	File permission analysis
	File content analysis

Dynamic methods tools

Network monitoring and traffic analyzing	Wireshark, BurpSuite
Runtime analysis	iOS GNU debugger, Snoop-it, Cycript
	Mercury, Intent Sniffer, Intent Fuzzer
File analysis	androidAuditTools

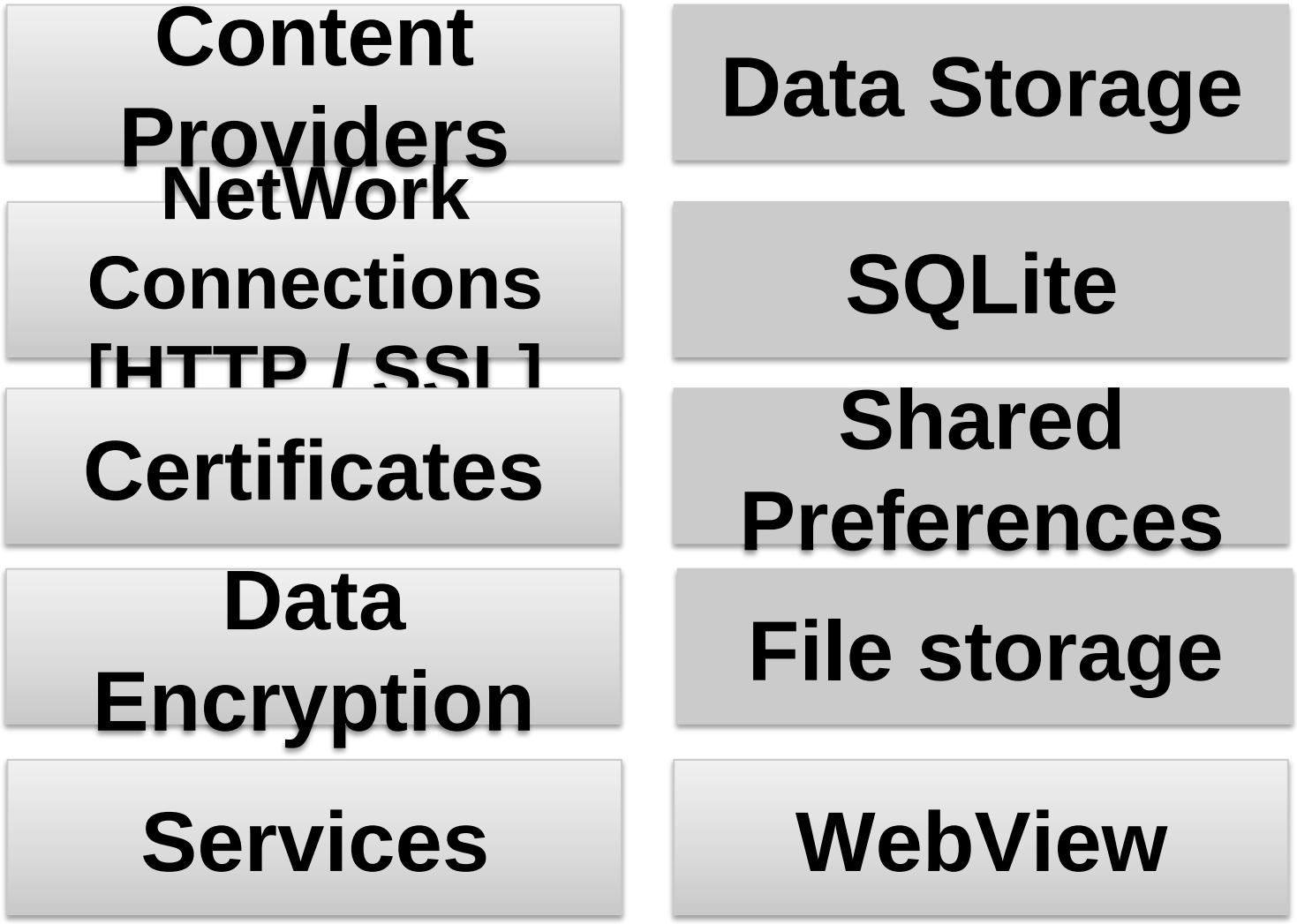
Testing vulnerabilities

Data flow	**Authentication**
Data Storage	**Authorization**
Data leakage	**Server-side**

OWASP Mobile Security Risks

Insecure Data Storage	Session handling
Transport Layer Protection, HTTP/ SSL	Weak Server Side Controls in backend services
Authorization and Authentication	Sensitive information [passwords,API keys,code ofuscation]
Cryptography / Encrypting data	Data Leakage [cache, logging, temp directories]

Testing components security



HTTPS and SSL can protect against Man in the Middle attacks and prevent casual snooping

Data Storage

Encrypt data

Never store user credentials

Tools like **SQLCipher** for storing database in applications

No global permissions granted to applications, using the principle of "**least privilege**".

Secure Storage on Android

The access permission of the created file was set to **WORLD_READABLE / WORLD_WRITABLE**

Other app could read /write the file if the file path is known.

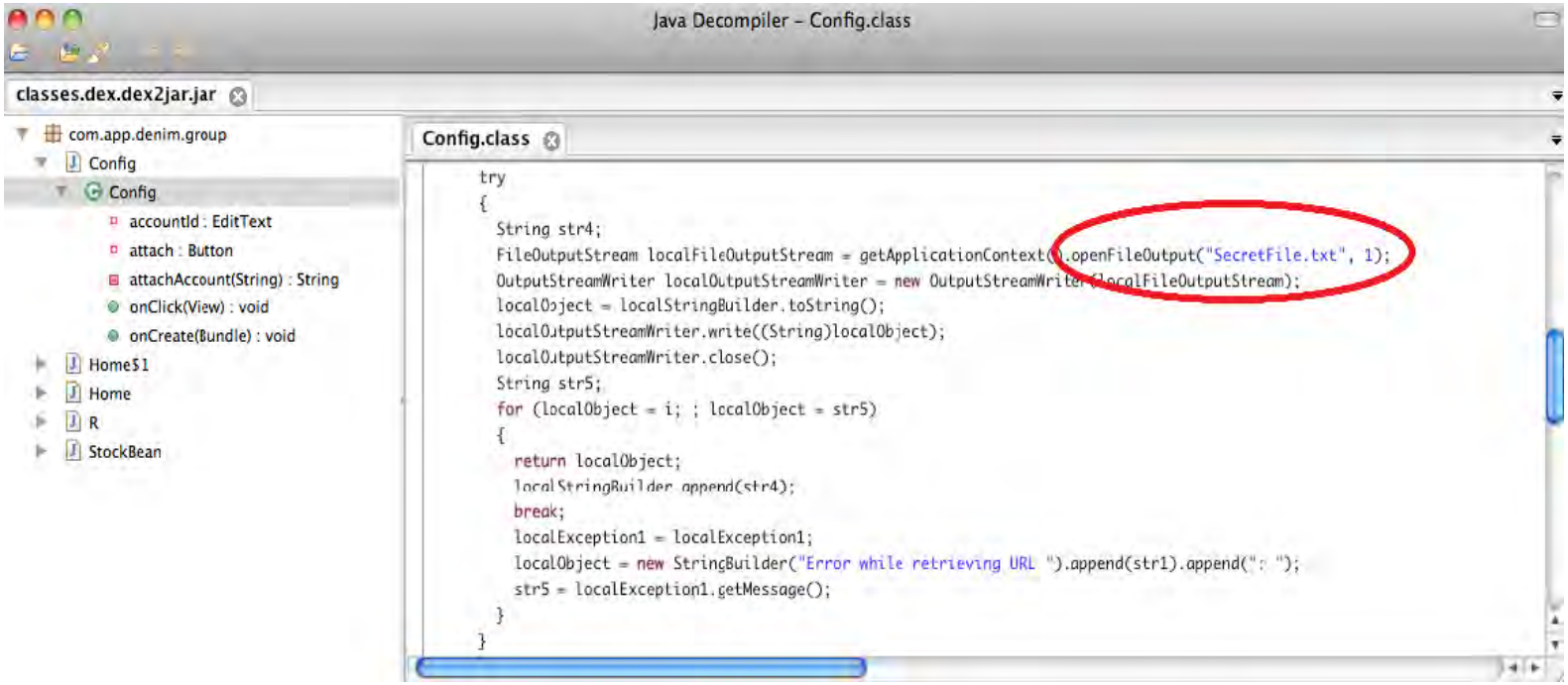
Application data (private files) should be created with the access permission **MODE_PRIVATE**

```
FileOutputStream fos = openFileOutput(“MyFile",
Context.MODE_PRIVATE);
fos.write(“contenido”.getBytes());
fos.close();
```

Insecure Data on Android

Look for file open operations using

- *Context.MODE_WORLD_READABLE*
- *(translates to "1")*



Secure Storage on iOS

iOS

NSFileManager class

NSFileProtectionKey attribute

- NSFileProtectionNone – Always accessible
- NSFileProtectionComplete – Encrypted on disk when device is locked or booting

```
[[NSFileManager defaultManager] createFileAtPath:[self
filePath]
 contents:[@"super secret file contents“
dataUsingEncoding:NSUTF8StringEncoding]
attributes:[NSDictionary
dictionaryWithObject:NSFileProtectionComplete
forKey:NSFileProtectionKey]];
```

DataBase Storage

3rd-party extensions

Support iOS / Android

https://www.zetetic.net/sqlcipher/open-source

256-bit AES Encrypt SQLite database

Secure Preferences on Android

https://github.com/scottyab/secure-preferences

Secure Shared Preferences

```
root@generic:/data/data/com.securepreferences.sample/shared_prefs # ls
com.securepreferences.sample_preferences.xml
s.xml                                                                    <
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="cgASlY/nymum+yv+S+21tmliSHPFDmElhAT9pQ/5DcA">uKJlx6JdM0JaCABUw
qxETBRvUliCmjJidGLtierH/6s</string>
</map>
.xml                                                                     <
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="o51qxitsA/Eag2EbNqSA9A">0vQ8+ILj9keS3fU3r//lRw</string>
    <string name="7cFWxWwcNqYvBRzgh5S5Pw">Jg3xcDITiyK4rN3+4biPwg</string>
    <string name="JD1l6inbJWv3QKurgJkpAg">Zeg6wpqkWjwNEOLQWHzsuQ</string>
    <string name="cgASlY/nymum+yv+S+21tmliSHPFDmElhAT9pQ/5DcA">uKJlx6JdM0JaCABUw
qxETBRvUliCmjJidGLtierH/6s</string>
    <string name="UZFTE9ZTSs49H1G4emERPA">W8KyltBxcICu10mMi7ysoQ</string>
</map>
```

Cryptography

Android and iOS implement standard crypto libraries such as AES algorithm

Android provides the **javax.crypto.spec.PBEKeySpec** and **javax.crypto.SecretKeyFactory** classes to facilitate the generation of the password-based encryption key.

```
try{

PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt,
PBE_ITERATION_COUNT, 256);

SecretKeyFactory keyFactory = SecretKeyFactory.getInstance(PBE_ALGORITHM);
SecretKey tmp = keyFactory.generateSecret(keySpec);

                // Encrypt
                SecretKey secret = new SecretKeySpec(tmp.getEncoded(), "AES");
                Cipher encryptionCipher = Cipher.getInstance(CIPHER_ALGORITHM);
                IvParameterSpec ivspec = new IvParameterSpec(initVector);
                encryptionCipher.init(Cipher.ENCRYPT_MODE, secret, ivspec);
                encryptedText = encryptionCipher.doFinal(cleartext.getBytes());

                // Encode encrypted bytes to Base64 text to save in text file
               result = Base64.encodeToString(encryptedText, Base64.DEFAULT);

                } catch (Exception e) {
                        e.printStackTrace()
 }
```

Avoid Data Leackage on Android

Don't expose data through logcat on production

```
public static final boolean SHOW_LOG =
BuildConfig.DEBUG;

public static void d(final String tag, final String msg) {
        if (SHOW_LOG)
                Log.d(tag, msg);
}
```

```
-assumenosideeffects class android.util.Log
{
public static *** d(...);
public static *** v(...);
public static *** i(...);
public static *** e(...);
}
```

Proguard configuration

Permissions in mobile apps

✓ Try to minimize permissions

AndroidManifest.xml on Android

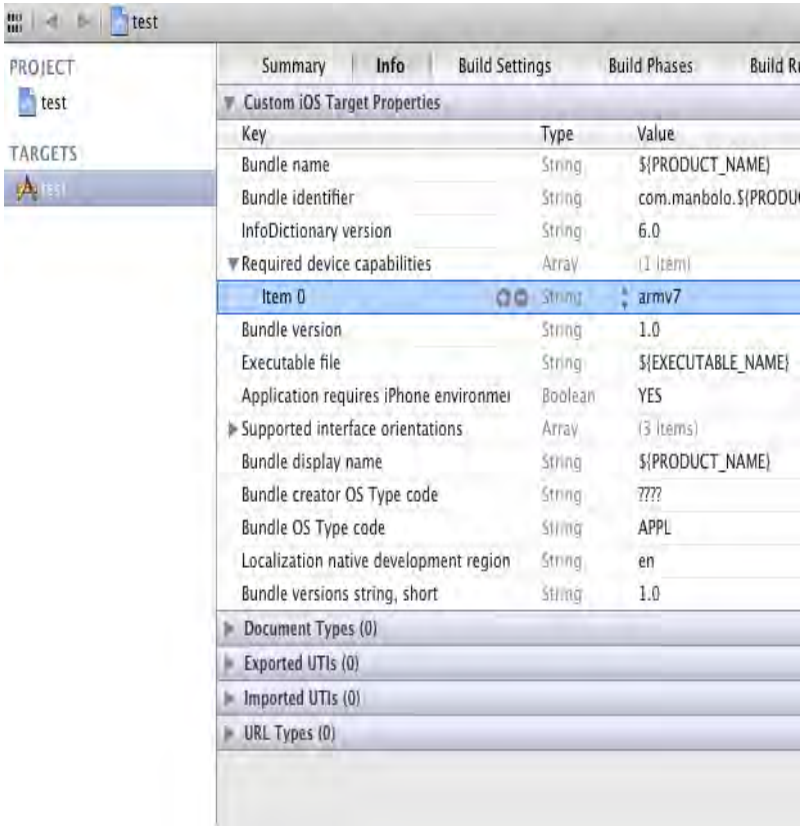
```
<manifest package="com.example.android" …>

<uses-permission android:name=“android.permission.
ACCESS_FINE_LOCATION"/>
<uses-permission
android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.READ_CONTACTS" />
                                    …
</manifest>
```

Permissions in mobile apps

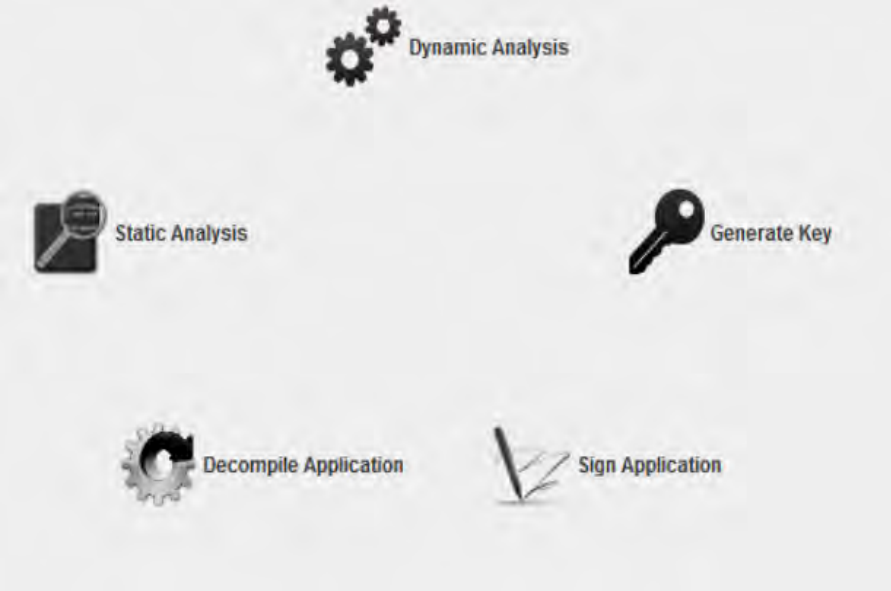
iOS

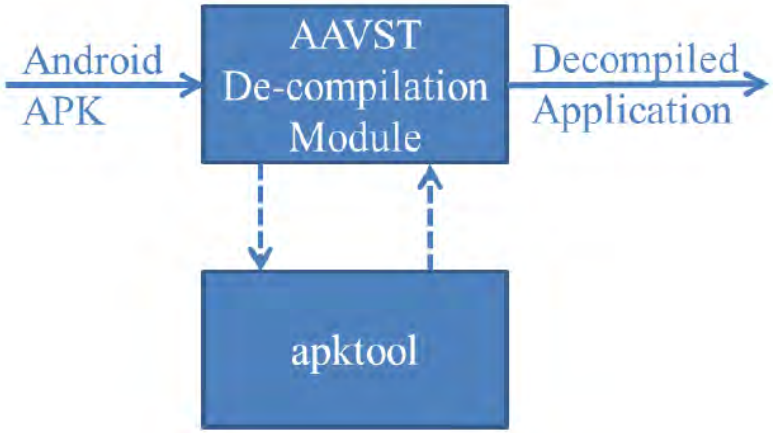
info.plist on iOS



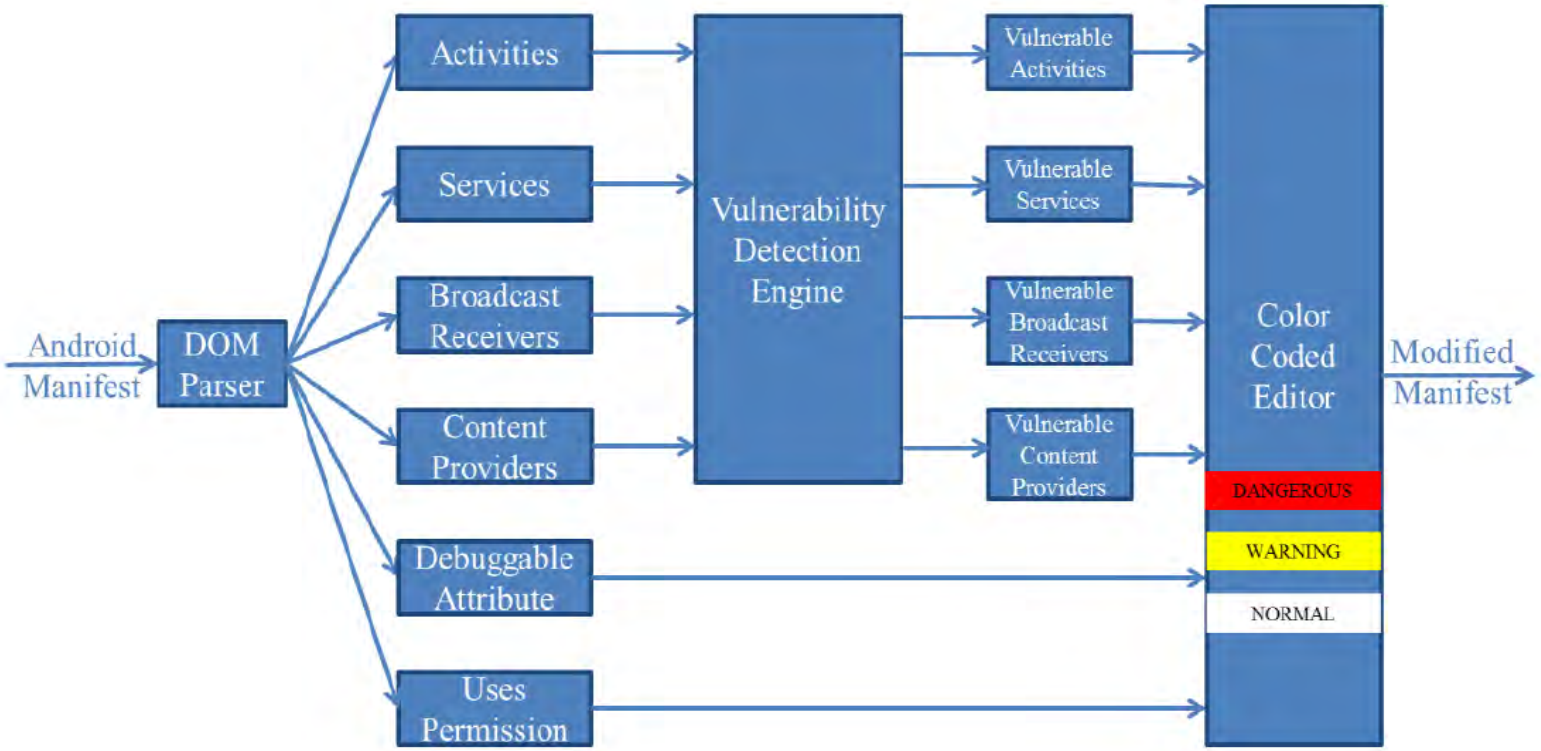

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Pr
opertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CFBundleDevelopmentRegion</key>
        <string>English</string>
        <key>CFBundleExecutable</key>
        <string>Maps</string>
        <key>CFBundleIdentifier</key>
        <string>com.apple.Maps</string>
        <key>CFBundleInfoDictionaryVersion</key>
        <string>6.0</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleResourceSpecification</key>
        <string>ResourceRules.plist</string>
        <key>CFBundleSignature</key>
        <string>????</string>
        <key>CFBundleSupportedPlatforms</key>
        <array>
                <string>iPhoneOS</string>
        </array>
        <key>CFBundleURLTypes</key>
        <array>
                <dict>
                        <key>CFBundleURLSchemes</key>
                        <array>
                                <string>maps</string>
                        </array>
                </dict>
        </array>
        <key>CFBundleVersion</key>
        <string>1.0</string>
        <key>DTCompiler</key>
        <string>4.2</string>
        <key>DTPlatformName</key>
        <string>iphoneos</string>
        <key>DTPlatformVersion</key>
"Info.plist" [readonly] 60L, 1500C
```

Vulnerability scanner on Android

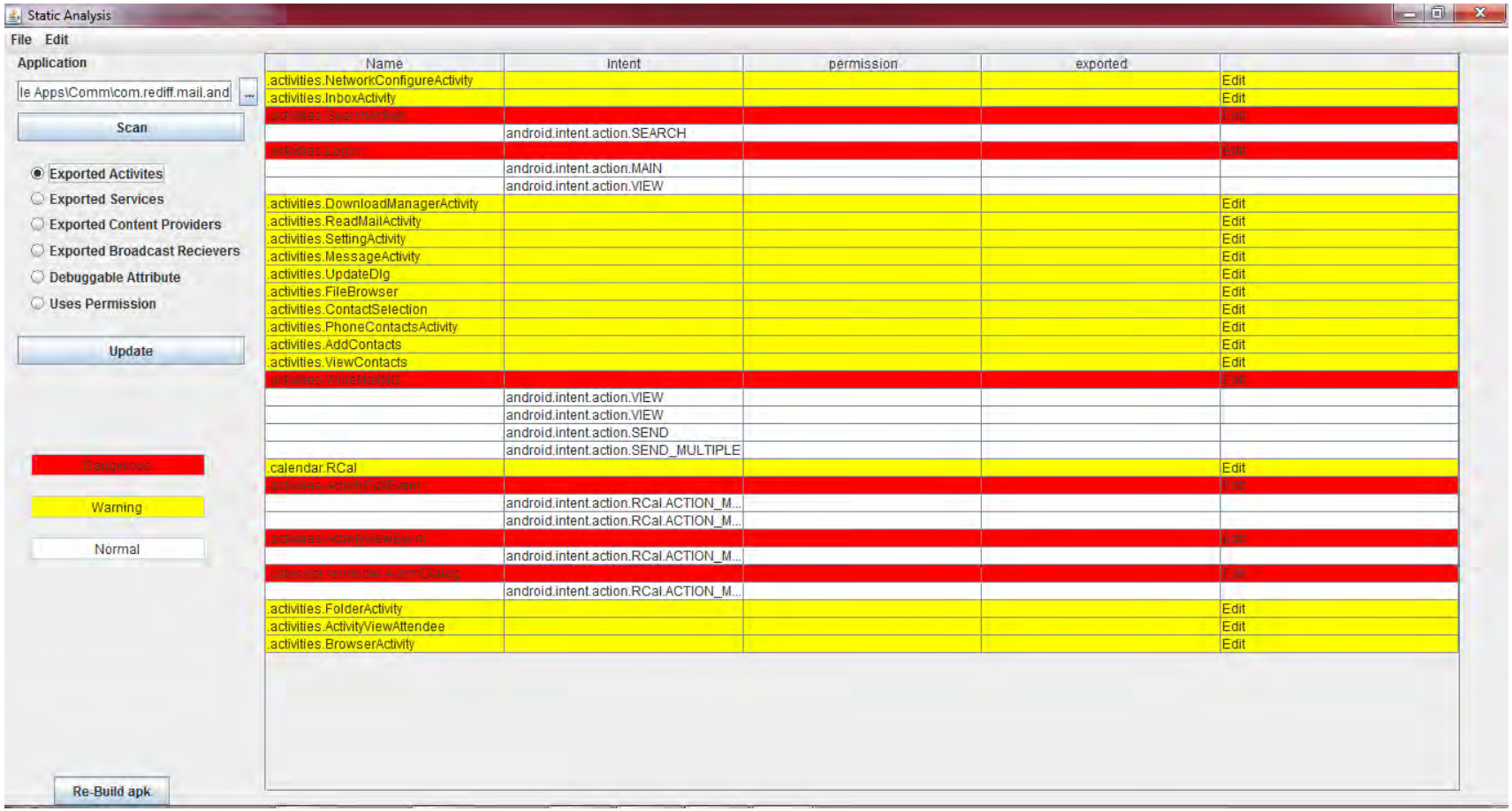




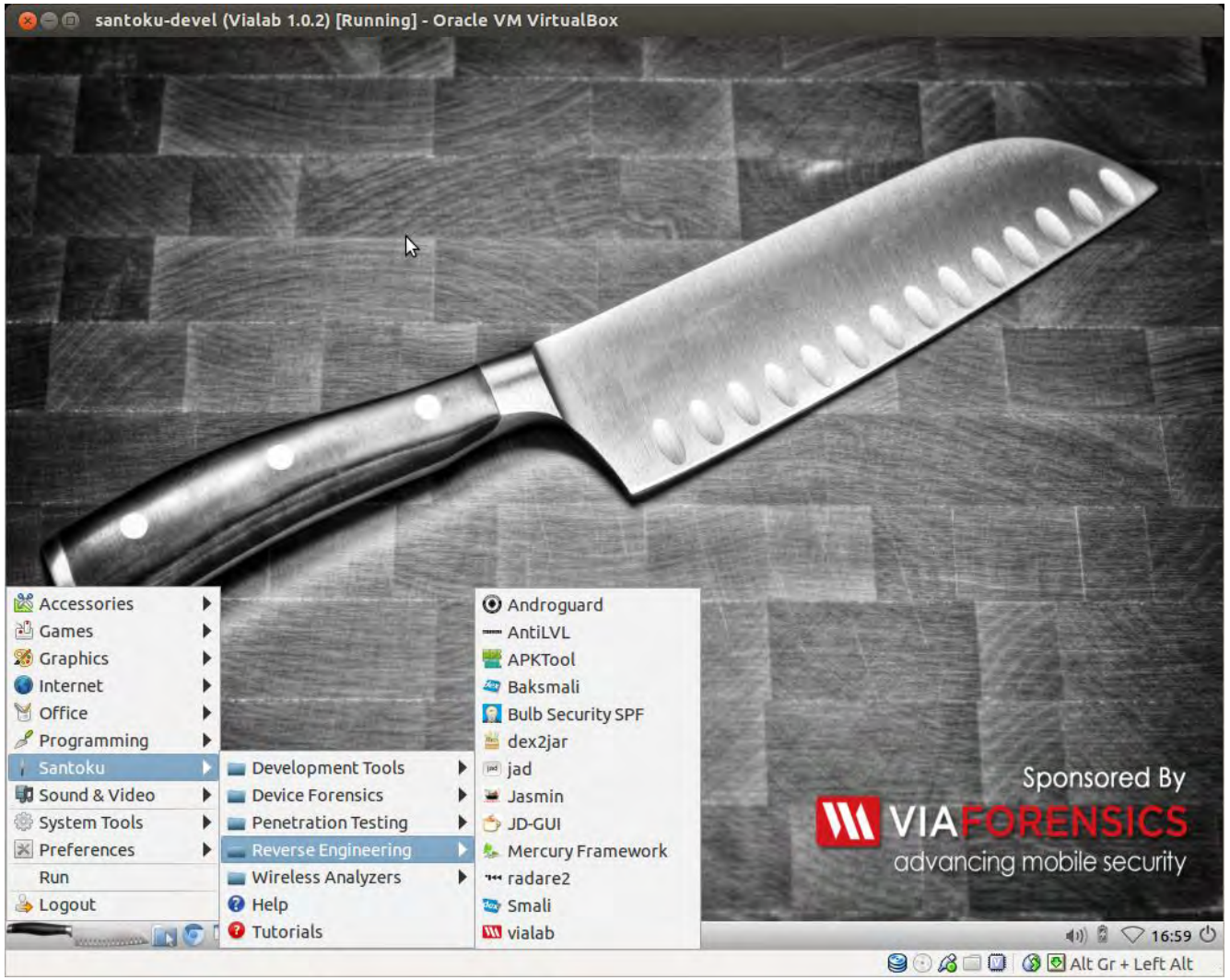
Vulnerability scanner on Android



Vulnerability scanner on Android



Tools / Santoku Linux



Security test plan

Test cases	Example
Test Title /level	**Encryption /critical**
Test Description	When connections are used encryption is used for sending / receiving sensitive data.
Details and tools	All sensitive information (personal data, credit card & banking information etc.) must be encrypted during transmission over any network or communication link.
Expected result	It has been declared that the Application uses encryption when communicating sensitive data.

Security test plan

Test cases	Example
Test Title /level	**Passwords /critical**
Test Description	Passwords and sensitive data are not stored in the device and not echoed when entered into the App, sensitive data is always protected by password.
Details and tools	The objective of the test is to minimize the risk of access to sensitive information should the device be lost, by ensuring that no authentication data can be re-used by simply re-opening the application

Security test plan

Test cases	Example
Test Title /level	**Passwords /critical**
Expected result	1. Entering a password or other sensitive data will not leave it in clear text if completion of the fields is interrupted but not exited. 2. Passwords, credit card details, or other sensitive data do not remain in clear text in the fields where they were previously entered, when the application is reentered. 3. Sensitive personal data should always need entry of a password before it can be accessed.

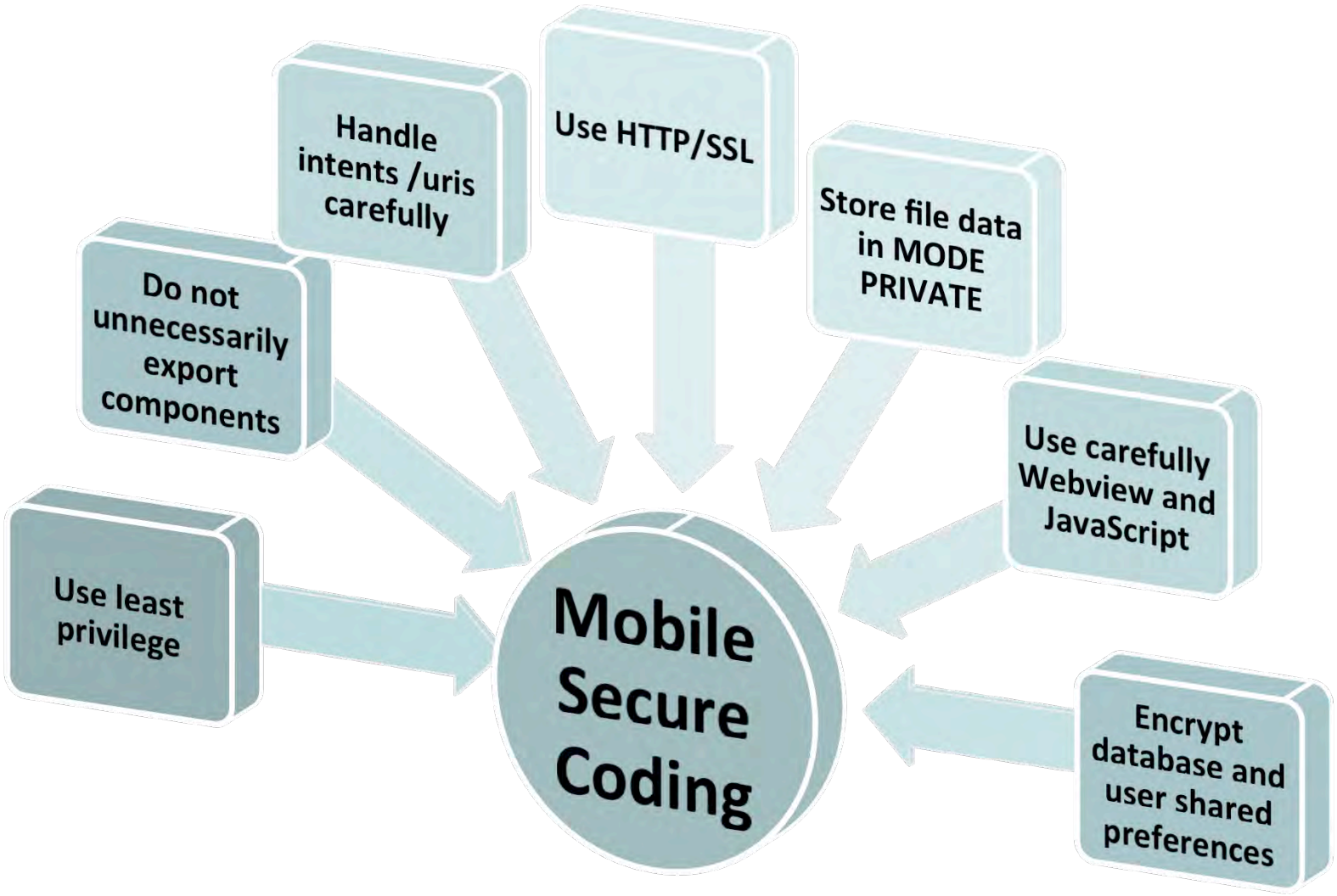
Best Practices

Integrate security in Continuous Integration (CI) process

Apply encryption /decryption techniques used for sensitive data communication

Detect areas in tested application that have more risks to detect vulnerabilities

Install an automated security vulnerability scanner, integrated with your continuous integration tool


Handle intents /uris carefully
Use HTTP/SSL
Store file data in MODE PRIVATE
Do not unnecessarily export components
Use carefully Webview and JavaScript
Use least privilege
Mobile Secure Coding
Encrypt database and user shared preferences

Attacker vs Defenders

Defenders	Attackers
Often have limited time to put defences in place	Can take time to plan their attack
Have limited opportunities to improve their defences	Can invent new ways to attack
Need to defend against a range of possible attacks	Need only pick the most effective one
Need to defend all entry points	Can choose where to attack
Has limited resources to defend	Can be multiple attackers

References

OWASP Mobile Security Project

https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

Android Security Best Practices

http://developer.android.com/training/articles/security-tips.html

References

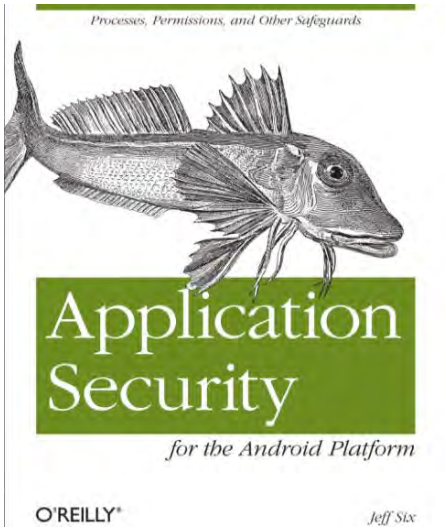
Tools

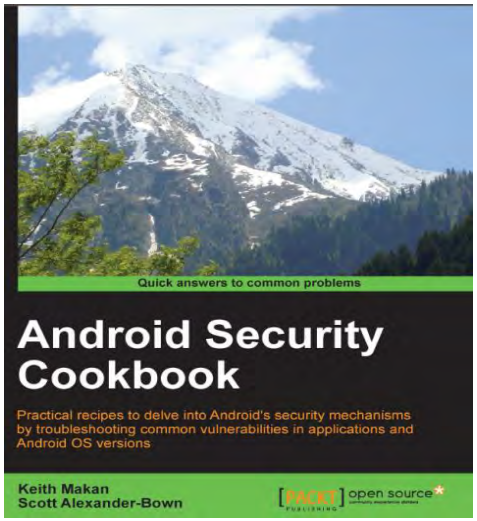
https://github.com/secmobi/wiki.secmobi.com

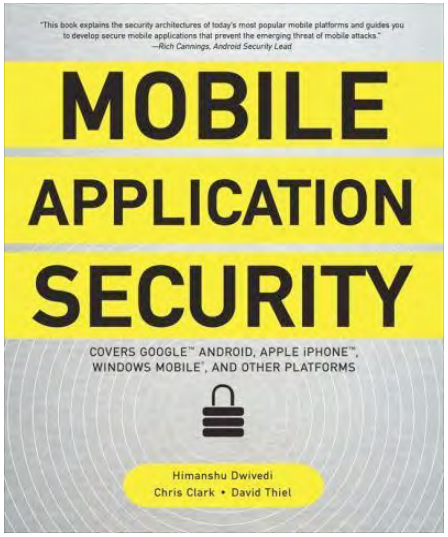
IOS Application Security Testing

https://www.owasp.org/index.php/IOS_Application_Security_Testing_Cheat_Sheet

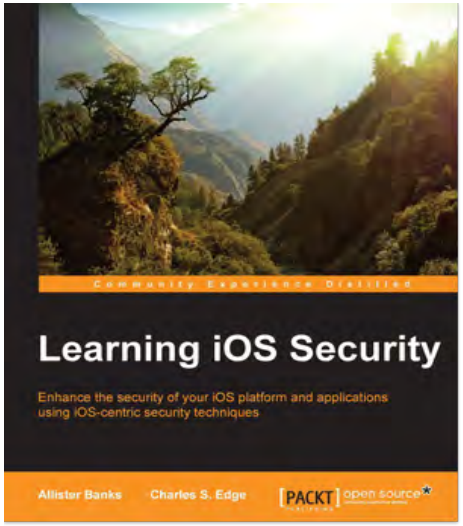
Books

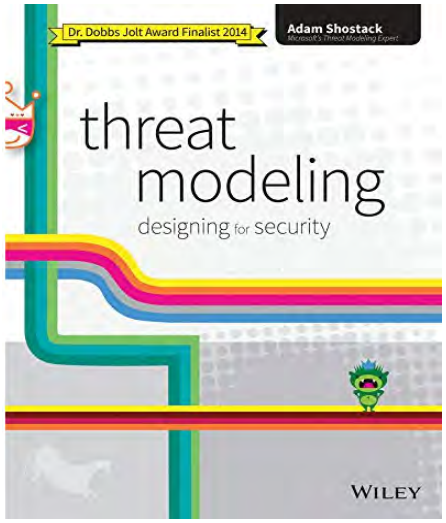

Processes, Permissions, and Other Safeguards
Application Security
for the Android Platform
O'REILLY
Jeff Six


Quick answers to common problems
Android Security Cookbook
Practical recipes to delve into Android's security mechanisms by troubleshooting common vulnerabilities in applications and Android OS versions
Keith Makan
Scott Alexander-Bown
PACKT open source


MOBILE
APPLICATION
SECURITY
COVERS GOOGLE™ ANDROID, APPLE iPHONE™, WINDOWS MOBILE®, AND OTHER PLATFORMS
Himanshu Dwivedi
Chris Clark • David Thiel


Android Apps
Security
Sheran A. Gunasekera
Apress


Learning iOS Security
Enhance the security of your iOS platform and applications using iOS-centric security techniques
Allister Banks
Charles S. Edge
PACKT open source


Dr. Dobbs Jolt Award Finalist 2014
Adam Shostack
threat
modeling
designing for security
WILEY

Thank You,

Questions?