Mobile Security

Security in Networked Computing System

Andrea Saracino

andrea.saracino@iet.unipi.it

Outline

- Mobile Security
  - Evolution in mobile security.
  - Challenges and Issues.
  - Mobile Security Attacks.
    - Sybil.
    - Geolocation.
    - Data Spoofing.

Outline (2)

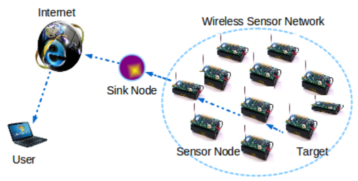
- Mobile Application Security
  - Malware and other threats
    - Malware Evolution
    - Common Malware
    - Trojanized apps
  - Android System and Security
    - Android Programming
    - Security Libraries
    - Android Security Attacks

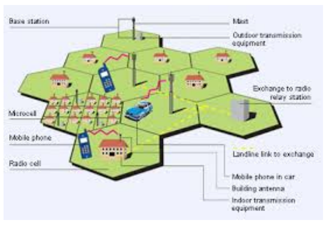
Mobile Security

- **Mobile Security** is a computer security branch studying methodologies to ensure all the security requirements in <u>mobile</u> and <u>distributed</u> systems.
- **Mobile System:** A system composed of roaming agents.
- **Distributed System:** System with several interacting autonomous agents that cohordinate themselves to provide and/or receive a service.

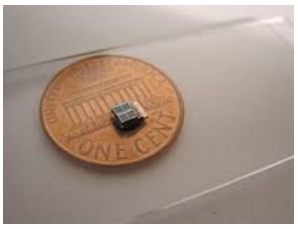
Mobile Security (2)

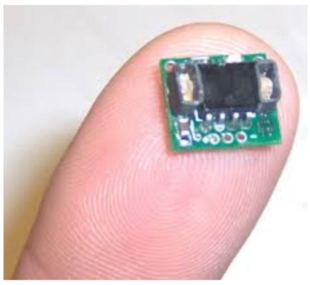
- Examples:
  - Distributed Computing
  - Wireless Sensor Networks
  - Participatory Sensing Systems
  - Peer to Peer (P2P) file sharing
  - Cellular Networks




Base station
Mast
Outdoor transmission equipment
Exchange to radio relay station
Microcell
Mobile phone
Radio cell
Landline link to exchange
Mobile phone in car
Building antenna
Indoor transmission equipment

Mobile Security: How it was



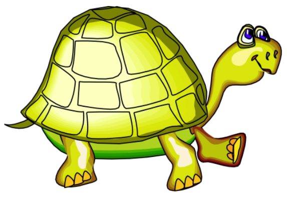


Mobile Security: How it was (2)

- Mobile devices have always been considered as ***constrained***, because of:
  - Limited Battery Power
  - Low Computational Power
  - Small Physical Memory
- Security imposes a consistent overhead
  - Asymmetric Criptography
  - Key Storage, distribution and retrieval

Mobile Security: How it was (3)

- Other examples of constrained devices are the more recent RFID (passive devices).
- Applying classical security techniques on constrained device is non feasible.
- Mobile security aims at solving these issues through:
  - Lightweight cryptographic functions
  - Distributed Computation
  - Low consumption security protocols (Bluetooth)



Mobile Security: How it is

Mobile Security: How it is (2)

- Mobile Security current main focus is on the so called *new generation mobile devices,* which are smartphones and tablets.
- Smartphones and tablets have several features and functionalities that makes them extremely more complex, compared to legacy mobile phones

VS

Mobile Security: How it is (3)

- There are several motivations that fosters attackers to target new generation mobile devices:
  - Private Data stored.
  - Possibility to access charged services (Phone Calls, SMS…).
  - High distribution and popularity.
  - High connectivity.
  - Possibility to attack several levels
  - Too fast evolution; security can't catch up!

Mobile Security: How it is (4)

- More features of smartphones and tablets:
  - Continuously connected to the Internet
    - Data Network (Provider)
    - WiFi
  - It is possible to install mobile applications.
  - Several accounts for different services (Google, Facebook, Twitter, Istant Messaging, Banking Apps…)

Mobile Security: How it is (5)

- Summarizing smartphones and tablets have:
  - High Computational Power
  - Several Connectivity Interfaces
  - High Customizability
- Thus, they can provide several more services compared to legacy mobile phones.
- However, at the same time, they are more vulnerable to security attacks.
  - More attack vectors
  - Greater attacker motivation
  - Possibility of providing malicious software hidden inside other applications.

Constrained Devices?

- CPU & ChipsetIntel® Atom N270 (1,6 GHz) (Single Core)
- Memory 1GB
- Camera 1.3M Pixel
- Weight 1.45kg

- CPU: Qualcomm Snapdragon™ 800, 2,26 GHz (4 core)
- Memory 2GB
- Camera 8 M Pixel + 1,3M Pixel
- Weight 130 g

nexus5

New Challenges

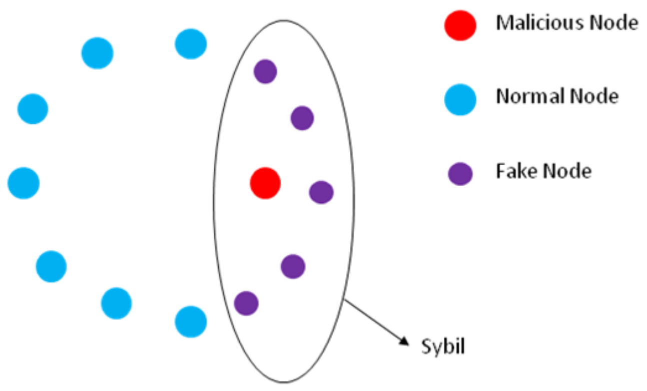
- Current mobile security challenges have to deal with constrained devices only for a subset of applications.
- New kinds of security attacks are exploiting both the *features* of new generation mobile devices, and the dynamic and distributed *nature* of mobile systems.
- Instead of taking advantage of the reduced capability of performing cryptographic operations, attacks are brought directly at the application level.

Sybil Attack

- Security attack toward distributed P2P systems.
- Let's consider a distributed system where each user (or agent) is considered a peer.
- Peer exchange information between them, through a network, in order to receive or provide a service.
- Each peer is identified, while active in the system, by an identifier (ID).

Sybil Attack (2)

- Definition: A ***sybil attack*** is performed when a malicious user pretends to be multiple peers in the system by creating fake identities or stealing existing ones.



Sybil Attack (3)

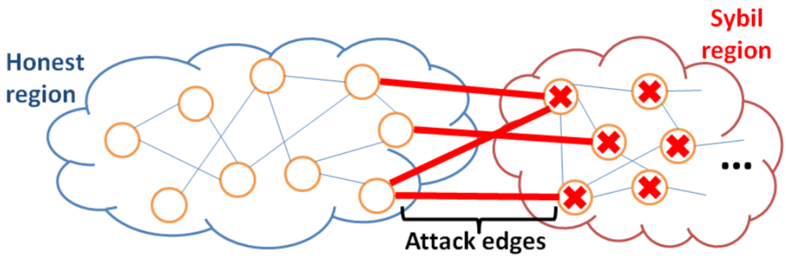
- Through the sybil attack, a single node is seen as a group on nodes, called *Sybil*, because it behaves like as she has a multiple personality (identities).
- When an identity is stolen (more common than creating false identities), the owner of that identity is generally silenced by the sybil.
  - DoS attack
  - MITM
- Other nodes believe they are talking with the silenced node, but exchanged information are controlled by the Sybil.

Sybil Attack (4)

- Applied in several systems:
  - Voting System
  - Reputation System (E Bay Feedbacks)
  - Traffic Redirection (eavesdropping)
  - Social Network
    - Creation of fake identities.
    - Stealing existing identities.

Sybil Attack (5)

- The network partition:
  - Attacker has direct control on some network nodes, called edge nodes.
  - Sybil create a network of fake or stolen identities, controlling all the messages exchanged in this network partition.



Sybil Attack (6)

- The attack is effective in those environment where there is a loose control on peers identity.
- Solutions:
  - Registration authority.
  - Identity protection (password).
  - Binding between identity and the owner: each physical peer should only have one identity.
    - Bind the identity to an e-mail address.
    - Bind the identity to the IMSI-IMEI of a mobile device.

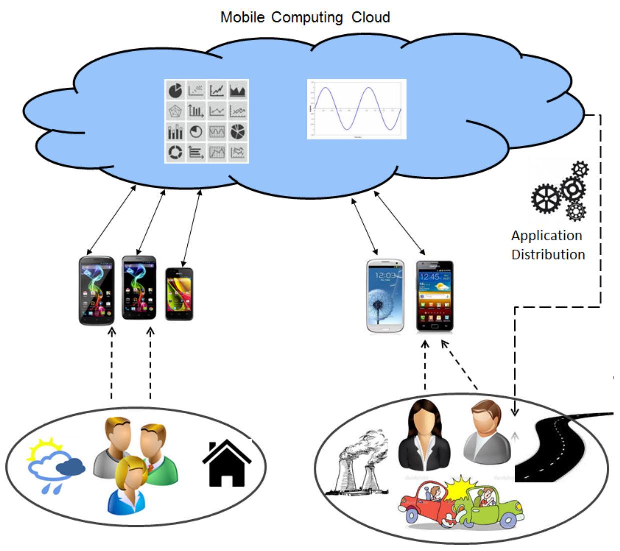
Sybil Attack (7)

- The Sybil attack has been proposed in 2002 (JR. Douceur, IPTPS 2002).
- Recent distributed systems have been designed to be resilient to the Sybil Attack.
  - Account, authentication, password, security question…
- Today still exists some extremely popular mobile applications relying on loose identity check mechanisms, thus prone to the Sybil Attack

Data Spoofing Attack

- Let's consider a mobile system in which mobile agents collect data on the surrounding environment and share it with the rest of the system.
- Smartphones are equipped with several sensors to collect data from the environment, making this kind of application desirable.
- Several names have been given to these applications:
  - Participatory Sensing
  - Crowd Sensing
  - Distributed Monitoring
  - …

Data Spoofing Attack (2)



Data Spoofing Attack (3)

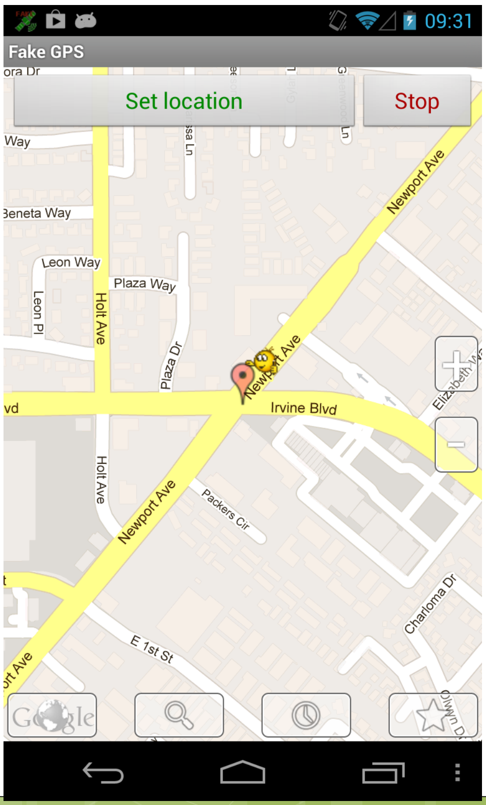
- Verification of data correctness?
  - It is not performed by default.
  - Users are supposed to provide correct data.
  - Users can also be malicious.
- Inserting false data in the network can cause severe misbehaviors.

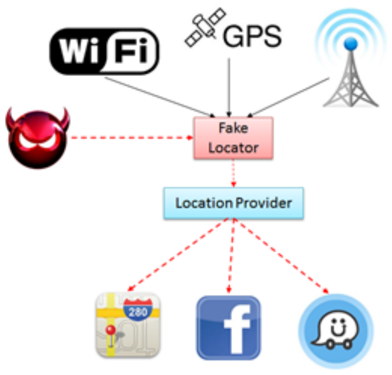
Location Spoofing

- An example of Data Spoofing is the *Location Spoofing attack*.
- Let's consider a system where mobile agents share pieces of information related to their geographical location such as:
  - The amount of car traffic
  - The pollution level of a specific area
  - Weather and temperature
  - The location itself
- In this application, providing a false location means to provide data which is completely false.

Location Spoofing (2)

- Performing this attack is really simple with both Android Smartphones and iPhones (jailbroken).
- ***Fake Locator*** is a group of software available also on the official application marketplaces. Fake Locator allows the user to choose a position on the world map. Then, this position is shared with all the applications running on the device.





Location Spoofing (3)

- Fake locator main purpose is user's fun, but the location obfuscation is generally allowed for privacy reasons.
- However, when fake locator is active, all the location based applications will misbehave, since the user is providing false data.
  - Such misbehaviors may also affect other users.
  - Privacy is a right of the user, but providing the correct location to participatory sensing application is a duty.

Location Spoofing (4)

- Solutions:
  - Selective obfuscation to preserve privacy without providing a fake location to all apps.
    - Location precision granularity.
  - Exploiting control elements to periodically check the real location of a user, comparing it with the one provided.
  - Use of a reputation index to identify users that lies on their location.
    - Reward proportional to the reputation (incentive).

More on Location Spoofing Attack

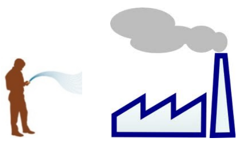
- A location based piece of data can be identified through the following tuple:

  <x,y,d>

- *x* and *y* are respectively latitude and longitude where the piece of information *d* has been collected.
- Removing the bound between *x,y* and *d* means to provide false data.
- Data *d* is correct but if the location is changed, the system will read and use wrong data, causing misbehaviors.

More on Location Spoofing Attack (2)

- Example: Pollution Monitoring
- Users collect data on the pollution level of different metropolitan areas.
- If the pollution level in an area is above a threshold, actions are taken to limit the pollution level.
  - Closing roads to car traffic.
  - Temporarily shutting down factories to reduce emissions.
- Such actions cause several inconvenience which may also imply monetary losses.
- A user residing in an area extremely polluted may send readings, while faking the position in a low pollution area.
  - This will trigger unneeded actions to reduce the pollution in that area.



Mobile Application Security

- Currently the greatest threat in mobile security is brought by Malicious Applications.
- More than 90% of mobile security attacks are based on malicious software (malware) installed on user's device.

- Targets of the attack:
  - Private data
    - SMS messages
    - Contacts
    - IMEI code
    - Username and Passwords (social network, home banking).
  - User Money
    - Leaking credit from SIM card.
    - Hidden subscription to premius services.

Mobile Applications (App)

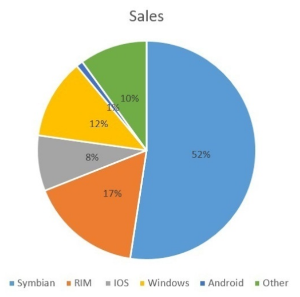
- New generation mobile devices can be customized installing mobile applications.
- Mobile applications (apps) are developed by third party developer.
- Apps can be malicious (malware).

PERCHÉ LI DEVI CHIAMARE APP??! PER FARLI SEMBRARE PIÙ SBARAZZINI??
BEH SI TENDE AD UNIFORMARE I LINGUAGGI, CON LE TECNOLOGIE INTEGRATE DEGLI SMARTPHONE...
MA IO NON HO COMPRATO UN CAZZO DI I-PHONE!
...
IO HO COMPRATO UN PC FISSO GROSSO QUANTO UN CATAFALCO CHE PE' STA' IN CASA DEVE PAGA' L'AFFITTO DALLO SPAZIO CHE OCCUPA!
È PIÙ GROSSO DE NA MUCCA, VOGLIO CHE MUGGISCE, NO CHE CI ABBIA LE APP!!
MUUU. mungimi calcare...
Image from zerocalcare.it

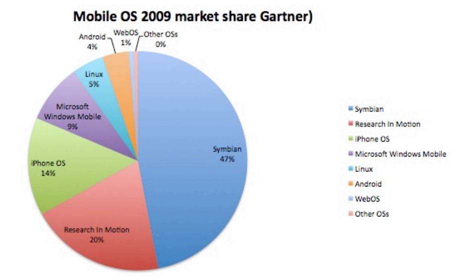
A bit of history

- First malware for mobile device?
- Target OS: **Symbian**
- Year: **2000**
- Effect: **Remove Third Party App**

A bit of History (2)

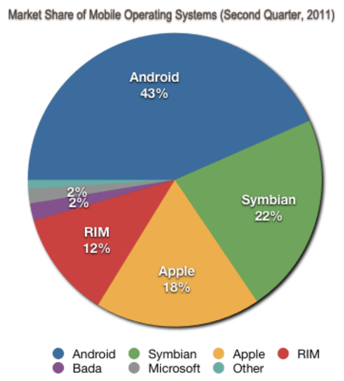


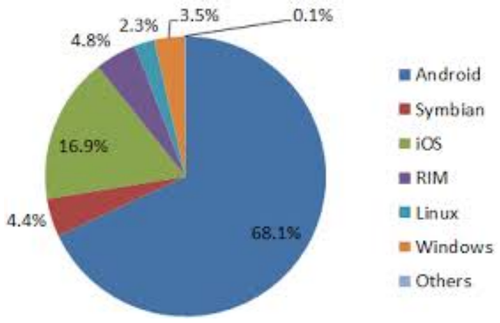
2008



2009

A bit of history (3)

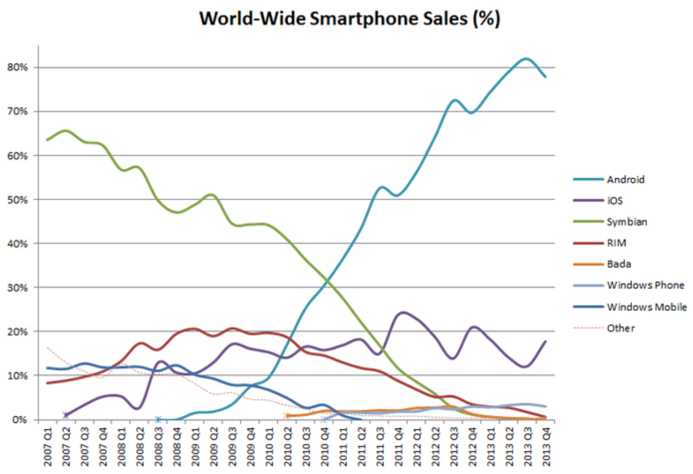


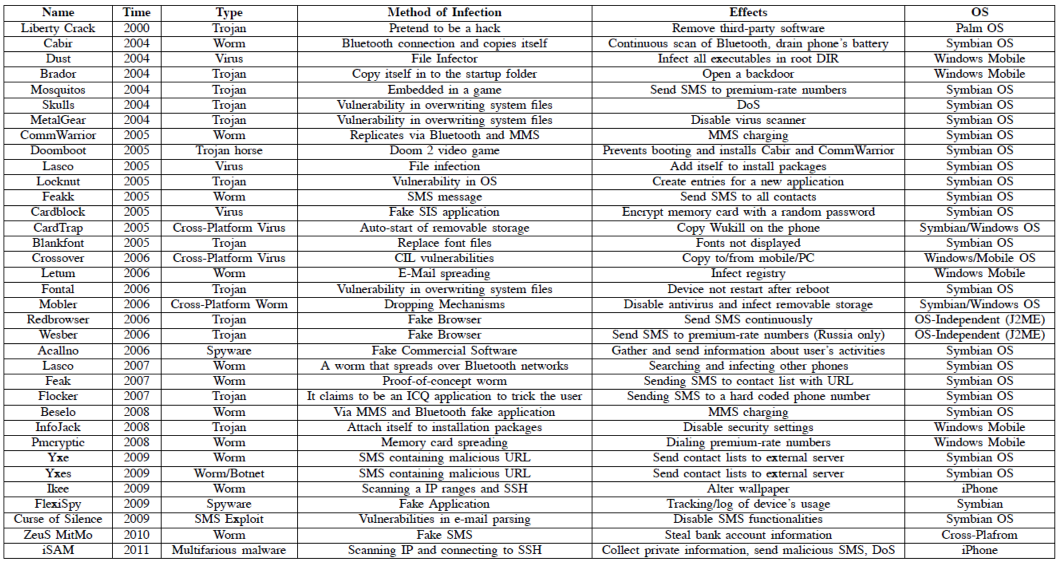


2011

2012

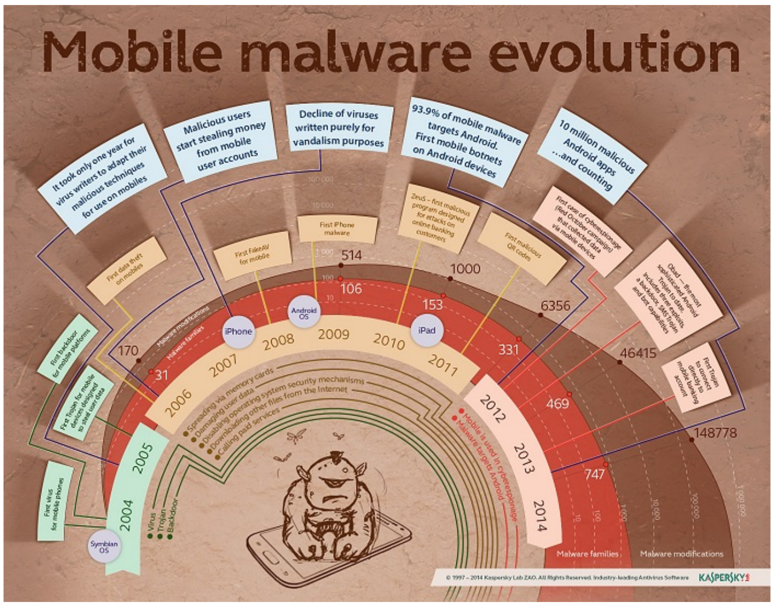
A bit of history (4)





Name	Time	Type	Method of Infection	Effects	OS
Liberty Crack	2000	Trojan	Pretend to be a hack	Remove third-party software	Palm OS
Cabir	2004	Worm	Bluetooth connection and copies itself	Continuous scan of Bluetooth, drain phone's battery	Symbian OS
Dust	2004	Virus	File Infector	Infect all executables in root DIR	Windows Mobile
Brador	2004	Trojan	Copy itself in to the startup folder	Open a backdoor	Windows Mobile
Mosquitos	2004	Trojan	Embedded in a game	Send SMS to premium-rate numbers	Symbian OS
Skulls	2004	Trojan	Vulnerability in overwriting system files	DoS	Symbian OS
MetalGear	2004	Trojan	Vulnerability in overwriting system files	Disable virus scanner	Symbian OS
CommWarrior	2005	Worm	Replicates via Bluetooth and MMS	MMS charging	Symbian OS
Doomboot	2005	Trojan horse	Doom 2 video game	Prevents booting and installs Cabir and CommWarrior	Symbian OS
Lasco	2005	Virus	File infection	Add itself to install packages	Symbian OS
Locknut	2005	Trojan	Vulnerability in OS	Create entries for a new application	Symbian OS
Feakk	2005	Worm	SMS message	Send SMS to all contacts	Symbian OS
Cardblock	2005	Virus	Fake SIS application	Encrypt memory card with a random password	Symbian OS
CardTrap	2005	Cross-Platform Virus	Auto-start of removable storage	Copy Wukill on the phone	Symbian/Windows OS
Blankfont	2005	Trojan	Replace font files	Fonts not displayed	Symbian OS
Crossover	2006	Cross-Platform Virus	CIL vulnerabilities	Copy to/from mobile/PC	Windows/Mobile OS
Letum	2006	Worm	E-Mail spreading	Infect registry	Windows Mobile
Fontal	2006	Trojan	Vulnerability in overwriting system files	Device not restart after reboot	Symbian OS
Mobler	2006	Cross-Platform Worm	Dropping Mechanisms	Disable antivirus and infect removable storage	Symbian/Windows OS
Redbrowser	2006	Trojan	Fake Browser	Send SMS continuously	OS-Independent (J2ME)
Wesber	2006	Trojan	Fake Browser	Send SMS to premium-rate numbers (Russia only)	OS-Independent (J2ME)
Acallno	2006	Spyware	Fake Commercial Software	Gather and send information about user's activities	Symbian OS
Lasco	2007	Worm	A worm that spreads over Bluetooth networks	Searching and infecting other phones	Symbian OS
Feak	2007	Worm	Proof-of-concept worm	Sending SMS to contact list with URL	Symbian OS
Flocker	2007	Trojan	It claims to be an ICQ application to trick the user	Sending SMS to a hard coded phone number	Symbian OS
Beselo	2008	Worm	Via MMS and Bluetooth fake application	MMS charging	Symbian OS
InfoJack	2008	Trojan	Attach itself to installation packages	Disable security settings	Windows Mobile
Pmcryptic	2008	Worm	Memory card spreading	Dialing premium-rate numbers	Windows Mobile
Yxe	2009	Worm	SMS containing malicious URL	Send contact lists to external server	Symbian OS
Yxes	2009	Worm/Botnet	SMS containing malicious URL	Send contact lists to external server	Symbian OS
Ikee	2009	Worm	Scanning a IP ranges and SSH	Alter wallpaper	iPhone
FlexiSpy	2009	Spyware	Fake Application	Tracking/log of device's usage	Symbian
Curse of Silence	2009	SMS Exploit	Vulnerabilities in e-mail parsing	Disable SMS functionalities	Symbian OS
ZeuS MitMo	2010	Worm	Fake SMS	Steal bank account information	Cross-Plafrom
iSAM	2011	Multifarious malware	Scanning IP and connecting to SSH	Collect private information, send malicious SMS, DoS	iPhone

Mobile Malware Evolution

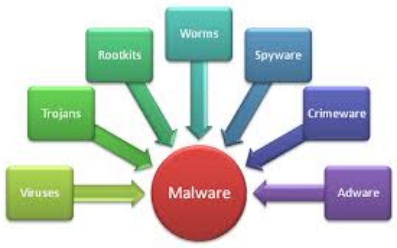


Mobile Malware Evolution (1)

- First mobile malicious applications were mainly created as research products.
  - Proof of concepts to spot system vulnerabilities.
- Current malware aims at generating a revenue for the attacker.
  - Directly stealing money (Zeus).
  - Stealing private information which can be sold for money.
  - Subscription to premium services.

Common Malware Type

- Rootkit: Application that exploits system vulnerabilities to acquire root (super user) privileges.
- Spyware: Collect information on the device user and her behavior.
- SMS Trojan: Send unsolicited text messages or intercept and drop incoming messages.

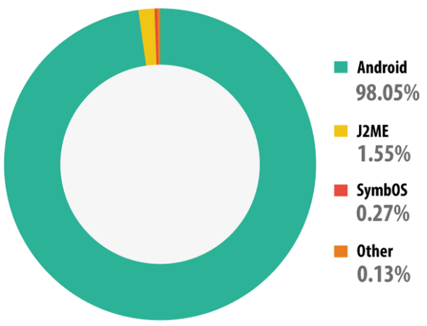


Trojanized Applications

- Trojanized application is the greatest vector of malicious software.
- A Trojanized Application is an application for mobile devices (app) that hides malicious code.
- Trojanized apps generally looks like apps that works correctly, providing a service to the user.
- Malicious code runs in background.
- They are quite hard to be detected by average users.

Target of the attack

- Given that Android is the most common operative system, it is also the main target for mobile malware.



Android

- The Android Open Source Project
  - Philosophy

open handset alliance

Android (2)

- Part I: Android System and Applications
  - Android Framework

  - Android Applications
  - Google Play

Android (3)

- Part II: Android Security
  - Native Mechanisms
  - Attacks
  - Innovative Solutions

Part I

- Android System and Applications

The Android Open Source Project (AOSP)

- What is Android?
  - A mobile OS? **No**
  - A framework for mobile devices **Maybe**
- Android is an **Open Source** Project held by the Open Handset Alliance.

Open Handset Alliance

- Consortium of Enterprises that work in the field of mobile communications.
  - Service Provider
  - Hardware Manifacturer
  - Smartphone Producer
  - Software developer
  - …

GARMIN
FIH
DELL
ASUS
ALCATEL mobile phones
acer
MOTOROLA
LG
lenovo 联想移动
KYOCERA
HUAWEI
htc smart mobility
ZTE中兴
TOSHIBA
Sony Ericsson
SHARP
SAMSUNG
NEC
NUANCE
myriad
Living Image
Google
ebY
Ascender Corporation
VisualOn
SVOX
SONiVOX sound that rocks
skypop.
pv
omron
tat
embinux
sasken
noser
BORQS
Aplix Corporation
中国电信 CHINA TELECOM
中国移动通信 CHINA MOBILE
Bouygues Telecom
WIPRO
WIND RIVER
TELECA
T-Mobile
Sprint
SoftBank
docomo
KDDI Designing The Future
China unicom中国联通
ARM
Audience
AKM
vodafone
Telefonica
TELECOM ITALIA
gemalto
freescale
CYPRESS
csr
BROADCOM
ATHEROS
RENESAS Renesas Electronics
QUALCOMM
NVIDIA
MIPS TECHNOLOGIES
MARVELL
intel
Ascender Corporation
VIA Telecom
TEXAS INSTRUMENTS
Synaptics
ST ERICSSON
NXP SOFTWARE
L&T Infotech
MOTOYA
Audience
MEDIATEK
CCI

Open Handset Alliance

- The Open Handset Alliance is led by one company:

Google

Android Philosophy

- Open Source:
  - All of Android source code is available and can be downloaded and modified.
  - Improvement can be uploaded as system patches.
  - Patches pass through a long review process.

Android Devices

- Android has been designed for three type of devices:
  - Smartphones
  - Tablets
  - Embedded Systems.

Versions and Distributions

- **Apple Pie** (Android 1.0) Developed for HTC Dream.
- **Cupcake** (Android 1.5) Several Graphic improvements.
- **Donut** and **Eclair**(Android 1.6 – 2.1).
- **Froyo**(Android 2.2) First version with a large distribution.
- **Gingerbread** (Android 2.3 – 2.6) Installed on several smartphone in particular: Samsung Galaxy, Galaxy S and Galaxy S2.
- **Honeycomb** (Android 3.0) Distribution for tablets only.
- **Ice Cream Sandwich** (Android 4.0) For tablets and smartphone. Large distribution, used on Samsung Galaxy Nexus.
- **Jelly Bean** (Android 4.1 – 4.3) Released in 2012, includes new graphical elements for applications.
- **Kit Kat** (Android 4.4) Latest Release for Smartphones and Tablets. Includes the Google Now service.

Android Full Code

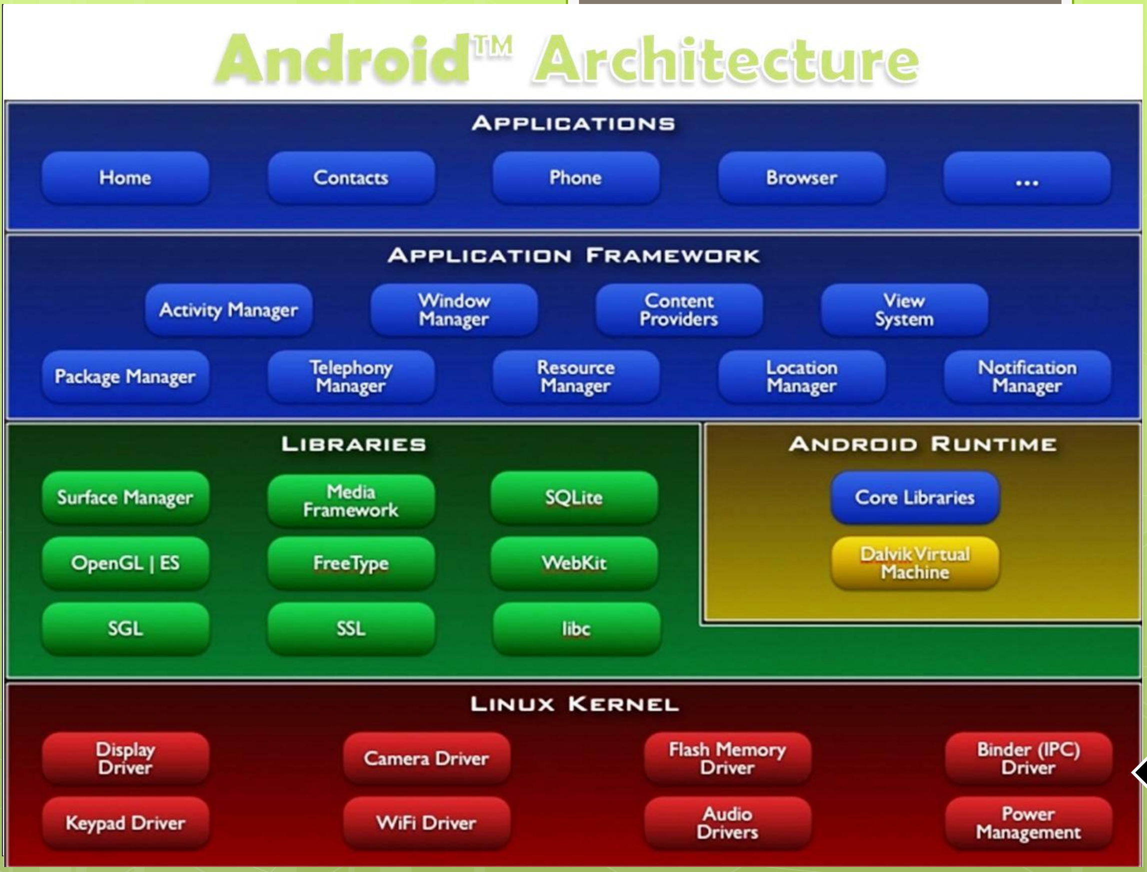
- The full code of Android is available at www.source.android.com as a **git** repository.
- Requires more than 10 GBs of mass storage and a swap of 20 GBs to be compiled.
- *Note: The source download is approximately 8.5GB in size. You will need over 30GB free to complete a single build, and up to 100GB (or more) for a full set of builds.*
- There is a version for smartphone (Maguro), one for Emulator (Goldfish) and one for embedded devices (Panda).

ROMs (1)

- The ROM is an Android Image (few MBs) of the OS that is installed on a device.
- Manufacturers ROM
  - Smartphone manufacturers equip their devices with custom ROMs.
  - Inclusion of manufacturer software (Samsung Kies…)
  - Some limitations on functionalities (Tethering).

ROMs (2)

- Custom ROM
  - ROMs modified by third party developers.
  - Inclusion of additional features.
  - No limitations on functionalities.
- Original ROMs
  - Installed on **Nexus** devices, which are devices produced by manufacturer under the guidance of Google.

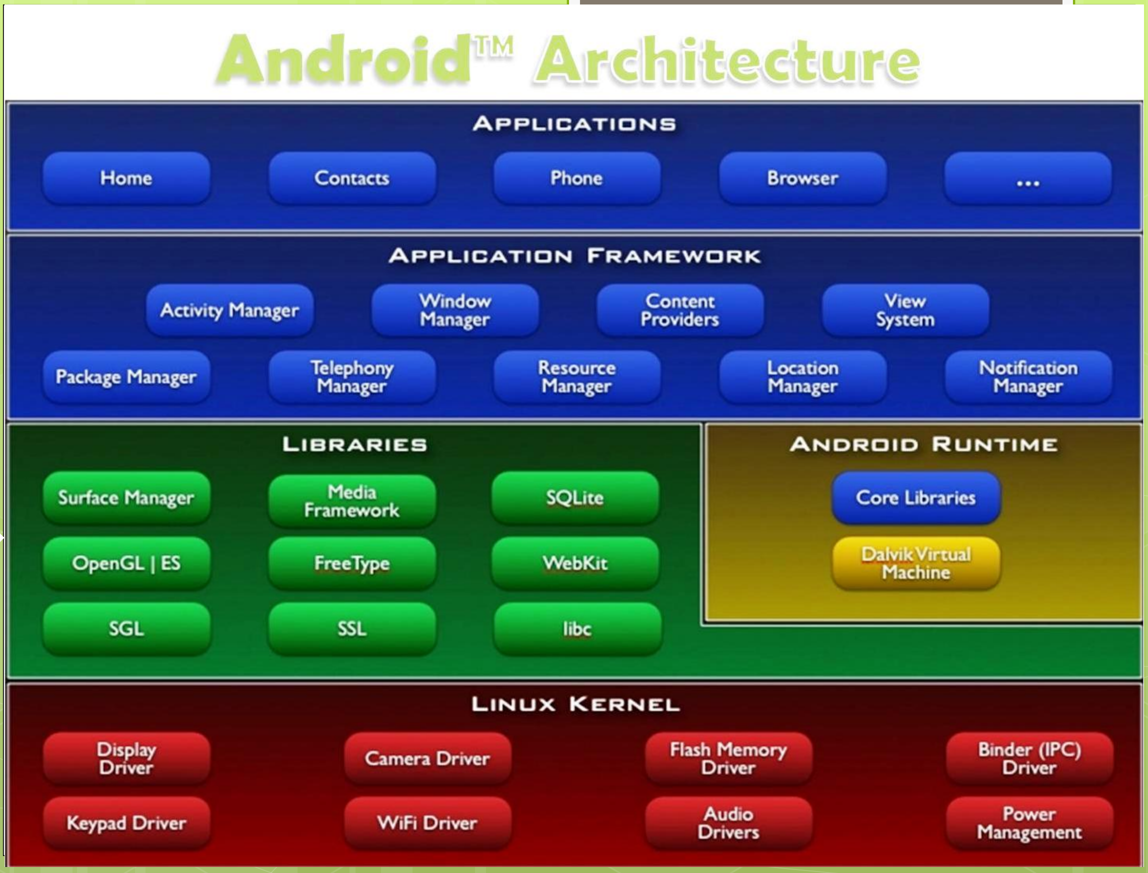


Kernel(1)

- The Android framework runs on top of a Linux Kernel.
  - Shell available.
  - Some commands are not available.
  - Some modules are not compiled.
  - In particular it is not possible:
    - To copy a file.
    - To create or modify users.
    - Become Super-User.

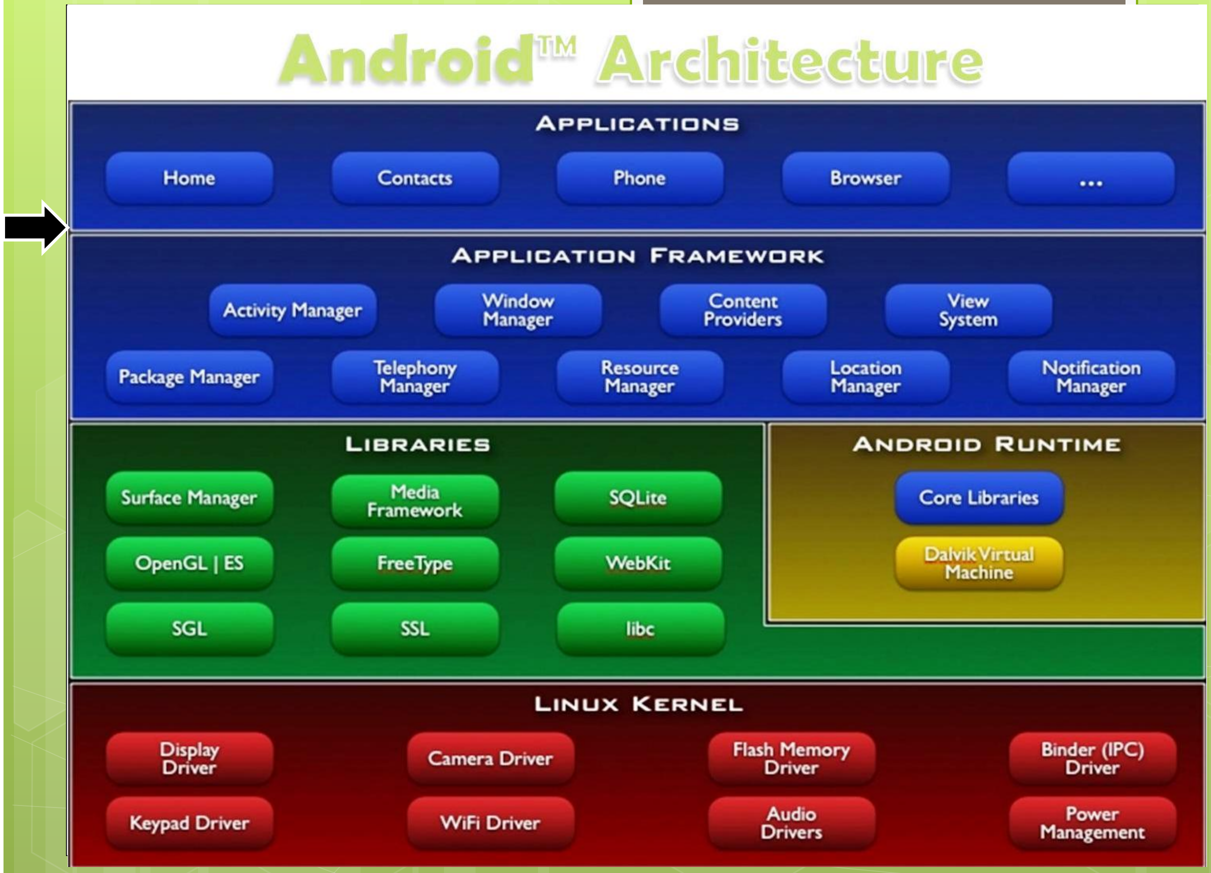
Kernel(2)

- Kernel Tasks:
  - Handles Inter Process Communication (IPC).
    - Processes cannot communicate directly.
  - Handles Inter Component Communication (ICC).
    - Hardware and Connection Interfaces.
  - Executes all of the low-level tasks.
  - Enforces Security.



Libraries

- Libraries written in C/C++.
  - They work as support for high performance and real time tasks (OpenGL).
  - Security (SSL).
  - Communication (Socket).
  - Database interaction (SQLite)
  - …



Application Level

- The application level of Android is entirely based on Java.
  - Android uses a slightly modified version of Java.
    - Clash between Google and Oracle.
- Android applications are programmed in Java.

Why Java?

- Open Source Language.
- Highly portable.
- Object-Oriented and extremely expressive.
- Use of Virtual Machine.
  - The Java Virtual Machine is an environment in which Java applications run.
  - Ensures portability and security.

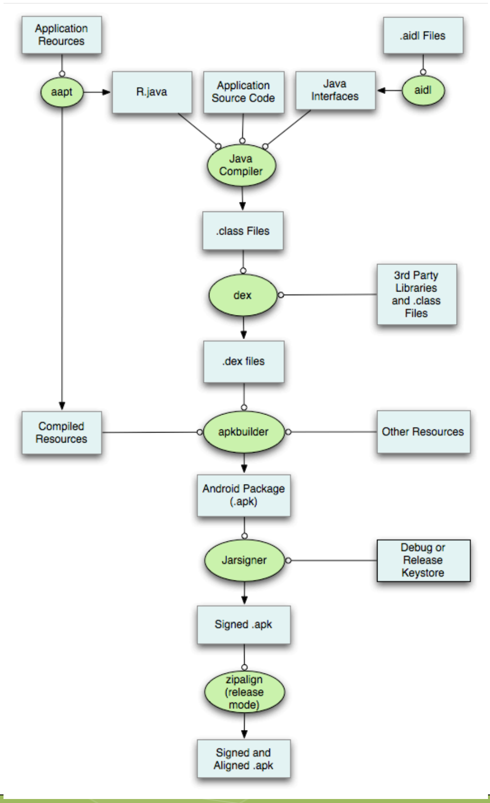
Android Applications (App)

- Android applications come as a unique file directly installed on the device.
- Application PacKage (**APK**) are a bundle of file that contains both executables and static resources.
- Android applications are developed in Java.
- They are distributed through marketplace.

Installing

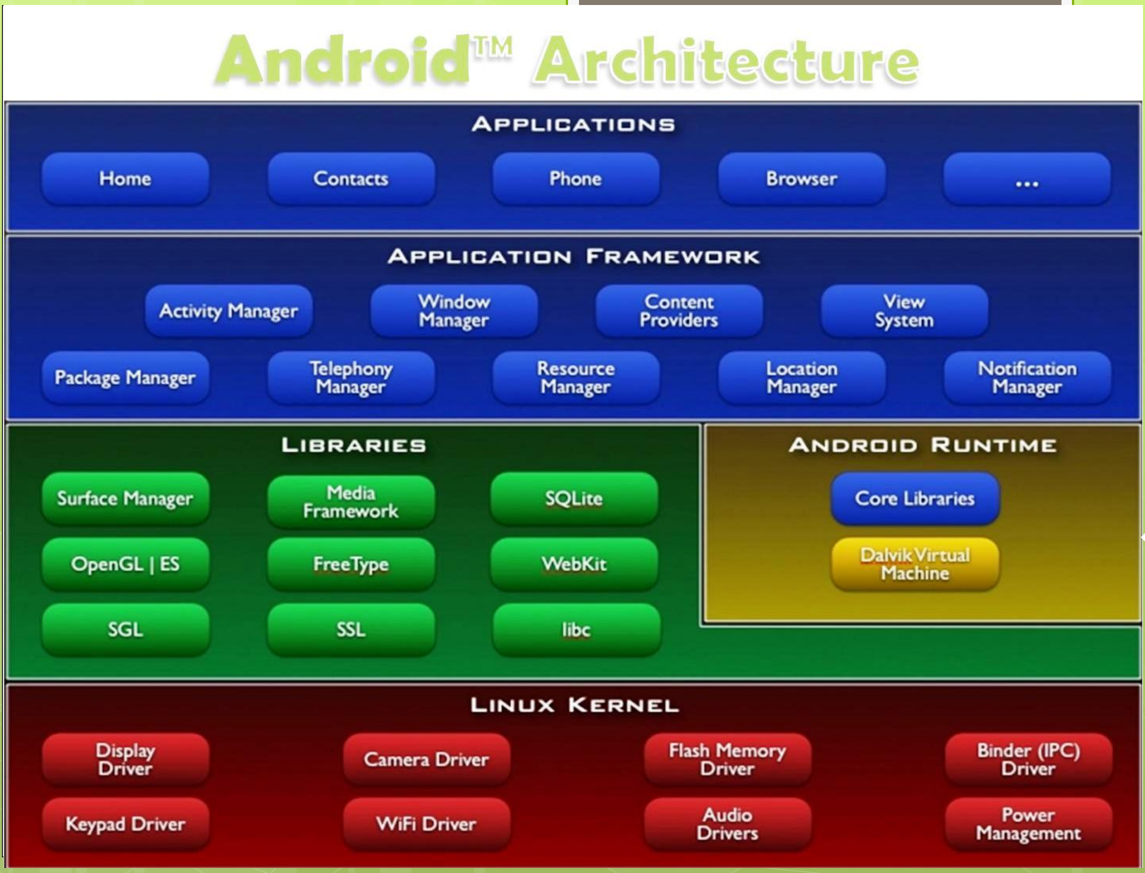
- How to install an app?
- There are three methods:
  1. Market Installer: Use an application like Play to browse choose and Install Apps
  2. File Browser: Put the app on the device memory and install it with a file browser .
  3. Use the Android Debug Bridge (ADB).

Building (1)



Building (2)

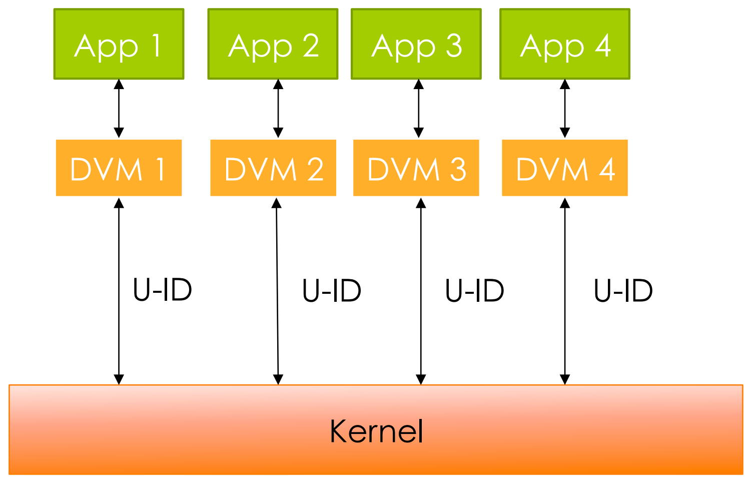
1. Interfaces, resources and source code are compiled by a classical java compiler.
2. Class files are **dexed**. The result is a **dex** (Dalvik EXecutable) file, an optimized version of bytecode.
3. Executable are merged with static resources to create an apk file.
4. The apk is signed to ensure integrity.
5. Further optimization through zip align.



Android Runtime

- In Android applications run on a modified version of the JVM.
- Dalvik Virtual Machine (DVM) is faster and lighter than the classical JVM.
  - Suitable for mobile devices.

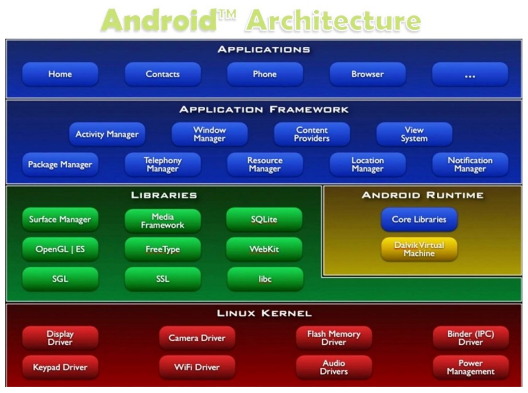
App Execution (1)



App Execution (2)

- Android applications run in the Dalvik Virtual machine. For each running application a different DVM is instantiated.
- The DVM interacts with the underlying Linux kernel.
- Every DVM has a Linux UID. Thus every Android application is considered a different Linux user.
  - The Linux UID is assigned to an application at install time and is not changed until the app is not uninstalled.

Device Side Components



Developer Side Components

Standard Development Kit (SDK)

- The Android SDK is a bundle of all the software and tools necessaries to develop, debug and test Android applications.
  - **APK Builder:** Creates ready-to-install applications from code
  - **Android Debug Bridge:** Allow the USB connection and management of an Android device.
  - **Emulator:** Android device emulator.
  - **Android Developer Tool (ADT):** Plugin for the Eclipse IDE.
  - **Fastboot:** Boot a connected device in different modes.
  - **Mksdcard:** Used to create a virtual SDCard**.**

Get it!!

http://developer.android.com/sdk/index.html

Checklist

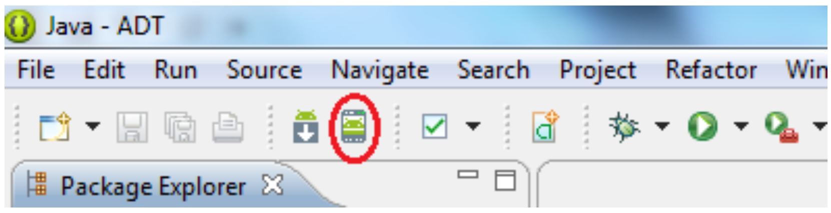
- Java Runtime Environment installed?
- Java Environment variables correctly set?
- PATH Environment variables correctly set?

ADT

- Plugin for Eclipse to develop Android applications.
- Includes DDMS to interact with other tools of the Android SDK.

Emulator

- An Android device emulator with (almost) all the functionalities of a real smartphone..
- Virtual devices are created through the Virtual Device Manager.



Android Debug Bridge (1)

- Used to connect and interact with an Android device.
- Some options:
  - adb shell: open a linux shell on the device.
  - adb push/pull: push or pull a file onto/from device.
  - adb install: installs an application on the device.

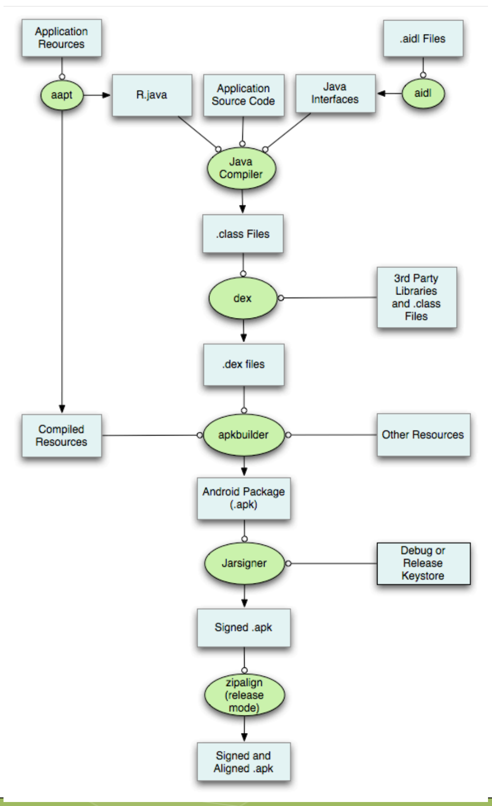
Android Debug Bridge (2)

- Other ADB commands:
  - adb reboot: reboots the connected device.
  - adb devices: lists the connected devices.
  - adb logcat: show the device logcat, used for debugging.
  - …

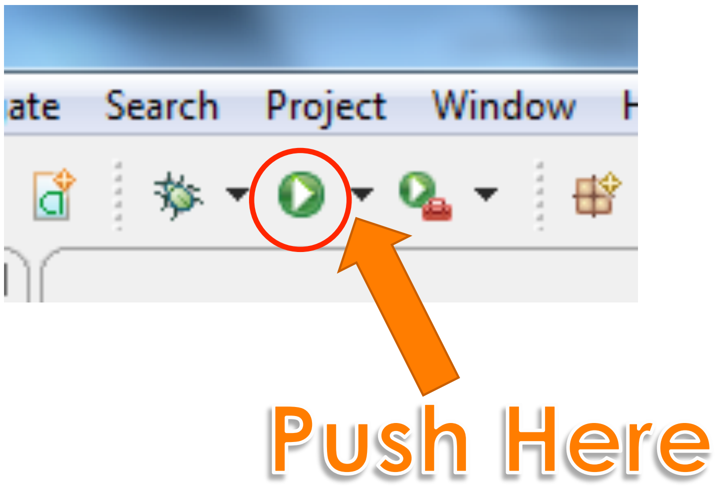
App Developer

- Since the Android SDK can be downloaded for free, virtually anyone can be an app developer.
- We can roughly divide developers in 3 categories:
  - Enthusiast Developers.
  - Professional Developers.
  - Google

Building



Building (with Eclipse)



Programming

- Hello World!
- Create a new project in Eclipse and call it: HelloWorld.
- When ready, build the project.

Application Project (1)

- Folders:
  - **src**: contains the source code written by the developer.
  - **gen**: auto-generated files. These files should not be manually modified.
  - **assets**: all non-pictures resources used by an application should be put here.
  - **bin**: automatically generated executable and files.
  - **libs**: external libraries.
  - **res**: icons, pictures and xml files to describe layouts and fixed values.

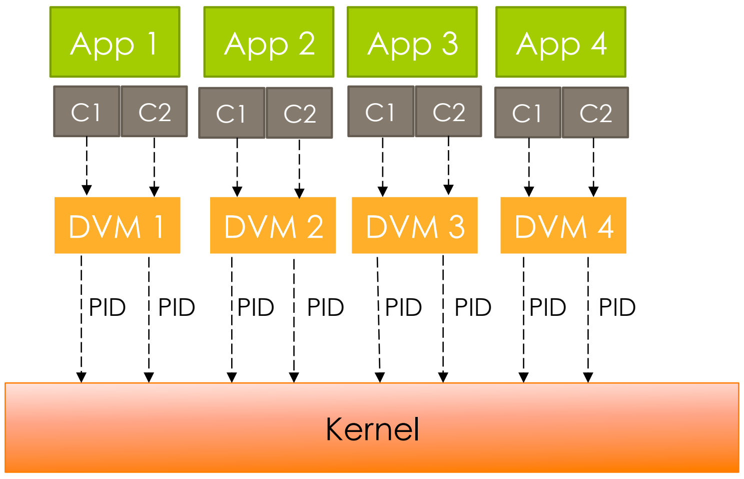
Application Project (2)

- Android Manifest
  - XML file that describes an application.
  - Contains info on developer and version.
  - Lists all the **application components**.
  - Lists all the resource accessed by the application (permissions).
  - Lists all functionalities offered to other applications (intent filter).

Application Components

- Android applications have 5 main components:
  - Activity
  - Service
  - Intent
  - Content Provider
  - Broadcast Receiver

App Execution (3)

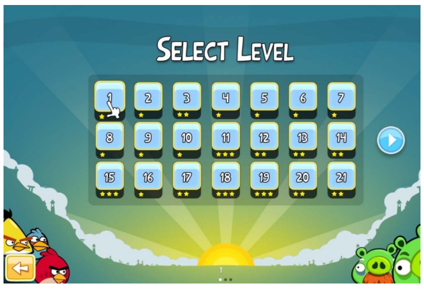


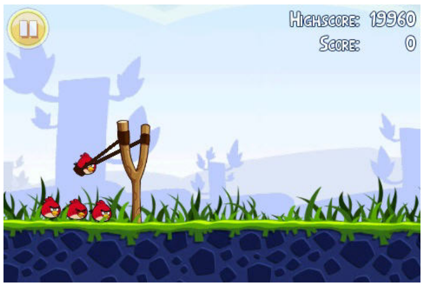
App Execution (4)

- An Android application may launch, through the DVM, different processes.
- Generally an application with several components launches a process for each running component. The user of this processes is the one assigned to the application.

Activity

- An Activity is a single screen with a user interface of an application.
- Generally an application is composed by several activities. Each activity represent a single application screen.





Creating an activity

- First Step?
- The activity is an application component, so it has to be declared in the **Manifest** file.
- Look at the code of the sample application…

Creating an Activity (2)

- Graphic Layout.
- Layout of activities are described statically through XML files stored in **res** folder.
- Text box, buttons, loading bars and so on can be added and customized.
- GUI. Simple with drag and drop functionality.
  - Hint: Insert the items in the activity screen using the GUI, then customize them statically from XML code.

Creating an Activity (3)

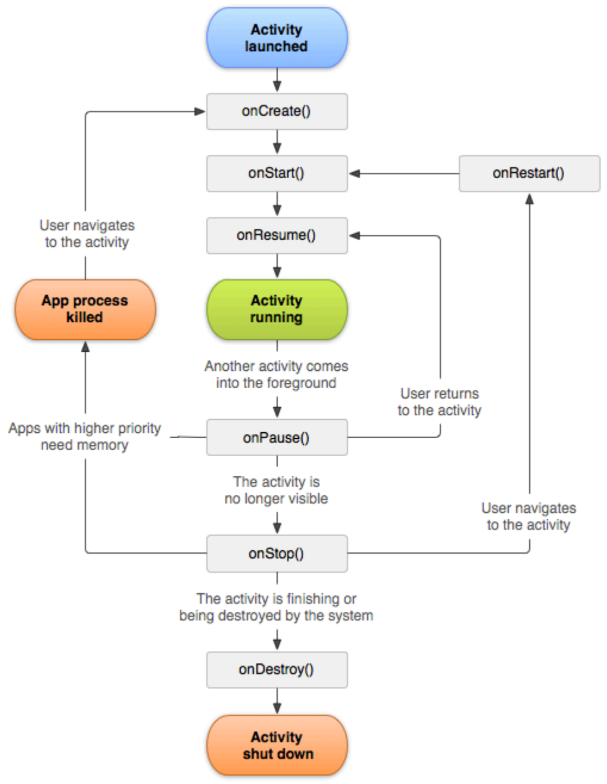
- Java class.
- Each activity should be a java class in your project.

  public class myActivity ***extends*** *Activity {*

  …

  }
- There is no main in an activity class. Everything starts from the method **OnCreate().**

Activity Lifecycle



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity Status

- Running Foreground.
  - The activity is visible and in the foreground.
- Running Background (Paused).
  - The activity is visible but another app has the focus.
- Stopped.
  - The activity is not visible.
- Destroyed.
  - The activity has been killed, all variables have been destroyed.

Zygote

- User does not decide when to close an application.
- The user starts an Android application (and thus one or more activities), then the application runs until the process manager **Zygote** does not assert that it is not useful anymore.
- What about variables stored in memory?

Variables

- If Zygote destroy an application, all variables will be deleted!

Bundle

- The Bundle is a data container. Is a structure where it is possible to push values of variables.
- Bundles are stored in memory and survive when an application is destroyed.
- Save data to bundle in *OnSaveInstanceState()* method, restore data from bundle in *OnCreate()*.

Warning!!!

- Do **not** save data in *onStop()* or *onDestroy()* methods. These methods could be not invoked in cases of extremely low memory.
- Use them only for clean-ups.

Starting an Activity (1)

- When an application is started the main activity is launched.
- The main activity is declared in the manifest file through an **intent filter** for the application launcher.
- If the main activity is not declared the application will not start.

Starting an Activity (2)

```
public void callActivity(){
Intent intent = new
Intent(this,CalledActivity.class);
startActivity(intent);
}
```

Intent

- **Intents** are used to send messages and data between applications or application components.
- Every Intent has a *sender* and a *receiver*.
- There are two types of Intents:
  - Explicit Intents: the sender specifies the intent receiver.
  - Implicit Intents: the sender specifies a class of possible receivers.

Intent Components

- **Action**: The action that should be performed by the receiver. Used to assess who should be the receiver. Example: ACTION_CALL, ACTION_MAIN
- **Data**: information sent from the sender to the receiver.
- **Category**: Additional info on the type of operation requestested.

Explicit Intent

- **Intent(this,CalledActivity.class);**

- The receiver is explicitly specified. The Intent has to be delivered to the CalledActivity class.
- The action can be considered implicit, since it is specified by the method: **startActivity(intent);**

Implicit Intents

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
startActivity(i);
```

- For implicit intents the receiver is not specified. With this intent the sender asks that the web page at www.google.com is displayed.
- The OS looks for all the applications or components that are able to satisfy such a request.
- Applications specifies that they are able to satisfy a request declaring an Intent Filter.

Intent Filters

- Intent Filters are declared in manifest file.

```
<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>
```

Exercise

- An easy **Money converter** with two activities.
- The first activity allows conversion.
- The second activity allows the definition of the conversion rate.

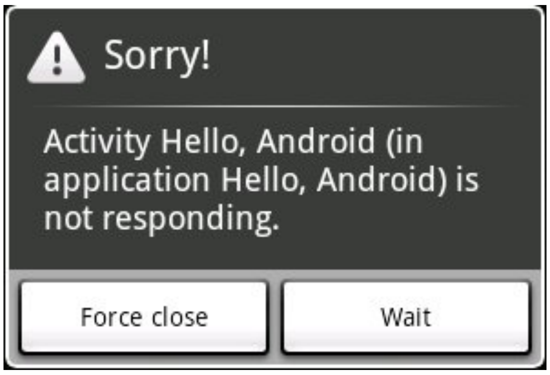
Service

- A service is an application component that perform long and heavy background operations.
- Services do not provide a user interface.
- A service is called by an activity or another service and generally run for a long period of time.

Example of Services

- GPS location service.
- Timers.
- Watchdogs
- Streaming managers.
- Loggers.
- …

Why do we need Services?


Sorry!

Activity Hello, Android (in application Hello, Android) is not responding.

Long Operations in Activities

- Each time an **activity** takes more than 30 seconds to perform an operation an alert is raised.
- This is considered a programming error.

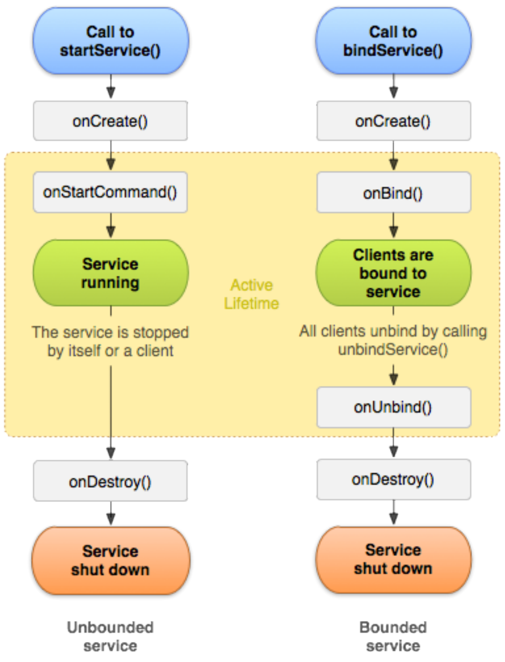
Long Operations in Activities

- When an Activity has to perform a long operation should start a service and go on with the execution.
- The service executes its task in background and eventually return results to the calling activity.
- An activity start a service similarly to other activities: using Intents.

Starting a Service

- There are two methods to start a service from an activity
  - *startService(Intent)*: Non-bounded call. The new service starts and live indipendently from the calling application.
  - BindService(Intent): Bounded call. The service is linked to the calling application and there is an interface of communication between the service and the application. The service dies when the application is called.

Service Lifecycle



Terminating a Process

- Differently from activities, services can be terminated programmatically using the *stopService()* or *stopSelf()* methods.
- If not terminated explicitely, the service will be stopped and destroyed by Zygote when the system needs memory.
- A service can be closed when it is still necessary.

Sticky Start

- The method *startService(intent, flags)* can be used to specify what happens when a service is destroyed by Zygote.

1. *START_NOT_STICKY:* After destruction the service is not re-created.
2. *START_STICKY:* The service starts again after destruction as soon as the system has enough memory.

Content Providers

- Content providers are data structures that allow to save and access data as in a relational database.
- Data are stored in tables.
- Tables are accessed by SQL like queries.