Xtreme Android Exploitation Lab

by
Anant Shrivastava
&
Anto Joseph

Introductions

Trainer (Anant Shrivastava)

- Information Security Consultant
- Admin - Dev - Security
- null + OWASP + G4H
- http://anantshri.info and @anantshri
- Speaker / Trainer : Blackhat, Nullcon, c0c0n, RuxCon, RootConf
- Regional Director NotSoSecure Global Services

Trainer Anto Joseph

Security Engineer @ Intel

DevOps / Security Guy

Speaker / Trainer : Blackhat ,Defcon,HITB,Troopers, AppSec EU,x33fconf,HackInParis,Brucon,NullCon …

github.com/antojoseph

Enthusiastic about Mobile Security / IOT / Machine Learning

Quick Introductions

Three things

1. Your Name
2. Your Level of Experience / Comfort with Android
3. What is your expectation from the Session

Theory v/s Hands-On

We can either spend time doing theory about android or we can learn how things work and can then use references to get theory side of it solid.

The entire lab is designed in a scenario based situation where we will perform the same attacks that an attacker can do to gain access.

Workshop Setup

- VirtualBox and Genymotion
- 2 VM’s provided by Trainers
  - Genymotion VM
  - Android Tamer (nullcon Edition)

How to Get Started

- Import VM
- Start Both VM
  - Credentials:
    - Username: android
    - Password: tamer
- Check connectivity
  - ping google.com from within Android Tamer VM
  - ping tamer vm ip from within Android VM

Day 1

Course

Understand android application code.

OWASP Top 10 Mobile Risk

How to Decompile android application

1. How to handling obfuscated code
2. How Dalvik Works

Traffic interception of android applications

1. How to handle SSL protections (cert validation, SSL Pinning)
2. How to intercept non HTTP Traffic

Defeating Root detection

HTML5 Application analysis

Static analysis of application

Assumptions

- Aware of Android SDK or basic components of SDK (adb, fastboot)
- Layman's View of Android
- Aware of using Linux Command line and Scripting

In case you are stuck in one of these feel free to google first.

If in doubt ask one of the trainers.

OWASP Top 10 Risk - 2014

M1: Weak Server Side Controls

M2: Insecure Data Storage

M3: Insufficient Transport Layer Protection

M4: Unintended Data Leakage

M5: Poor Authorization and Authentication

M6: Broken Cryptography

M7: Client Side Injection

M8: Security Decisions Via Untrusted Inputs

M9: Improper Session Handling

M10: Lack of Binary Protections

Top 10 Risk - 2016

Candidate release list is publically available

Expect a large amount of changes on it.

https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10

Is this course OWASP Top 10 focused

In Short **NO**

This course is designed to be as realistic as possible based on real world problems that everyone in infosec community faces when it comes to android.

- Decompilation
- Traffic interception
- Root Detection
- HTML5 Application
- static and dynamic analysis
- manual and automated analysis
- bug hunting in core android

Understanding Application Code

APK (Android Package)

modified Java code

Can be decompiled to

- Java (easy to read but may not be accurate all time)
- Smali (bytecode representation)

Challenge : Decompile Applications

- APK ~ JAR ~ ZIP
- unzip APK it contains
  - classes.dex
  - resources
  - XML’s (encrypted)
- classes.dex -> Java classes combined together
  - decompile using dex2jar or enjarify
- resources : images etc
  - no need to decompile
- XML’s : binary version to save processing time
  - apktool : to decode XML to human readable format

How to Decompile Applications

All tools present on AndroidTamer

apk2java is custom script which performs decompilation

- decrypts all XML files
- gives java as well as smali code
- gives java code decompiled via 2 different decompiler (jad and jadx)

```
android@tamer ~ $ apk2java package.apk
```

Retrieve APK For Device

```
android@tamer ~ $ adb shell pm list packages -f

android@tamer ~ $ adb shell pm path <packageName>

android@tamer ~ $ adb pull <package_path> ./
```

Now follow previous steps.

Exercise 1

Retrieve the application challenge1 from Android VM and then perform decompile operation.

Things to Understand

Challenge 1 Source code available inside Android-Studio

Very accurate retrieval of source code.

Accuracy ~ complexity

Challenge2

1. Decompile XYZ.apk and identify the key

How to read Obfuscated Code

1 . Cool example on how to not do Obfuscation : http://obfuscat.ion.land/

Usual Techniques :

1 . Control Flow

2. String Encryption

3. Class - renaming

4 .Method -renaming

5. Java Reflection to hide method calling

Understanding Obfuscated Code

- Tools Available
    1. Simplify (generic de-obfuscation)
    2. JEB - (Modules)
    3. eg : https://gist.github.com/AKosterin/af8c2dd2aa372c99b507
- How obfuscation works
    - Useless arithmetic
    - class-renaming
    - Infinite loops
    - control flow obfuscation
    - String encryption
    - Method Daisy Chaining / Class Daisy Chaining

Challenge : Obfuscated Code

Try out the challenge and try to crack it !

Failures

decompile fails/can’t be understood coz its obfuscated

So how do we defeat obfuscation

```
static String source="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
static String target="Q9A8ZWS7XEDC6RFVT5GBY4HNU3J2MI1KO0LP";

public  String obfuscate(String s) {
    try {
        char[] result = new char[10];
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            int index = source.indexOf(c);
            result[i] = target.charAt(index);
        }

        return new String(result);
    }catch (Exception e){

        Toast.makeText(getApplicationContext(),"Figure the key"+e.toString(),Toast.LENGTH_LONG).show();
        return "1";
    }
}
```

How Dalvik / ART VM

Register Based

Bytecode is different from a standard JVM

dex = device independent code

odex /oat = optimised for your device

DALVIK : Use dexopt to optimized dex files

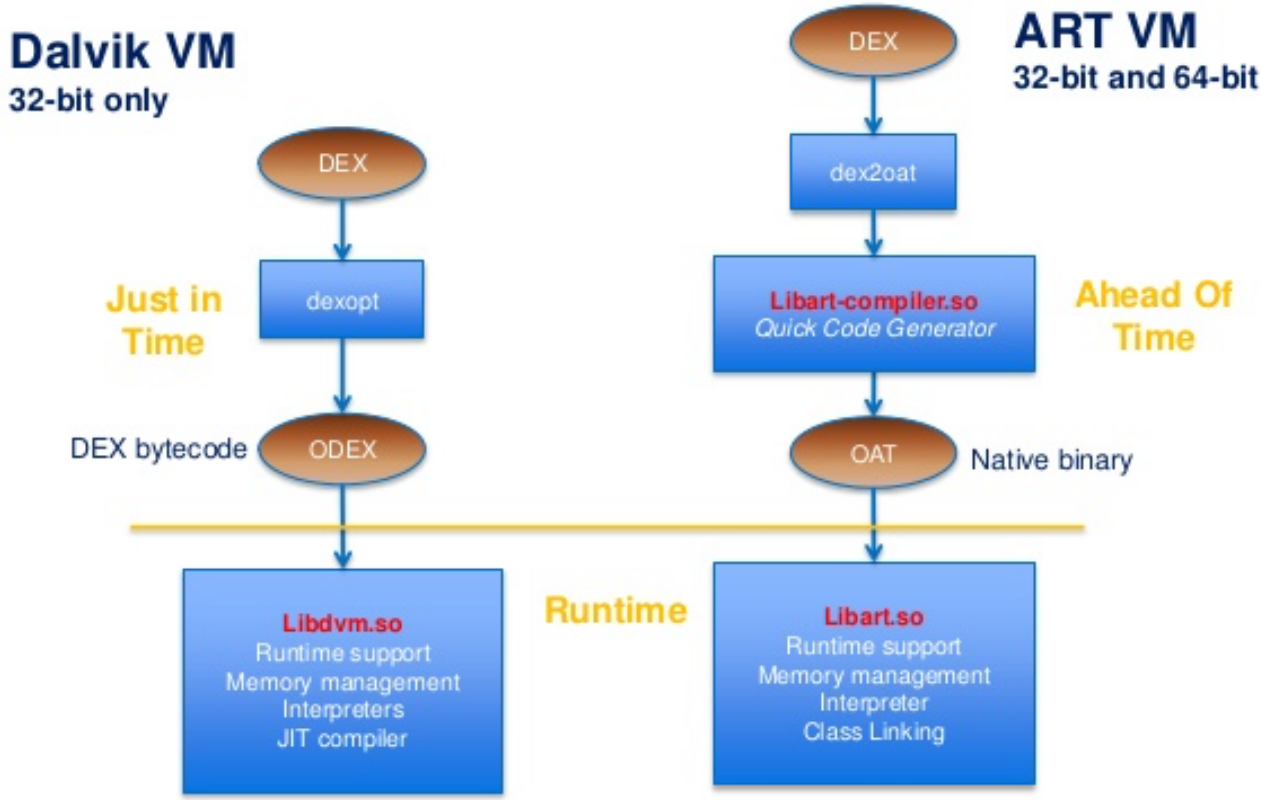
stored in dalvik-cache

System apps usually ship as odex / OAT files

ART : No More JIT

ART : Oat2dex converter to decrypt Lollipop apps and Jars

Dalvik VM vs. ART VM



Why obfuscate?

Good to have feature

defers application analysis.

should not be considered a replacement for best practices. ugly code / logic / human error behind obfuscation is still applicable.

Challenge : Traffic interception

Often Applications will have some kind of traffic going over internet.

As part of assessment we need to be able to see this traffic.

Proxy Configuration: Demo

How to configure burp, charles, ZAP etc for proxy interception

- Start proxy in Tamer (note ip and port)
- Set proxy in wifi settings on Device
- Check traffic interception via http traffic via browser

Exercise 3

Intercept traffic of OKVolleyHTTPSample

Challenge

Try GET HTTPS section

PKI is broken

1. System Trust **all CA in Trust Store** (PortSwigger CA)

2. System Trust's ROOT CA **not certification chain**

3. **Any CA** can issue certificate to **any website** (Diginotar, Trustwave, NIC and many more)

4. Certificate **Stolen**: Welcome to **Revocation** hell and **CRL** Nightmare

5. **OCSP** to the rescue **over port 80**

6. many more read: https://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf

Break Protection

1. Extract ROOT CA from your proxy software
2. Import ROOT CA into android via commandline or browser

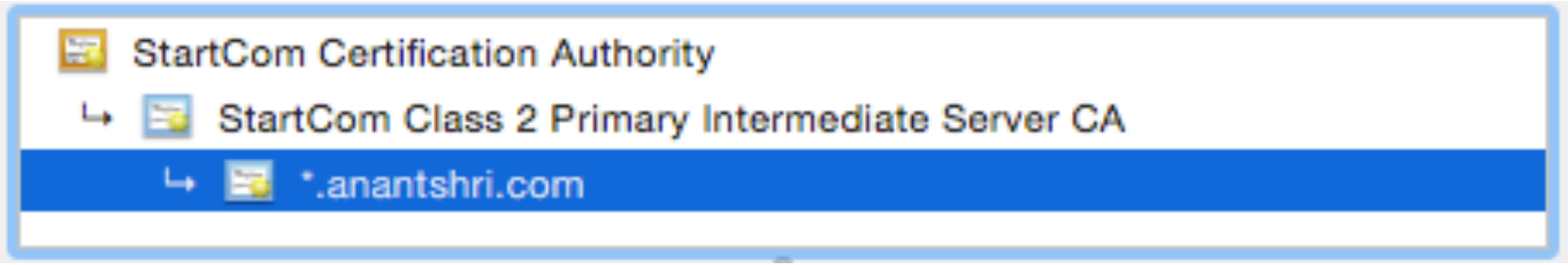
```
android@tamer ~ $ adb push rootca.pem /mnt/sdcard/Downloads/
```

Exercise 4

intercept https traffic

Challenge: Cert Pinning

Advanced form of HTTPS where certificate is validated not via OS trust store but via its own checks.



How it works

1. Identify Which certificate you want to pin.
2. Generate Sha1 / md5 sum of the certificate
3. Hardcode the cert pin inside your application
    a. use default platform code
    b. use a framework
    c. use custom self written code

Detailed Readings

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

http://www.slideshare.net/anantshri/ssl-pinning-and-bypasses-android-and-ios

Ways to defeat it

1. APK Decompile and modify code

Excellent example: http://blog.dewhurstsecurity.com/2015/11/10/mobile-security-certificate-pining.html

2. Xposed Module

What is Xposed module and how they work i will leave for Day2.

1. Root your device
2. Install Xposed Framework
3. Install JustTrustMe module
4. Enable JustTrustMe
5. Reboot

Exercise 5

Decrypt traffic for CERT PINNING Action

Challenge : Root detection

- To ensure that device is secure and any data on it can be assumed to be secure.
- No one is tampering with application / traffic
- and many more reasons

For Pentesters

Big Headache coz pentest tool works on rooted devices where the application may not work

How it works

Mainly Blacklist

- Detect binary presence (/usr/bin/su)
- Detect commands (which)
- Detect superuser controller (superuser.apk)

Good coverage : https://www.blackhat.com/docs/eu-15/materials/eu-15-Benameur-All-Your-Root-Checks-Are-Belong-To-Us-The-Sad-State-Of-Root-Detection.pdf

How to defeat it

Hide binaries

Overload system calls to check for the binaries

Exercise

Bypass Root Checks on Root Inspector

HTML5

Used for hybrid Cross platform applications

Easy to build using HTML / CSS / JS

Common HTML5 issues

- Source code disclosure
- Javascript issues mainly Cross Site Scripting (DOM XSS)
- SSL Configuration
- Local Storage and caching data leakage
- Framework specific issue (phonegap, titanium etc)
- Easy to repackage

How to Find Issues

Decompile Application

HTML5 source code will be in assets/www folder

For in device storage application will use localstorage (refer /data/data/<app>)

Exercise

Analyze TripCase Application and see if any data is leaked

Challenge : Static Analysis

Identify flaws without running application

requires deeper understanding of code

however task made more simpler with opensource code scanners

How to Scan

AndroidTamer

- Mobilizer
- droidscan.sh

Challenge

c0c0n Application : Get the Key

Note: we want you to decompile change code and recompile

How to decompile and recompile via smali

Decompile

```
android@tamer ~ $ apktool d <apk>
```

Re-compile

```
android@tamer ~ $ apktool b <folder>
```

```
Is that all
```

```
android@tamer ~ $ keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

```
android@tamer ~ $ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore coc.on-1.apk alias_name
```

Day 2

Outline of the day

Manual and Automated dynamic analysis

Application hooking and dynamic instrumentation with writing your own module

Fuzzing Android (core and applications)

CTF challenge to be solved based on learnings during class. (expected to write a code or use proper tools)

Dynamic analysis

Runtime Analysis of application

Gives out accurate runtime status of the application

Tools to be used: Manual

adb

ddms androidmonitor

pidcat

Things to look

/data/data/<app>

/sdcard/

/sdcard1/

Exercise

Perform manual dynamic analysis and identify flaws in base CRM

Automated Analysis

Let the automaton take over

We can perform most of these checks dynamically

Multiple Frameworks in W.I.P. Status

- MobSF (Ajin)
- Marvin
- cuckoo-droid
- drozer
- Qark

How to

Configure MobSF

Download and setup VM

Start MobSF

Run MobSF

provide an application

Analyze

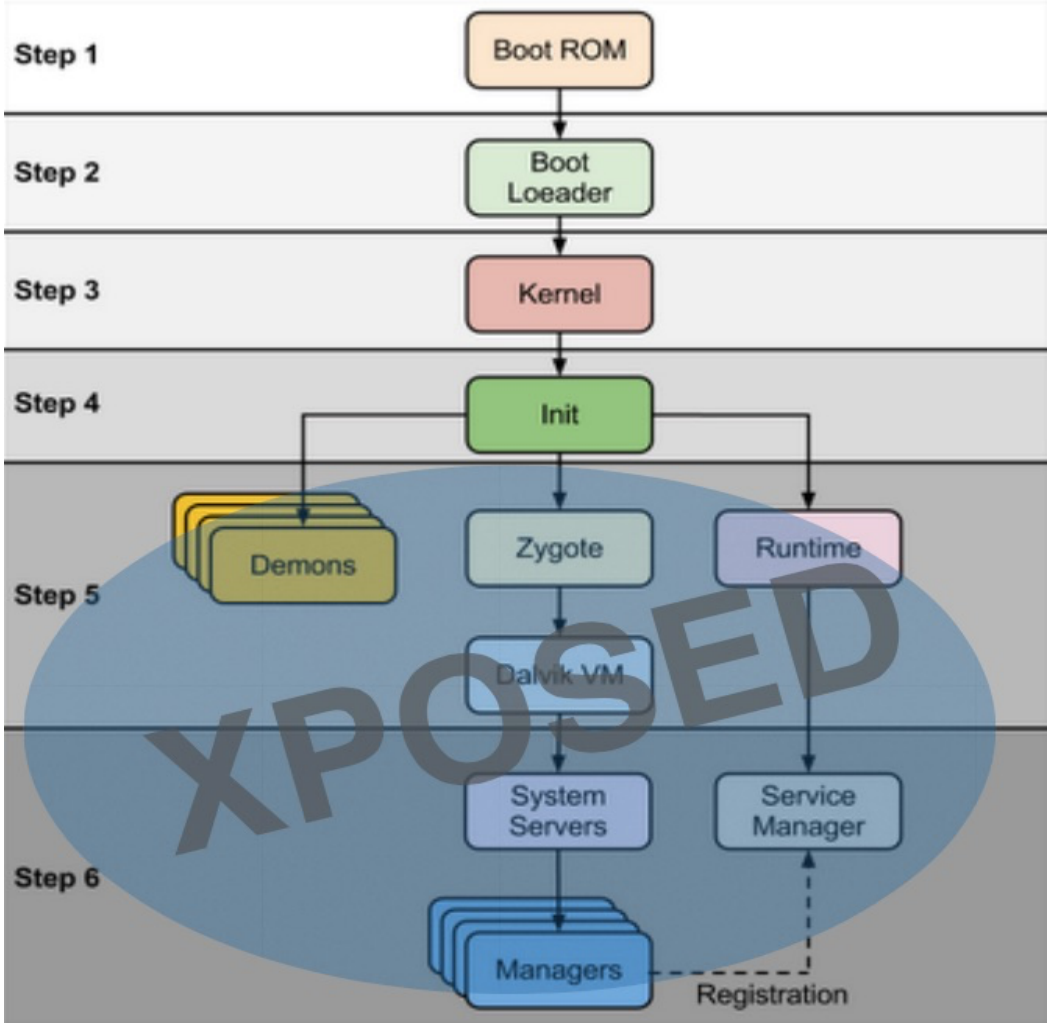
Exercise

Dynamically analyze CrackMe’s

Hooking and Dynamic instrumentation

No Recompilation

Runtime behaviour modification



Ref: http://www.slideshare.net/AbhinavChourasiaGMOB/null-xposedinternals

Demo + Exercise

Lets write a module for Xposed

Frida

It’s a dynamic code instrumentation toolkit.

Inject snippets of JavaScript into native apps on Windows, Mac, Linux, iOS and Android.

API Bindings for Python, node-js

navigate to : https://github.com/antojoseph/frida-android-hooks

Other instrumentation framework

Frida : Hands On

- Root Detection Bypass
- Debugger Check Bypass
- WebView Logging
- Device Id Spoofing
- Certificate Pinning Bypass
- Login Screen bruteforce

Identify more issues

Identify issues in Core of android

Issues beyond what’s found via tools

Introducing Fuzzing

Fuzzzzzzzzzzzzzz

What is Fuzzing

How fuzzing works

More Fuzzing Applications

How and what

intent fuzzing

c binary fuzzing

and more

Challenge: Finding flaws in Core

How to find next stagefright

Core Fuzzing Setup

Software and tools required

Paid and free alternatives (lets prefer free here)

Setup

How it works

Exercise

Generate dataset

Run Dataset against target

write the glue script

write the log collection script

What we learned : A recap

- How does an Android Application Looks like
- How to decompile android application
- How to Intercept traffic of an android application (http/ssl/non-http)
- How to analyze html5 Applications
- Manual Analysis (static and dynamic) of android application
- Automated Analysis (Static and Dynamic)
- Dynamic Instrumentation of Code using Xposed and Frida
- Fuzzing of Android Code Via DroidFuzzzer

How much of OWASP top 10 we covered

- M1: Weak Server Side Controls (Attend XWH for this)
- ~~M2: Insecure Data Storage~~
- ~~M3: Insufficient Transport Layer Protection~~
- ~~M4: Unintended Data Leakage~~
- ~~M5: Poor Authorization and Authentication~~
- ~~M6: Broken Cryptography~~
- ~~M7: Client Side Injection~~
- ~~M8: Security Decisions Via Untrusted Inputs~~
- ~~M9: Improper Session Handling~~
- ~~M10: Lack of Binary Protections~~

CTF Challenge

Challenge solutions can be submitted here or after session over email to anant@anantshri.info and cc: antojoseph007@gmail.com

Thank You