OWASP
Open Web Application Security Project

OWASP German Chapter Stammtisch Initiative/Ruhrpott

Android App Pentest Workshop 102

About

- What we try to cover in the second session:
  - MITM attacks
  - Understanding smali code
  - Removing smali code
  - Adding smali code
  - JNI (Java Native Interface)

Setup

- You will need the following:
  - A laptop or any hardware that can run a VM
  - VM: Ubuntu 16.10 Yakkety(64bit).vdi
  - Android VM: Android-x86.5.1 rc1.vdi
  - Virtualbox (recommended)
  - Internet connection to google up things

DL: https://drive.google.com/drive/folders/0BwhtuArcTcxMWlhvTW5SYkFsbWc

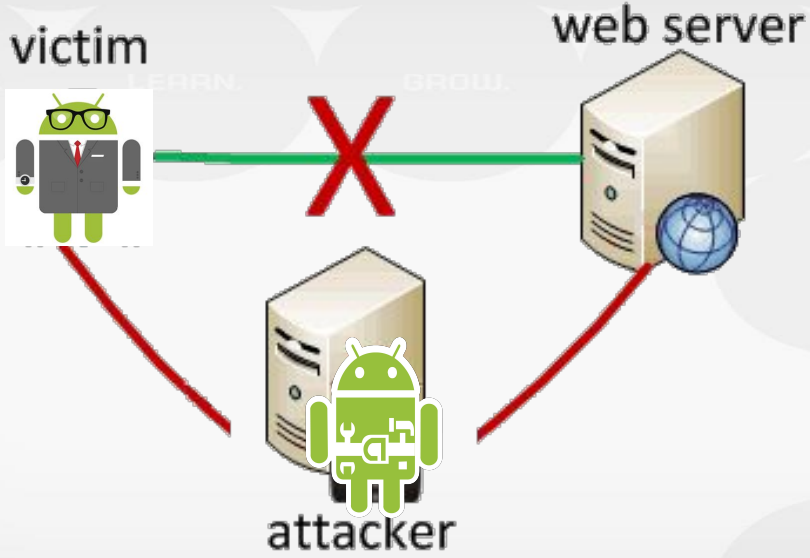
Attended last session?

Follow the instructions at
https://github.com/OWASP-Ruhrpott/owasp-workshop-android-pentest/blob/master/CHANGELOG.md

Recap - Session 1

- Setup of a Mobile Application Pentest Environment
- Basics of Mobile Application Pentests
- Common issues in Mobile Applications

Man In the Middle Attacks



How to MITM

- There are many way to redirect traffic

How to MITM

- There are many way to redirect traffic
  - Layer2 attacks
  - Redirect via iptables on a router/switch
  - Android proxy settings (not working in VM)
  - etc.

Task – Basic HTTP Request

- You will find iptables rules in /sdcard/Download/proxy.iptables

  iptables -t nat -A OUTPUT -o eth0 -p tcp --dport 80 -j DNAT --to 10.13.13.102:8080
- Start HTTP server: sudo python ~/Resources/http/http.py

[10 min]

Basic HTTP Request – Commands

Command	Comment
iptables -t nat -A OUTPUT -o eth0 -p tcp --dport 80 -j DNAT --to 10.13.13.102:8080	Traffic redirection
Start Burp with a proxy listening on Port 8080	
Open android application and submit HTTP request	
View Burp Logs	

Task - Basic HTTPS Request

- This time the application uses a SSL/TLS secured connection
- Modify your iptables
- Start https server: sudo python ~/Resources/https/https.py

Can you still intercept the traffic?

[10 min]

Basic HTTPS Request – Commands

Command	Comment
iptables -t nat -A OUTPUT -o eth0 -p tcp --dport 443 -j DNAT --to 10.13.13.102:8080	Traffic redirection
Start Burp with a proxy listening on Port 8080	
Import Burp CA into android VM	
Open android application and submit HTTPS request	
View Burp Logs	

Task – SSL Pinning

“Pinning is the process of associating a host with their expected X509 certificate or public key. Once a certificate or public key is known or seen for a host, the certificate or public key is associated or 'pinned' to the host. ”

Hint: You do not have to modify the source code!

[10 min]

SSL Pinning – Commands

Command	Comment
iptables -t nat -A OUTPUT -o eth0 -p tcp --dport 443 -j DNAT --to 10.13.13.102:8080	
Replace assets/owasp.crt with Burp certificate	
Import Burp CA on android VM	
Open android application and submit HTTPS request	
View Burp Logs	

Introduction to smali code

- We will not discuss all kind of syntax elements
- But we will discuss enough to get you prepared for the next tasks

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"


# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"


# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"

# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

any of static, private, or constructor

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"


# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

none of static, private, or constructor

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"

# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method

# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

any of static, private, or constructor

none of static, private, or constructor

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"


# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

Registers are used for storing. Starts with 0, can be up to 15 [some „move“ instructions can use other, too]

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"


# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

Registers are used for storing. Starts with 0, can be up to 15 [some move“ instructions can use other, too]

Some more about registers

- Further information: https://github.com/JesusFreke/smali/wiki/Registers

```
.class public Lcom/example/context/testproject/MainActivity;
.super Landroid/support/v7/app/AppCompatActivity;
.source "MainActivity.java"


# direct methods
.method public constructor <init>()V
    .locals 0

    .prologue
    .line 12
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    return-void
.end method


# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
    .locals 1
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;

    .prologue
    .line 16
    invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

    .line 17
    const v0, 0x7f04001a

    invoke-virtual {p0, v0}, Lcom/example/context/testproject/MainActivity;->setContentView(I)V

    .line 19
    return-void
.end method
```

Primitive Types

Symbol	Type
V	void - can only be used for return types
Z	boolean
B	byte
S	short
C	char
I	int
J	long (64 bits)
F	float
D	double (64 bits)

Some more about types

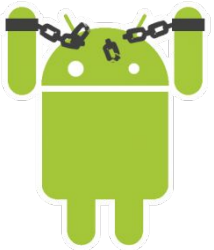
- Further information: https://github.com/JesusFreke/smali/wiki/TypesMethodsAndFields

Even more links

- https://github.com/JesusFreke/smali/wiki
- https://source.android.com/devices/tech/dalvik/dalvik-bytecode.html
- https://source.android.com/devices/tech/dalvik/index.html
- http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

Task –Root Detection Bypass

```
package com.example.context.testproject;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import java.io.File;
import android.widget.TextView;


public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        File file = new File("/system/xbin/su");
        if(file.exists()) {
            TextView text = (TextView) findViewById(R.id.root_detection);
            text.setText("Rooted device identified");
        }
        else {
            TextView text = (TextView) findViewById(R.id.root_detection);
            text.setText("You got a clean device ");
        }

    }
}
```



Root Detection Bypass – Commands

Command	Comment
java -jar apktool_2.2.0.jar d <AppName>	Decode apk
Browse to decompiled code and open .smali file	Open file with your favourite editor
Change e.g. file path, delete check or change if statement	Modify smali code
java -jar apktool_2.2.0.jar b ./app-debug	Rebuild
keytool -genkey -alias mystore -keyalg RSA -keystore KeyStore.jks -keysize 2048	Create keystore (needs to be done only once)
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore <name> <apk> <alias> -storepass <pw>	Sign apk - othervise you wouldn't be allowed to install
adb uninstall <app>	Uninstall old version
adb install <apk>	Install new version

What about new code?

- Sometimes you want to extend the code with your own, e.g. logcat messages

const-string v0, “owasplog"

const-string v1, “Hello"

invoke-static {v0, v1},
Landroid/util/Log;->e(Ljava/lang/String;Ljava/lang/String;)I

The “simple” way

- We want to avoid writing our own smali code
- We can use Java2Smali (https://github.com/ollide/intellij-java2smali) which is an Android Studio plugin

Task - Secure Encryption

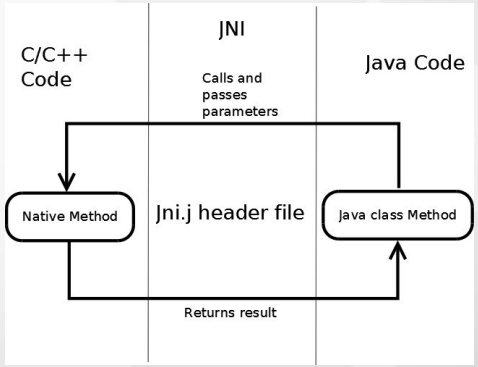
- Add your own log messages to the application
- Understand the encryption/decryption process
- Break it!

Secure Encryption

- Cipher text AxoYZ2hsHi1VVSE5MEdbJG0LQVA+PTZATGw/NldTPw (base64)
- Based on android.os.Build.ID
- Cipher text XOR android.os.Build.ID

Java Native Interface

"The Java Native Interface (JNI) is a programming framework that enables Java code running in a Java Virtual Machine (JVM) to call and be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages such as C, C++ and assembly." *https://en.wikipedia.org/wiki/Java_Native_Interface*



Risks that come with JNI

- Vulnerabilities that do not occur within JAVA such as Buffer Overflows

=> Android itself has protections/mitigations against those

Task - Library Call

Your task is to extract the library and build your own project which can use the library!

Library Call – Commands

Command	Comment
adb pull /data/app/ruhrpott.owasp.com.vuln_app_1 .	Download apk + library folder
strings ./lib/x86/liblibcall.so | grep -i java	Retrieve name of function and namespace
Create new Android Studio project with same namespace as apk	
Create same class and function name	
Load Library	
https://developer.android.com/ndk/samples/sample_hellojni.html	

END OF SESSION 2

Other CTF Challenges

https://drive.google.com/drive/folders/0B_nKK17ymHGpUF9uR1BWRWNCbU0?usp=sharing (goo.gl/LKJRPT)