

Deep Dive Android

Agenda

- Introduction to Android
- Application Architecture
- Pentesting Android
- Pentesting with Android

Trainer Profile

Name: Anant Shrivastava

Profession: Information Security Consultant.

Certifications: RHCE, CEH, SANS-GWAPT

Speaker / Trainer: Nullcon, ClubHack, c0c0n

Membership: Null and Garage4Hackers

Website: <http://anantshri.info>

My Work : <http://anantshri.info/work/>

Whitepapers: <http://anantshri.info/articles/>

Android:

<https://play.google.com/store/apps/developer?id=Anantshri>

Email : anant@anantshri.info

Agenda

- Introduction to Android
 - Operating System Overview
 - File system Overview
 - Security Model
- Application Architecture
- Pentesting Android
- Pentesting with Android

USP : Android

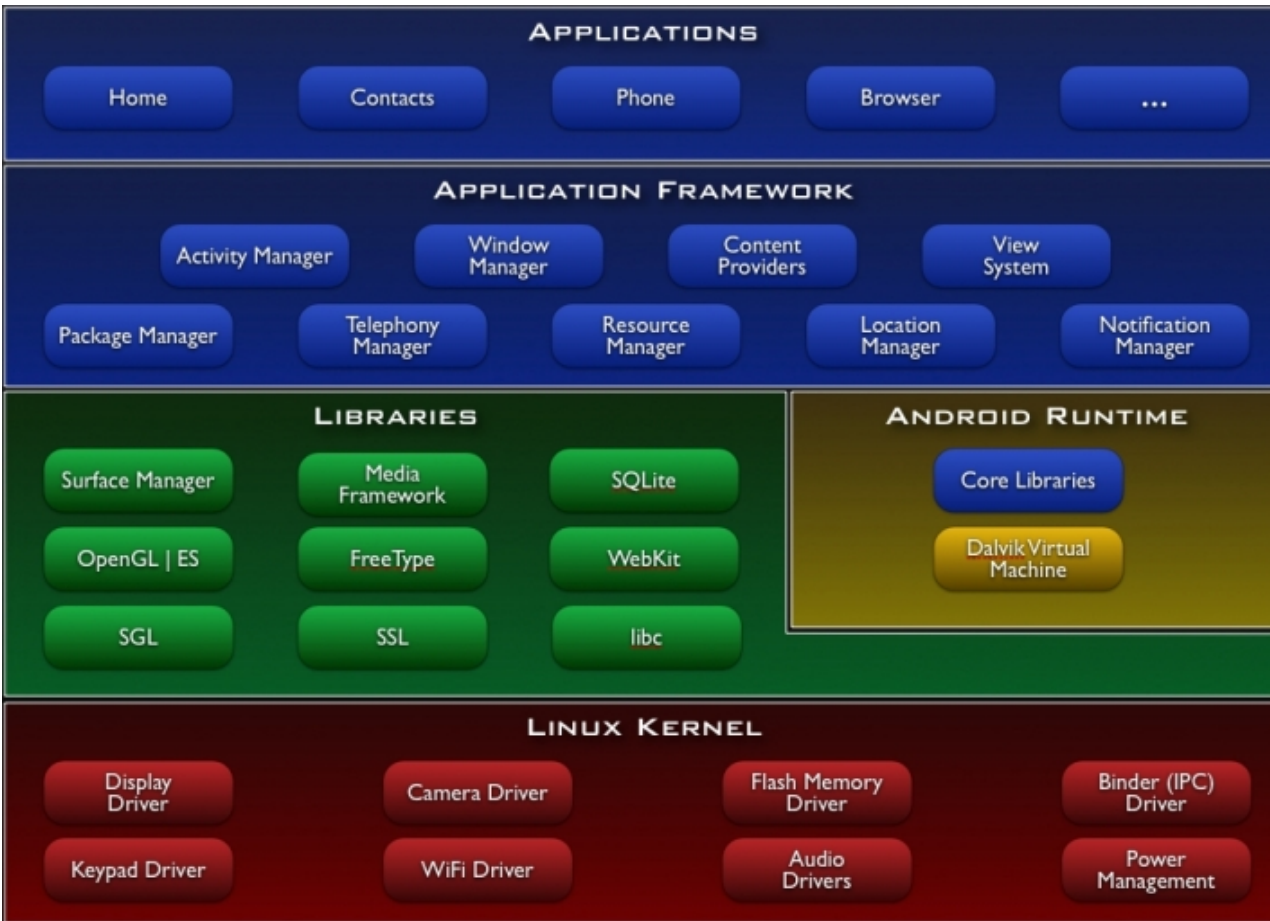
- 56% Smartphone marketshare – Gartner May12
- Sources Available free of cost
- Minimal license cost for developers (25USD).
- Easy to setup development environment.
- Based on Linux
- App-stores filled with large number of apps.
- By 2014, mobile internet to take over desktop internet usage (Source: Microsoft Tag, 2012)

History : Android

- Android Inc. founded in 2003 in Palo Alto, California by Andy Rubin, Rich Miner, Nick Sears and Chris White.
- Acquired in August 2005 by Google Inc. Key employees retained.
- Design continued on a Linux powered mobile device. Marketed by Google to carriers as a flexible and easily upgradable OS.
- On November 5, 2007, a consortium of mobile operators, software companies commercialization companies, semiconductor companies and handset manufacturers formed the Open Handset Consortium, with the stated aim of developing open standards for mobile devices.
- On the same day, they released their first product Android.

Operating System Overview

System Architecture



- A software stack for mobile devices.
- Linux-based kernel (merged back to mainstream in 3.4)
- Middleware, libraries and APIs in C.
- Java-based application framework.
- Custom Dalvik virtual machine with a JIT Java compiler.
- Applications coded primarily in Java.

File System Overview

File System

- Block Devices (available as /dev/block/mtdblock*)

Important Partitions

- /system : OS partition (generally RO)
 - /data : User-Data (Application Storage)
 - /sdcard : SD CARD storage location
- Partition Types
 - /system,/data : Yaffs2 Or ext3/ext4
 - /sdcard : vfat (till 2.3.3)
 - newer nexus devices don't have separate /sdcard rather a folder which is mounted as MTP when connected to desktop

Note : a detailed description of all files and folders is available here

http://anantshri.info/andro/file_system.html

data		2012-09-21	01:53	drwxrwx--x
▶ anr		2012-09-21	01:53	drwxrwxr-x
▶ app		2012-07-14	02:30	drwxrwx--x
▶ app-asec		2012-09-20	23:16	drwx-----
▶ app-private		2012-09-20	23:16	drwxrwx--x
▶ backup		2012-09-20	23:17	drwx-----
▶ dalvik-cache		2012-09-20	23:16	drwxrwx--x
▶ data		2012-09-20	23:17	drwxrwx--x
▶ dontpanic		2012-09-20	23:16	drwxr-x---
▶ drm		2012-09-20	23:16	drwxrwx---
▶ local		2012-09-20	23:16	drwxr-x--x
▶ lost+found		2012-09-20	23:16	drwxrwx---
▶ misc		2012-09-20	23:16	drwxrwx--t
▶ nativebench		2012-07-14	02:26	drwxrwx--x
▶ nativetest		2012-07-14	02:26	drwxrwx--x
▶ property		2012-09-21	01:52	drwx-----
▶ resource-cache		2012-09-20	23:16	drwxrwx--x
▶ ssh		2012-09-20	23:16	drwxr-x---
▶ system		2012-09-21	01:53	drwxrwxr-x
▶ user		2012-09-20	23:16	drwx--x--x
default.prop	116	1970-01-01	05:30	-rw-r--r--
▶ dev		2012-09-21	01:52	drwxr-xr-x
▶ system		2012-07-14	02:30	drwxr-xr-x
▶ app		2012-07-14	02:31	drwxr-xr-x
▶ bin		2012-07-14	02:27	drwxr-xr-x
build.prop	1433	2012-07-14	02:18	-rw-r--r--
▶ etc		2012-07-14	02:31	drwxr-xr-x
▶ fonts		2012-07-14	02:26	drwxr-xr-x
▶ framework		2012-07-14	02:30	drwxr-xr-x
▶ lib		2012-07-14	02:27	drwxr-xr-x
▶ lost+found		2012-09-21	01:52	drw-rw-rw-
▶ media		2012-07-14	02:21	drwxr-xr-x
▶ tts		2012-07-14	02:21	drwxr-xr-x
▶ usr		2012-07-14	02:24	drwxr-xr-x
▶ xbin		2012-07-14	02:26	drwxr-xr-x
ueventd.goldfish	272	1970-01-01	05:30	-rw-r--r--
ueventd.rc	3879	1970-01-01	05:30	-rw-r--r--
vendor		2012-09-21	01:52	lrwxrwxrwx

Security Model Overview

Security model

System level

Unix permission based restriction.

SE Linux (4.3 onwards : Permissive mode in 4.3, enforced in 4.4)

App sandboxing

each application a unique id

Permission Model

Permission need to be taken first time

(AppOps introduced as hidden feature in market and can be leveraged to fine tune the permission model)

SEAndroid

SE Linux port to Android

Demo

- File system navigation using ADB
- Permission model view using
 - ls -l (Long listing)
 - ls -lZ (SELinux Context)
- Permission view for Applications
 - /data/data/
 - /data/app/

Agenda

- Introduction to Android
- Application Architecture
 - Application Components
 - Application Structure
 - The SDK and Android Tools
 - Developing a basic application
- Pentesting Android
- Pentesting with Android

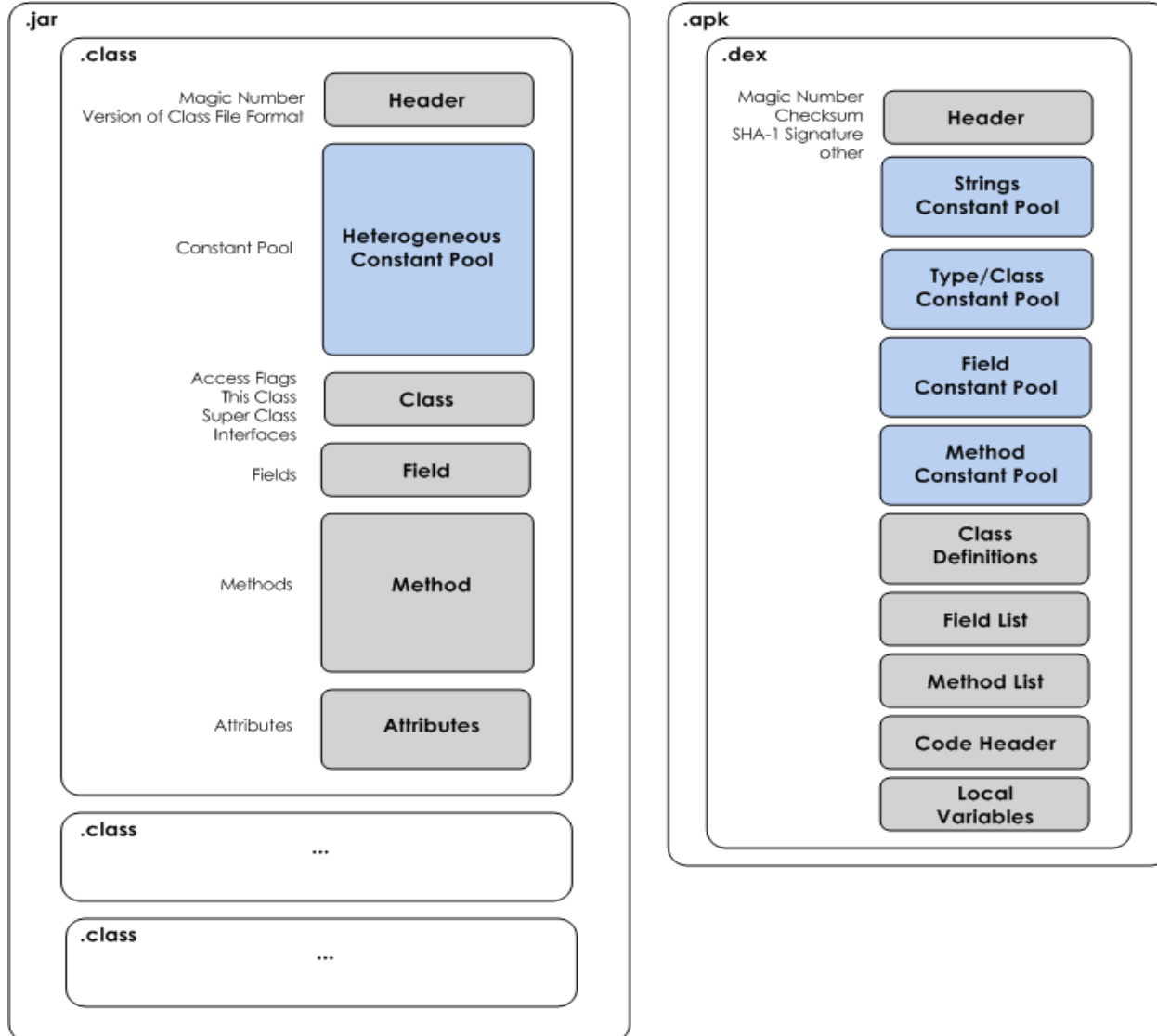
Dalvik Virtual Machine

- Designed and written by Dan Bornstein
- Virtual machine for running android apps
- Android apps written in Java, compiled and converted to Dalvik bytecode format (dex – Dalvik Executable)
- Dalvik bytecode different from Java bytecode
- Dalvik was created to for computers with memory and performance constraints
- Dalvik is a register-based VM as apposed to stack based VM for Java and uses a different instruction set

Dex

- Dex file format details: <http://www.dalvikvm.com/>
- Dex format is optimized for minimal memory footprint
- Dex contains multiple classes per file as opposed to one class per .java file
- Uses shared type-specific constant pools to conserve memory by decreasing redundancy

Dex v/s Class

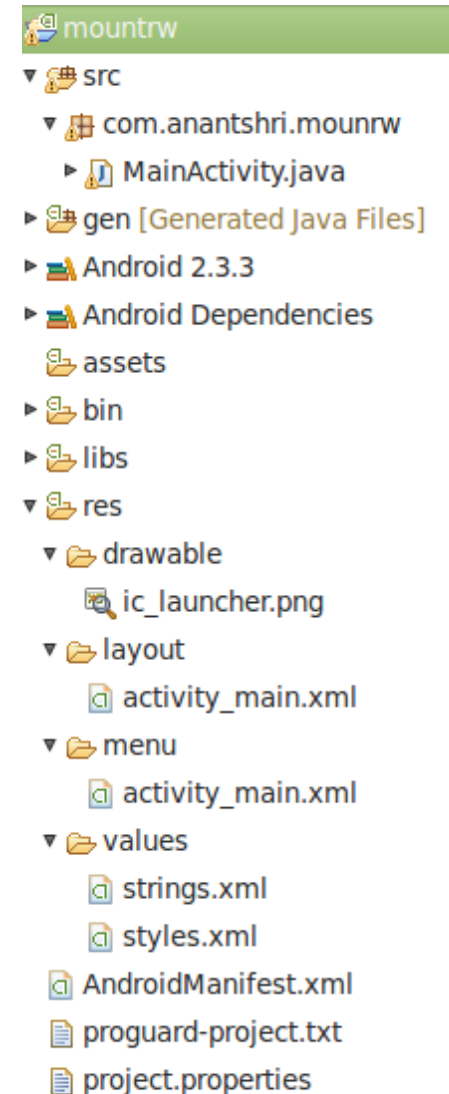


Zygote

- It is the VM process that starts at boot time
- Initializes core library classes and shares them across different forked VM instances.
- Listens on UDS `/dev/socket/zygote` for VMs (app) to fork and launch.
- Also sets appropriate UID/GID and groups based on the arguments and the requester
- Code
 - `dalvik/*`
 - `dalvik/vm/*`
 - `frameworks/base/core/java/com/android/internal/os/Zygote*.java`

Application Structure

- Majorly written in java.
- /AndroidManifest.xml
 - Project Details
 - Permissions, Intents, Recievers
 - Author, Application Name
 - Max and min SDK supported
- /Res/Layout
 - GUI Layout for all Activities
- /src
 - Contains the whole Java source code
- /res/drawable
 - Icons and images
- /res/menu
 - Right click Menu entries
- /res/values
 - Static values to be used in Application



App components

- Activity
- Intent
- Service
- AndroidManifest.xml

Activities

- UI component for one focused task
- Usually a single screen in your application
- Stack based approach where visible activity/screen is topmost activity on stack.
- Activity association is defined in the `AndroidManifest.xml`

Activities

Main Activity
Java Code

```
package nullcon.xah.test2;

import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {

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

Android
Manifest
Definition

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="nullcon.xah.test2" android:versionCode="1" android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <application android:icon="@drawable/ic_launcher" android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity" android:label="@string/title_activity_main" >
        </activity>
    </application>
</manifest>
```

Intent

- Intents == Operations / Actions
- Defined in Manifest (AndroidManifest.xml)
- application → activity → intent-filter

Intents Sample

Main Activity plus Launcher Entry

```
<action android:name="android.intent.action.MAIN" />  
<category android:name="android.intent.category.LAUNCHER" />
```

Registering yourself as browser

```
<activity android:name=".BrowserActivitiy" android:label="@string/app_name">  
  <intent-filter>  
    <action android:name="android.intent.action.VIEW" />  
    <category android:name="android.intent.category.DEFAULT" />  
    <data android:scheme="http"/>  
  </intent-filter>  
</activity>
```


Service

- Background Jobs (No UI)
- Long running process. No effect on response.
- Declare Service application → service

```
<service
  android:name="MyService"
  android:icon="@drawable/icon"
  android:label="@string/service_name"
  >
</service>
```

- extends IntentService (one-time) or Service (Multiple)
- protected void onHandleIntent(Intent intent)

Android Manifest.xml

- XML structure defining properties including
 - Activities
 - Intents
 - User-permissions
























Android Manifest.xml

- `<uses-permission />` - list of **required** permissions from OS.
- `<permission />` - list of permission calling party must have.
- `<uses-sdk />` - min max and target sdk versions.
- `<uses-configuration />` - hard and software configuration
- `<uses-feature />` - specific features (filters)
- `<application>`
- `<activity>` - activities provided by the application
- `<intent-filter>` - various intents raised by application
- `<service>` - background activity.
- `<receiver>` - catch holder for system / broadcast intents.

Android SDK, NDK and Tools

Android SDK, NDK and tools

- SDK – Software Development Toolkit.

 Name	API	Rev.	Status
▼ <input type="checkbox"/>  Tools			
<input type="checkbox"/>  Android SDK Tools		20.0.3	 Installed
<input type="checkbox"/>  Android SDK Platform-tools		14	 Installed
▼ <input type="checkbox"/>  Android 4.1 (API 16)			
<input type="checkbox"/>  <i>Documentation for Android SDK</i>	16	2	 <i>Not installed</i>
<input type="checkbox"/>  SDK Platform	16	2	 Installed
<input type="checkbox"/>  <i>Samples for SDK</i>	16	1	 <i>Not installed</i>
<input type="checkbox"/>  ARM EABI v7a System Image	16	2	 Installed
<input type="checkbox"/>  <i>Intel x86 Atom System Image</i>	16	1	 <i>Not installed</i>
<input type="checkbox"/>  <i>Mips System Image</i>	16	1	 <i>Not installed</i>
<input type="checkbox"/>  <i>Google APIs</i>	16	2	 <i>Not installed</i>
<input type="checkbox"/>  <i>Sources for Android SDK</i>	16	2	 <i>Not installed</i>

Android SDK, NDK and tools

- NDK – native development kit
 - Allows development of components in C / C++.
 - allows reuse existing code libraries.
 - possibly increased performance.
- Typical usage
 - Self-contained,
 - CPU-intensive operations,
 - Signal processing,
 - Physics simulation
 - Games

Tools provided by SDK / NDK

- GCC compiler for ARM
- Tools/android → sdk/avd manager
- Tools/ddms → debugging tool
- Tools/emulator → emulator executable
- Platform-tools/adb → debug bridge
- Platform-tools/fastboot → flashing utility

ADB : Android Debug Bridge

ADB has ability to perform operations on android device remotely.

Adb client -> adb server -> adb daemon

(Development machine) -> (device)

Some common usage

push : Push data inside Device

pull : Pull data from Device, file / folder

install : Install software in device. (apk)

logcat : realtime debug messages

With Recent version's adb connects only to verified devices. (verification taken on first connect)

Alternate Android VirtualMachines

- Geanymotion (recommanded)
 - Virtualbox based x86 version of Android
 - OpenGL rendering supported (speed++)
 - Registration Required
- Jar of Beans (community project)
- BlueStack (non compatible with Android SDK)

Hello World App

- Exercise
- A simple application which prints hello world using a label with eclipse

Agenda

- Introduction to Android
- Application Architecture
- Pentesting Android
 - Android Tamer & Environment Setup
 - Black Box PT
 - Reverse Engineering
 - Rooting basics
 - Understanding Pentesting Frameworks
- Pentesting with Android

Android Tamer

What is Android Tamer

- VM / Live ISO / Installable environment.
- Specific focus on Android Security.
- First Launched in Dec 2011 @ Clubhack 2011.
- Second Release with large set of enhancement
- Provides the most extensive Collection of tools for android security.

More

- Based on Debian 7.
- Environment customized to keep all tools in path.
- Browser loaded with Pentesting toolkit + Bookmarks
- All non essential software's removed. But could be added once installed on local machine.
- Next updates will be through repositories only.

Tools List

- ROM Modding
 - Rom kitchen
 - Flashing utility
- Rooting
 - Zergrush (GB)
 - adb restore (ICS / JB)
 - APK based rooting options
- Development
 - Eclipse + ADT
 - SDK + NDK
- Pentesting
 - OWASP ZAP proxy
 - BURP proxy
 - Firefox + pentest plugins
- RE and malware Analysis
 - Drozer (aka Mercury)
 - Androguard
 - Dex2Jar
 - JD-GUI
 - APKtool
 - Baksmali / smali
 - Bulb Pentesting Framework
- Wireless Capture
 - Wireshark
 - Tcpdump
- Forensics
 - AF logical OSE
 - Sleuthkit

Rooting kits

- ZergRush – Valid for GingerBread
- Superoneclick
 - ZergRush
 - psneuter
- Gingerbreak
- Z4root
- superandRoot

ROM Modding

- ROM Kitchen
 - Allows to modify existing ROMs add or remove content or modify settings (ro.secure=?)
- Flashing Utilities
 - Flashtool :SONY Xperia Series
 - Heimdall : SAMSUNG Galaxy Series
 - SBP_flash : MOTOROLA phones
- Single Click ADB SHELL and LOGCAT access

Reversing toolset

- JD-GUI
- JED
- DEX2JAR
- Smali / Baksmali
- APKtool

Malware Analysis

- DroidBox
- Mercury
- Androguard

Agenda

- Introduction to Android
- Application Architecture
- Pentesting Android
- **Pentesting with Android**
 - Setting up the environment
 - Various tool usage
 - Writing custom tool in android

Pen Testing

Agenda

- Understanding Mobile Security Issues
- Setup Pen testing environment

Mobile Security Issues

Agenda

Data / Activity Sniffing

Unauthorized access to telephony layer (dialing, sms etc)

Unauthorized network access

Unsafe Data at transit / rest (XML / SQLite)

Hardcoded values Password / key / salt Untrusted inputs / intents

Data leakage Side channel

Information Disclosure

Logic / Time Bomb

UI impersonation

Rooting

Application Security update cycles

OS level security updates

SQLi

Click / Tap jacking

Playing with Javascript

Data / Activity Sniffing

- Data and activities could be monitored on real time
 - Messaging (SMS and Email)
 - Audio (calls and open microphone recording)
 - Video (still and full-motion)
 - Location
 - Contact list
 - Call history
 - Browsing history
 - Input
 - Data files
- Example :Secret SMS Replicator

Access to telephony layer

- Malware now a days are targeting SMS / Calls.
- Premium SMS / Call -> high charge
- USSD based purchases
- Location specifics

- Example
 - FakePlayer : Premium SMS sending app

Unsafe data at transit

- Data Transmitted using insecure Channels.
 - HTTP
 - FTP
 - Unsigned XML
- Protection : Use HTTPS
- Ever Heard of sslstrip ?

Hardcoded values

- Reverse Engineer the source Code and check for hardcoded values
 - Db connection strings
 - Api keys for third party apps.

Side channel leakage

- Data leakage occurring through residual data like cache or temp files or keyloggers.
- Root Cause
 - Bad coding practice from developer.
 - Inherent OS specific features.
- Identification Techniques
 - Before launching application take a snapshot of system. Launch application perform all operations and then again take a snapshot. Find the change in system look for residual file and data specially in temporary folders.
- Action / Remediation:
 - Avoid web data caching by setting proper headers.

Information disclosure

This risk is based on the fact that many developers hardcode the API or password, also lots of applications are now shifting business logic to client side.

- Root Cause
 - Most of the web applications could be easily reverse engineered.
 - Hardcoded API Keys, passwords and other sensitive information.
 - Embedding business logic in client code.
- Identification Techniques
 - Decompile application and check if some hardcoded values are visible and if business logic could be altered. (especially in case of financial applications)
- Action / Remediation:
 - Values should not be hardcoded.
 - Business logic should be kept separate at server side only.

Logic / Time Bomb

- Code to be activated
 - Specific dates
 - connecting to a network
 - Dialing a number
 - Receiving an sms

 - You can think of some more

UI Impersonation

- Application Posing as a known legitimate Apps or websites.
- Prevention : Google app store has started rejecting and banning applications performing such tactics. (other app stores ??? And side loading)

Rooting

- Devices are by default running in a restricted user environment (refer permissions section)
- Root user holds ultimate authority over system.
- All released android versions are vulnerable.
- Exploits used to gain root access are
 - OS based (Os level binary flaws)
 - Devices specific files

Application Updates

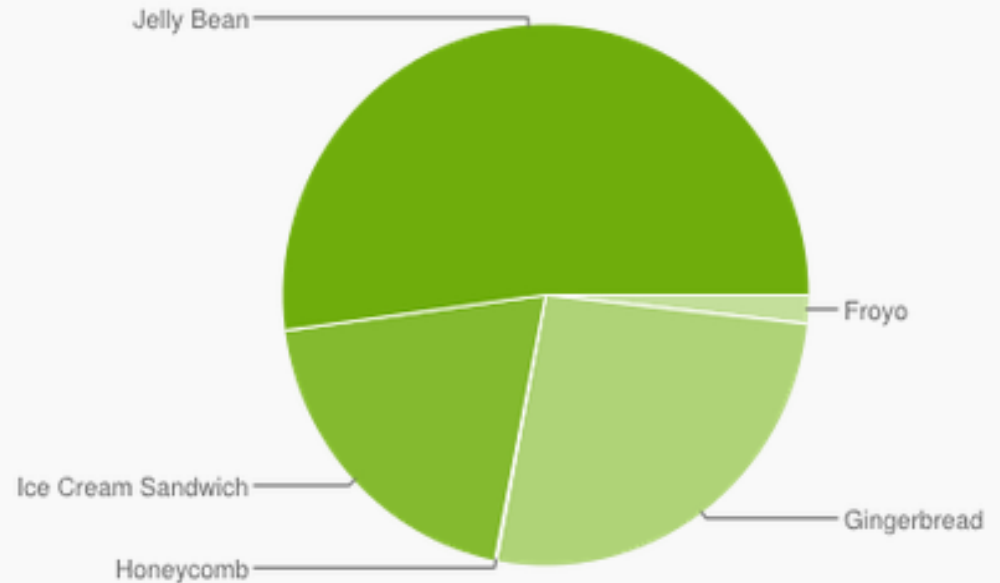
- Application Updates are send over the air.
- If update happening over Wifi sniffing is easy.
- Google play store may apply security. But not all stores are having all securities in place.
- Play store is only available with Google authorized phone manufacturers

OS level updates

- Android updates are largely carrier and manufacturer dependent.
- Google updates AOSP others (manufacturers and carriers) download and distribute.
- Only independent devices as of now
 - Google nexus series

Current OS distribution

Version	Codename	API	Distribution
2.2	Froyo	8	1.7%
2.3.3 - 2.3.7	Gingerbread	10	26.3%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	19.8%
4.1.x	Jelly Bean	16	37.3%
4.2.x		17	12.5%
4.3		18	2.3%



*Data collected during a 7-day period ending on November 1, 2013.
Any versions with less than 0.1% distribution are not shown.*

SQLi

- Large amount of application have backend running on a web server + db server backend.
- So tradition SQLi still works the deal is to find the backend.

Click / Tap jacking

- Clickjacking for mobile is Tap jacking.
- Similar techniques like clickjacking.
- Transparent frame placed on top of legitimate looking button's.
- Could be used to earn ad revenue on clicks.

Javascript

- Javascript is the new playground.
- Iframes and various javascript calls are hard to detect on mobile browser.
- With HTML5 in picture now the vectors availability has increased multifold.

Setup Pentesting Environment

Setup

- Static Analysis Tools
 - Reversing the apk
 - Dex2jar + Jd-gui / jad
 - Smali
- Network traffic interception
 - OWASP ZAP
 - Burp suite
- Backend and frontend scanning
 - Emulator as isolated environment.
 - Server side scanning (nikto, w3af, nmap)

Reversing the APK

- APK == JAR == TAR
- .dex ~~~ .classes merged
- Simplest process
 - Unzip
 - Dex2jar convert .dex to jar file
 - Jd-gui / jad to decompile jar.
 - Apktools : extract resources and correct binary xml

Network traffic interception

- Using emulator or device define proxy.
- Emulator `–http-proxy http :// 127.0.0.1:8080`
`–avd <name>`
- Settings -> networks -> access point -> proxy
host -> port
- For emulator localhost / base machine's ip =
10.0.2.2

Network interception cont...

- Issues in listed approach
 - 1) SSL traffic most of the time will not be intercepted and app will crash with connection failure due to invalid certification.
 - 1) Solution is to import the certificate of the proxy server.
 - 2) Export proxy cert from application
 - 3) Adb push .cer /sdcard/
 - 4) Settings -> security -> install from sdcard
 - 1) Will have to set a pin for device.

Network traffic interception cont..

- Application traffic not proxified

For emulator's

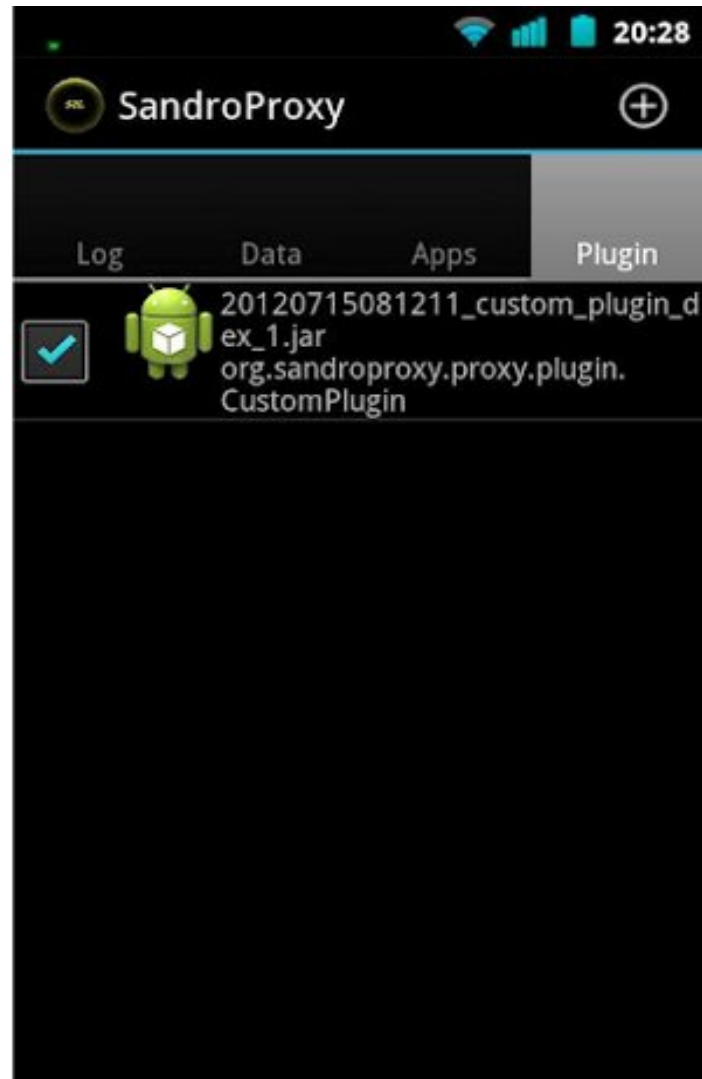
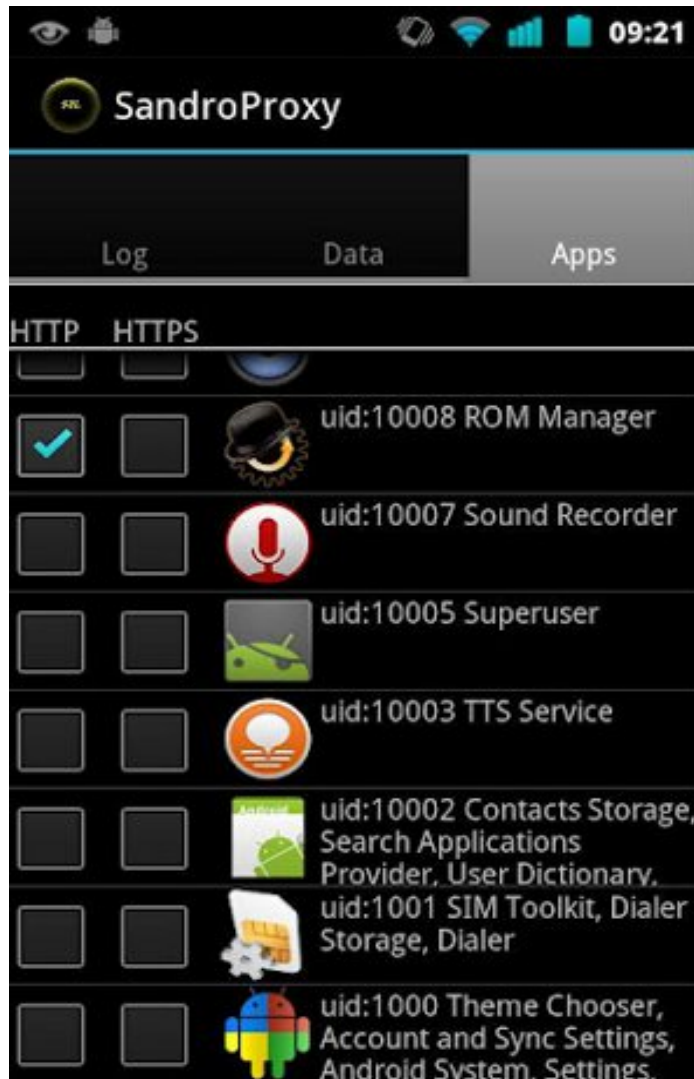
this is applicable till 2.3.3 emulator.

Tested above settings on 4.0 and 4.1 series and all apps are by default proxified.

IPTables Based App level intercept

- Android runs each app with individual userid
- IPTables can be used to set rules for each user.
- Combine these two and you can do all the traffic interception you want.

SandroProxy



Tests at a Glance

device level tests

- Data by app stored in
 - /data/data/<app_package_name>
- /sdcard content.
- Look for xml or db's for unencrypted data
- File system could be scanned for changes before and after install / usage / removal of application

Backend Scanning

- During Network interception you can easily identify the backend server ip's / url's
- nmap,w3af,nikto scan on backend could be made to assess it.
- Server side flaws need not be web flaws only, any service running of server could be our potential target.

Excercise

- App Protector.
 - Protects your phone specific functions from unauthorized access
 - Or does it.
 - Refer : /data/data/com.ruimaninfo.appprotect/
- Defender
 - A simple application where you can play and earn powers at offline level and then compete with opponent online.

Pentesting Frameworks

- Mercury / Drozer
- AFE (Android Framework for Exploitation)
- Smartphone Pentest Framework

Pentesting Through Android

Available Toolset

- DroidSheep
- Dsploit
- Interceptor
- Network Discovery
- Shark
- Network Tools
- zAnti

Pentesting through Android

- Setup Environment
 - SL4A
 - Py4a
 - Pl4a
 - setup standalone shell for them
- Porting Existing tools
- write basic scripts for python to perform basic operations
 - username password bruteforce attack
 - task automation using python
 - username enumeration wordpress script
- creating packages from scripts

Sample Scripts and Demo

- Python Based Wordpress Username Enumeration
- Python based bruteforcing tool

Recap

We Learned

- What is Android
- Internal Working
- Application development
- Vulnerabilities
- How to Pentest an android application
- Penetration Testing Using Android

Any Questions

Thank You