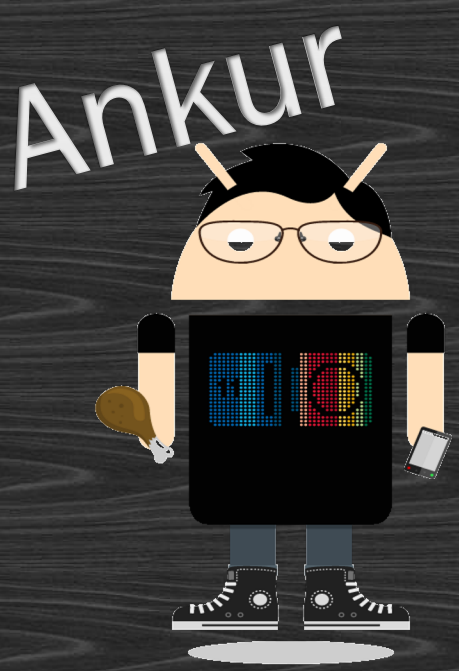
ANDROID HACKING VILLAGE

ANKUR | MRIGESH | ANANT

WHOAMI



TOPICS

Day 1 : Basics

- Android Architecture
  - Operating System Overview
  - File system Overview
  - Security Model
- Developer Overview
  - Application Components
  - Application Structure
  - The SDK and Android Tools
  - Developing a basic application
- Intro to PenTesting
  - Setting up the environment

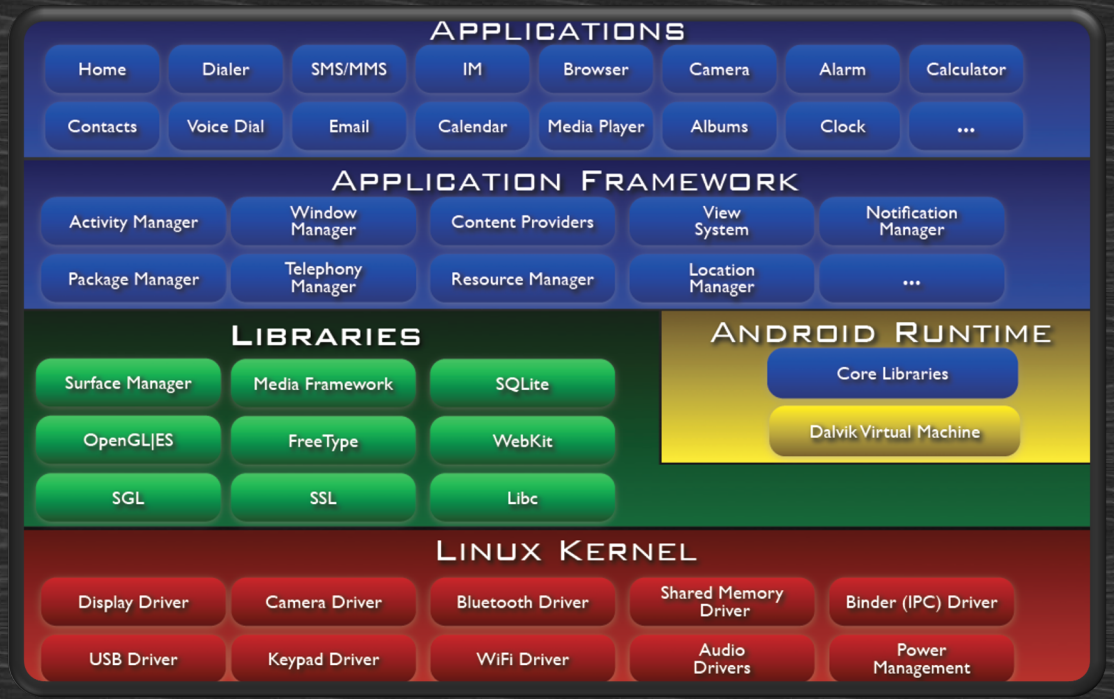
Day 2 : Advanced

- Malware Analysis and Design
  - Exploits survey
  - Common malware samples
  - Detection, prevention and cure
- ROM cooking
  - Rooting basics
  - Simple Mods
  - Mid-range Mods
  - Hardcode Cooking
- Advanced Pentesting and Forensics
  - Black Box PT
  - Reverse Engineering
  - Memory Analysis

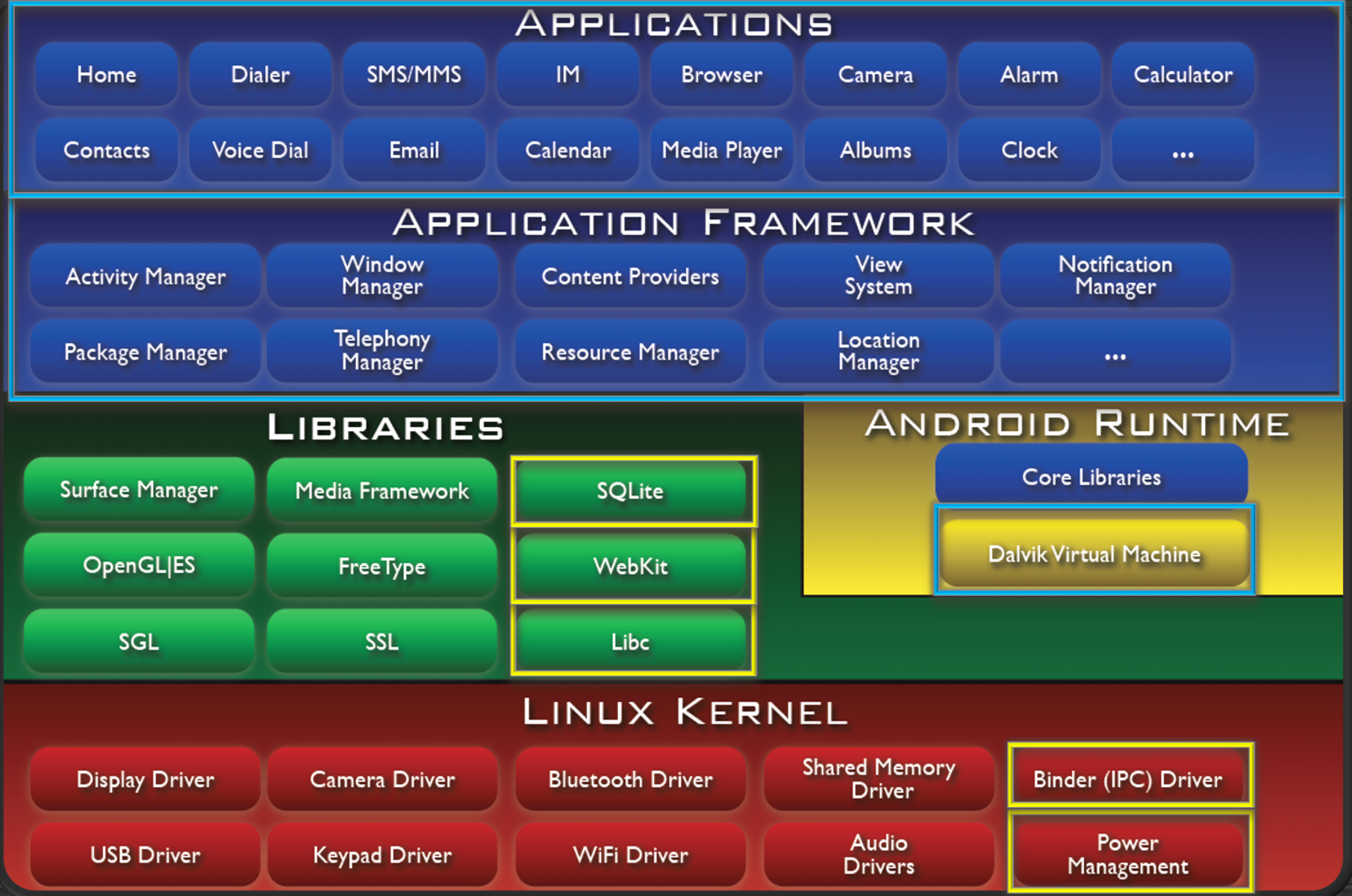
Android Arrives

- 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.

Android – An Introduction



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


Applications
Home
Dialer
SMS/MMS
IM
Browser
Camera
Alarm
Calculator
Contacts
Voice Dial
Email
Calendar
Media Player
Albums
Clock
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
...
Libraries
Surface Manager
Media Framework
SQLite
OpenGL|ES
FreeType
WebKit
SGL
SSL
Libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Bluetooth Driver
Shared Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

ANDROID ARCHITECTURE (FILESYSTEM OVERVIEW)

- Android Devices generally contain the following mounts

/	ro	The root. Cannot be accessed in normal scenarios.
/system	ro	Contains the core binaries and enforced APKs
/data	rw	Contains installed packages
/proc	rw	
/mnt/sdcard	rw	SD Card mount location
/mnt/sdcard/asec	rw	Secure storage for APKs on SD Card

ANDROID ARCHITECTURE (FILESYSTEM OVERVIEW)

- Some important folders on a typical Android device:

/system/app	ro	
/system/framework	ro	
/data/app	rw	
/data/data	rw	
/sdcard/asec	rw	

Android - Security Model – The Good

- **Traditional Access Control**
  Idle-time/ one-click phone locking

- **Isolation (Sandboxing)**
  Every application runs within its own VM, and each VM is in a unique Linux process.
  No process can access the resources of any other process.

- **Permissions based access control (Application-defined and user-granted)**
  Developer configured whitelist for accessing Android and 3rd party resources.
  Can also enforce permissions preventing unauthorized components from executing app.

- **Application Provenance**
  Applications signed using a X509 digital certificate.
  $25 charge for uploading app onto Android market.

- **The Kill Switch**
  Allows remote kill, uninstall and data cleanup

Android - Security Model – The Bad

- No hardware-based encryption
- No non-executable memory area
- Limited Developer Accountability
- Poor Code Obfuscation options
- Applications can easily be ‘trojanized’
- Difficult environment for anti-virii
- Long Patch cycle
- Recovery/Boot process
- Security enforcement reliant on end-users

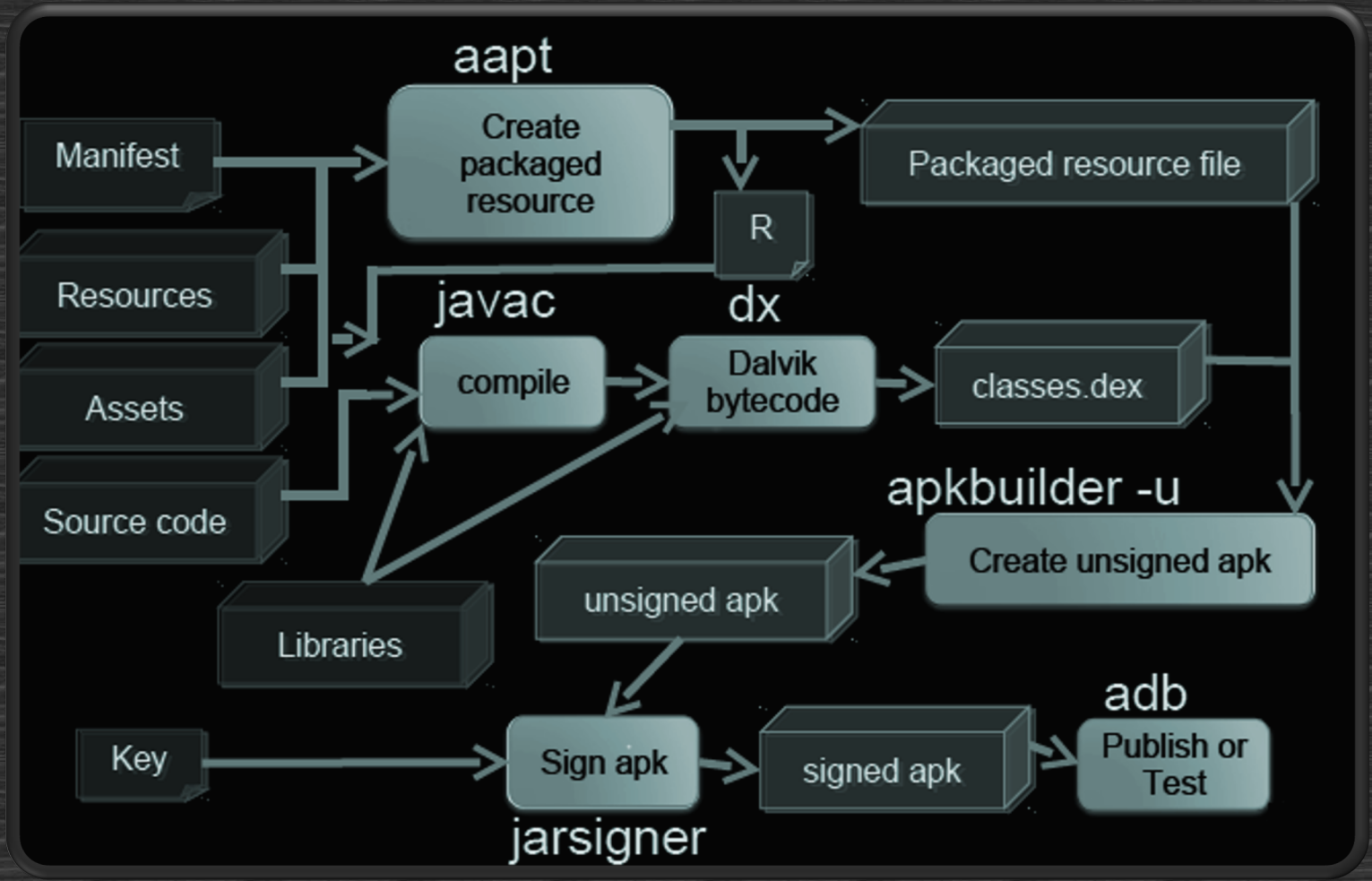
Android - Security Model – Updates

- 2.2 (Froyo)
  - Built-in remote wipe capability.
  - Built-in 4-digit-pin and alphanumeric password options for enterprise security.
  - Enterprise security additions like remote-wipe, device admin and password protect.
- 2.3 (Gingerbread)
  - In certain architectures (ARMv6), Android supports non-executable pages by default including non-executable stack and heap using an eXecute Never bit.
- 3.0 (Honeycomb)
  - Android after 3.0 provides full filesystem encryption
- 4.0 (Ice Cream Sandwich)
  - Introduced face-unlock
  - Introduced WiFi Direct as well as data sharing via NFC using Android Beam.

Android Applications - An Introduction

- Android installables are known as Application File Packages (APKs)
- ZIP-formatted packages based on the jar file format
- An APK is a collection of components. The components share certain resources.
- These include a Linux process, a JVM, SQLite DBs, Shared Preferences, a File Space…
- An APK also contains a store of other non-code resources

Android Applications - Structure



Android Applications – The SDK

- Includes a vast array of development tools.

- The officially supported IDE is Eclipse using the Android Development Tools Plugin.
  *You may also use any text editor along with various command-line tools to build and debug apps as well as connect to an emulator/device.

- ADT includes a suite of tools that can be easily integrated with Eclipse's IDE. Some important ones are.:
  - adb
  - android
  - DDMS
  - emulator
  - logcat
  - ProGuard

- ADT controls synchronizes all the above tools with as many devices (real/virtual) as you can connect to.

Android Applications – Building a sample app

[THE COMPONENTS]

- Activities
  - UI component for one focused task.
  - Usually a single screen in your app.
  - Stack-based lifecycle
- Services
  - Long-lived worker code. No UI.
  - Can be connected to using Binder.
- Intents
  - Mechanism for IPC.
  - Defined using an Action/data pair
- Intent Filters
  - Describes what intents a comonent can handle
  - Registers activities/services/receivers.
- Broadcast Receivers
  - Responds to broadcast intents
  - Must be declared in manifest/code
- Content Providers
  - Manages Persistent Data
  - Publishes it to other apps
- Shared Preferences
  - Persistent data stored as name-value pairs
- Uses-Permissions
  - Describes permissions used by your app
- Permissions
  - Restricts access to your components

Android Applications – Building a sample app

[GUI BUILDING]

- Layouts are defined in XML (similar to Swing)
- In Android, we implement the Activity interface in Java.
- We need to overwrite a single function, i.e. onCreate()
- XML layouts are later expanded into Java to generate a UI.
- This is done using setContentView(Reference.to.the.XML.file)
- Any controls defined in your XML layouts can be referred to in Java by their ids using the findViewById() function.
- For each view, a set of event listeners and responders can be defined.

Android Applications – Building a sample app

[COMMON ERRORS]

- Build path error.
- AAPT unable to parse
- Invalid Imports
- Component unimplemented in manifest
- Attempting to access without taking the required permissions
- Expired debug certificate

Mobile Application Penetration Testing

Web Application Testing	Mobile Testing
Application/Business Logic not present locally.	App stored in the device.
No database present locally. Except cookies,cache	App do store some data in device to increase the performance.
All apps run in browsers	Need to install emulators for each platform
Reversing not Applicable	Applicable as installable is inside the device itself.
Read more	

Android – Setting Up a PT Lab

- Setting up the environment
  - Using proxy tools
  - Using Echomirage
  - Using Autoproxy
- Android Debugging tools
  - adb
  - Ddms
- Reversing Android Apps
  - apktool, baksmali, dex2jar

TOPICS

Day 1 : Basics

- Android Architecture
  - Operating System Overview
  - File system Overview
  - Security Model
- Developer Overview
  - Application Components
  - Application Structure
  - The SDK and Android Tools
  - Developing a basic application
- Intro to PenTesting
  - Setting up the environment

Day 2 : Advanced

- Malware Analysis and Design
  - Exploits survey
  - Common malware samples
  - Detection, prevention and cure
- ROM cooking
  - Rooting basics
  - Simple Mods
  - Mid-range Mods
  - Hardcode Cooking
- Advanced Pentesting and Forensics
  - Black Box PT
  - Reverse Engineering
  - Memory Analysis

Android Threat Model

THREATS

- Remotely Infect via Market/Browser
- Privilege escalation attacks
- User tracking
- Data stealing
- Resource misuse

VULNERABILITIES

- Insecure storage
- Insecure IPCs
- Insecure component starting
- Insecure WebKit

MALWARE FOR THE FUTURE

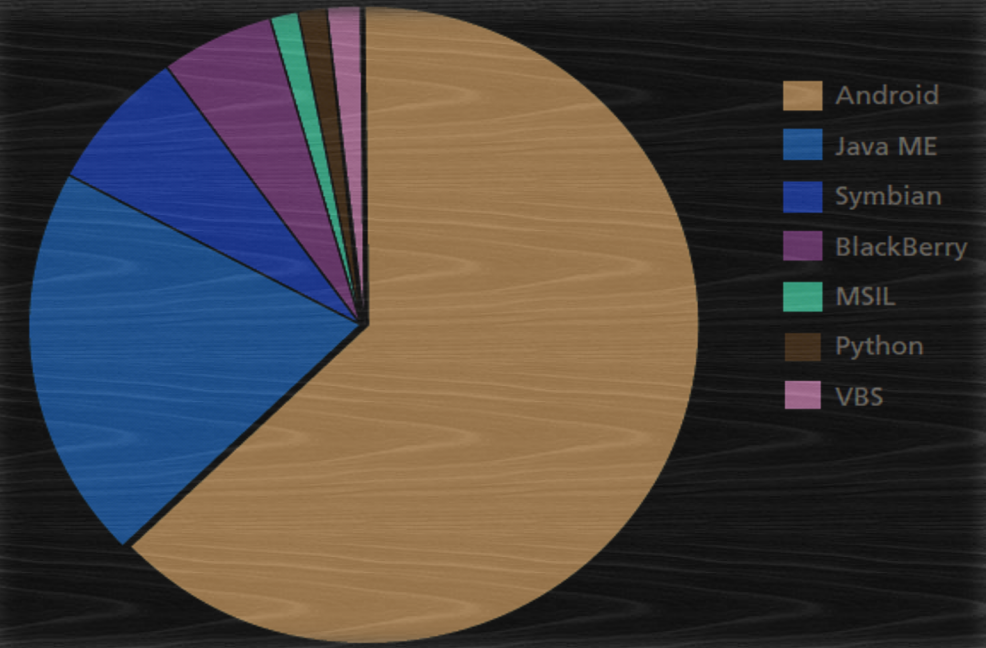
Botnet Capabilities

Application Harvesting
Key Loggers

Android Malware facts

•Reports say that chances of an android phone being compromised is 2.5 times more than any other platform.

•Count of new Android-specific malware moved to number one, with J2ME (Java Micro Edition), coming in second while suffering only a third as many malware.



Android Malware

•Increase in for-profit mobile malware, including simple SMS-sending Trojans and complex Trojans that use exploits to compromise smartphones

•Mobile threats already take advantage of exploits, employ botnet functionality, and even use rootkit features for stealth and permanence.

•Maliciously modified apps are still a popular vector for infecting devices: Modify a legitimate app or game and users will download and install malware on their smartphones by themselves.

Android- Popular Malware Apps

Falling	DownSuper	Guitar
SoloSuper	History	EraserPhoto
EditorSuper	RingtoneMaker	Chess
SupersexPositions	HotSexyVideos	Falldown
FallingBallDodge	Scientific Calculator	Dice Roller
APPUninstaller	Funny Paint	Spider Man
AdvancedCurrencyConverter		

Android- Popular Malware

- The Android/DrdDream infected over 50+ apps in google market. Included root exploits and installed one more application in system/app. It uses DES to encrypt the data it sends to the attacker.

- The Android/DroidKungFu family is similar to Android/DrdDream; it also uses a pair of root exploits to maintain itself on a device.The exploits are actually identical to those used by the Android/DrdDream except they have been encrypted with AES. These variants can also load URLs and install additional software and updates.

Android- Exploits

- KillingInTheNameOf
  - Affected Android <= 2.2
  - Remapped Android property space to writable
  - Vulnerability in Ashmem implementation
  - Applications having permissions to change shared properties.
  - Toggled ro.secure property to 0
  - ADB Daemon now runs as root
  - Physical local root through ADB shell

Android- Exploits

- RageAgainstTheCage
  - Affected Android <= 2.2
    - ADBd initially runs as root and use setuid() to move to uid shell
    - If NPROC resource limit is reached for uid shell,
    - setuid() from uid root to uid shell will fail
    - If setuid() fails, ADBd continues running as root
    - If ADBd runs as root, ADB shell also runs as root
  - Fork()'s processes as shell user until NPROC is reached
    - Restart ADBd (bringing uid shell to NPROC-1) and fork()again (as uid shell) right before ADBd (as uid root) tries to setuid() back to uid shell, setuid() fails, Rage wins

ROM AND ROOT

ANKUR | MRIGESH | ANANT

AGENDA

- Custom v/s stock
- Types of Modding
  - Simple Mod : GUI element modding + automated
  - Mid level mod : decompiling, reversing.
  - Hard Core : cross compiling, drivers modding
- Rooting Fundamental’s
- How Exploits Work

CUSTOM ROM V/S STOCK ROM

- Stock Rom are build for generic usage and large masses, they are non rooted by default.
- Custom ROMs
  - can have variants like
    - Gaming ROM
    - Battery Saver
    - Overclocking, Undervolting etc.
  - Bleeding edge
  - Pre-Rooted

SIMPLE MODDING

- Removing unnecessary /system/app
  - + More free space.
  - + Faster system (low ram consumption)
- Adding general usage applications as system app.
  - + Disadvantage : your updates will take dual space.
- Removing/replacing av files, animation etc
- Changing the GUI elements.

Simple Modding

DEMO TIME

MID LEVEL MODDING

- This is where we start getting serious ☺
- Application decompiling and reversing to modify default behavior of application.
- Application will include CORE Applications also.
- Focus area's
  - /system/app
  - /system/framework

Mid Level Mod

DEMO TIME

HARD CORE MOD

- This is where we start getting serious ☺
- Application decompiling and reversing to modify default behavior of application.
- Application will include CORE Applications also.
- Focus area's
  - /system/app
  - /system/framework

Simple Modding

DEMO TIME

HOW ROOTING WORKS

- Simplest Allow user to have root access
  - + Temporary and
  - + Permanent
- Temporary root, Exploit an application gain root access
- Permanent make abover condition permenent by installing a backdoorr
- SU and SUPERUSER.apk
- Review su source code

Questions?

CONTACT US

- Anant Srivastava
  Email – anant@anantshri.info
  Blog – http://anantshri.info

- Ankur Bhargava
  Email – ankurbhargava87@gmail.com
  Blog – http://www.hakers.info

- Mrigesh
  Email – me@mrigesh.com

Appendix : Androidisms & the Component Model

- Binder → Component Framework
- Activity →UI
- Service → Background Processing
- Content Providers → Data Sharing
- Broadcast Receivers → Listener
- Intents → IPC/ICC
- Manifest →Metadata
- Permissions → Ensures authorization
- API Levels → Separate Android Versions
- Ashmem → Anonymous Shared Memory
- Pmem → Physical Memory Allocation
- ROM → Refers to /system folder
- Stock ROM → Pre-installed Android OS
- Custom ROM→ Unauthorized Modded OS
- Rooting → a
- ADB→ a