Agenda

1. Introduction to Android
2. Android’s Security Model
3. Android Threat Landscape
4. Behind the scenes: Android App-Infection
5. Demo: Android.Geinimi
6. Q&A
Introduction to Android
What is it?, Who’s made it?

- Linux based Operating System for mobile devices
- Developed by the Open Handset Alliance (OHA)
  - Consortium of 84 firms that develop standards for mobile devices, among its members are:
    - Mobile Operators
    - Handset Manufacturers
    - Semiconductor Companies
    - Software Companies
    - Commercialization Companies
  - Led by Google with other 34 members
Android OS features

- Multiple handset layouts: VGA, 3D, traditional smartphone layouts...
- Connectivity: GSM, CDMA, Bluetooth, Wi-Fi, WiMAX, UMTS, etc...
- Messaging: MMS/SMS, Android Clout to Device (CD2M), Google Cloud Messaging (GCM)
- Web Browser (WebKit + V8) - 100% ACID on Android 4.0
- Java support (Dalvik)
- Additional hardware support (GPS, accelerometers, gyroscopes, ...)
- External storage (FAT32/Ext3/Ext4)
- Multi-tasking, multi-touch, video-calling, tethering, wide range of media supported formats, streaming support, etc...
How spread is Android?

- Over 330 million devices worldwide
- 934,000 daily activations in Q1 2012
- Android 2.3 (Gingerbread) being the most widespread as of June 2012 with 65% of the devices.

* Data above provided by Wikipedia
World-Wide Smartphone Sales (%)
How are third-party apps built?, where do they run?: The Dalvik VM

- Based on the Java VM
- Register-based
- Optimized for low memory requirements
- Applications are developed in Java and compiled into Java bytecode, a tool called dx translates them into Dalvik bytecode
- It is not a security boundary!
- Applications are allowed to run native code
How are applications distributed?

- Applications are self-contained within Android application package files (APKs)
- Google Play is the main distribution hub of Android Apps, as of September 2012:
  - ~675,000 applications available
  - 25 billion app downloads
- There are plenty of other alternative Android markets
- Online file-share services
Android’s Security Model
Android Permission scheme

• Apps that require additional capabilities must statically request them

• Android permissions:
  • android.permission.*
  • android.permission.RECEIVE_SMS
  • android.permission.INTERNET

• Apps may also share resources by declaring permissions enforced by their signature level

• Permissions are granted to the App by the user at install time
Application Isolation

• Apps run in a sandboxed environment.
• The application sandbox does not rely on the Virtual Machine technology they run in, it is instead based on:
  • Processes
  • File permissions
  • IPC mechanisms
• Isolates both native code and the Dalvik VM(s)
Application Signing

• Apps are signed with self-generated certificates
• No Certificate Authorities are involved
• Apps signed with the same certificate may share UID or get specific permissions granted
Android Threat Landscape
Top mobile threats

**Web- & Network-based Attacks**
Launched by malicious websites or compromised legitimate sites
Attacking site exploits device’s browser
Attempts to install malware or steal confidential data that flows through browser

**Malware**
Includes traditional computer viruses, computer worms and Trojan horse programs
Example: IKE worm targeted iOS-based devices
Example: Pjapps enrolled infected Android devices in botnet

**Social Engineering Attacks**
Leverage social engineering to trick users
Attempts to get users to disclose sensitive info or install malware
Examples include phishing and targeted attacks

**Resource Abuse**
Attempt to misuse network, device or identity resources
Example: Sending spam from compromised devices
Example: Denial of service attacks using computing resources of compromised devices

**Data Loss**
Employee or hacker exfiltrates sensitive info from device or network
Can be unintentional or malicious
Remains biggest threat to mobile devices

**Data Integrity Threats**
Attempts to corrupt or modify data
Purpose is to disrupt operations of an enterprise or for financial gain
Can also occur unintentionally
What’s of interest to malware creators in Android devices?

- Premium Rate Number Billing
- Spyware
- Search Engine Poisoning
- Pay-Per-Click
- Pay-Per-Install
- Adware
- mTAN stealing
Breakdown of mobile malware capabilities (2011)

- 28% Collect Data
- 25% Track User
- 16% Traditional Threats
- 7% Change Settings
- 24% Send Content

24 Collects Device Data
21 Spies On User
19 Sends Premium SMS
9 Downloader
9 Back Door
8 Tracks Location
4 Modifies Settings
4 Spam
4 Steals Media
2 Elevates Privileges
2 Banking Trojan
1 SEO Poisoning
Volume statistics

- Symbian signing
- Ikee worm on rooted phones

101 Families
16,000 Samples

Symbian	IOS	Android
Android Malware statistics

June 2012
> 16,000 malicious APKs identified

Symantec
The rise of polymorphism

Exponential growth of malicious APKs (families: samples)

1:8
The rise of polymorphism

Exponential growth of malicious APKs (families: samples)
The rise of polymorphism

Exponential growth of malicious APKs (families: samples)
Android.Opfake (2012)

Driving mobile polymorphism

Heavy use of social engineering

Fake app markets

Sends Premium SMSs

Unique (bad) APK every time

Unique (bad) APK every time

Send Premium SMSs

Unique (bad) APK every time
Behind the scenes: Android App-Infection
Anatomy of an Android Application package file (APK)

- Based on Java’s JAR file format
- Self-signed, signatures can be found within the META-INF directory
- **Android Manifest**, essential information about the application, including:
  - Java package name, components (activities, services..), required permissions, etc...
- **Classes.dex**, classes compiled in DEX format (Dalvik VM)
- May contain **native code**, usually within the `lib\{platform}\` directories.
- **Resources** within `res` directory
Infecting an APK for dummies, step by step guide

1. Unpack the given APK
2. Inject malicious code
3. Bump permissions up (if needed)
4. Repack the new APK
5. Sign APK with self-gen certificate
Threat propagation

1. Game is taken from the app market
2. Malicious code is added
3. Trojanized version is uploaded to file share sites
4. Unauthorized file share site/marketplace
5. File is replicated by other file shares/sites/marketplaces
6. Malicious version is downloaded
Demo: Android.Geinimi
Android.Genimi profile

- Discovered on December 2010
- One of the first known malwares with botnet capabilities for Android
- Uses a XML-based protocol + DES encryption
- Obfuscated code: ProGuard + encrypted strings (DES)
- Three wake-up entry points:
  - Execution of infected App
  - BOOT_COMPLETED broadcast
  - SMS_RECEIVED broadcast
- Implements 36 different server-side commands divided in two categories, some of them overlap
- Believed to be a proof of concept malware prototype
Thank you!

Mario Ballano
Symantec