Android Security

Stuart O. Anderson
June 23, 2011
Stuart O. Anderson

- Background in robotics and applied math
- Fellow at the Institute for Disruptive Studies
- Co-founded Whisper Systems with Moxie Marlinspike
Stuart O. Anderson
What this talk covers

The Android System
Android's security model
Malware and exploit examples
Best practices for improving security
The Android System: Overview

Android is
- A system architecture
- A business and legal framework

Security is affected by both aspects
Android: Embedded and Consumer

Android's design is driven by

- Resource constraints
  - Memory
  - Battery
- Consumer market
  - iPhone competition
Android System Architecture
Android Kernel

Modified for resource constrained environments

- Binder
- Ashmem and Pmem
- Logger
- Wakelocks
- Out-Of-Memory Handler
Android Userspace

Driven by resource and legal constraints

– Bionic (Non-POSIX libc)
– Prelinked system libraries
– Dalvik VM
– Native Libraries
Bionic: Android libc

BSD License
  – No GPL in userspace
Small
  – About 200K
Fast
  – Especially pthreads
Apriori: Android Prelinker

System libraries are internally pre-linked
Must be loaded at specific vaddr
Look in /build/core/prelink*.map
Dalkvik

Virtual Machine runs most apps
DEX byte code compiles from Java
Register and not stack based
  – i.e. trying real hard not to be a JVM
Native Libraries

Webkit
Media
SQLite
SurfaceManager
...

Android Framework

Components
- Activities
- Services
- Receivers
- ContentProviders
Android Framework

Intents connect components through Binder

- Action
- Data
- Categories
- Extras
- Flags – can grant permissions…
Android: Business Relationships

Google – Develops platform
Chipset vendors – Broad market
OEMs – Shorter time to market
Carriers – Easier to customize
Developers – Easy to publish, free SDK
OEMs

Chipset vendors are limited
  - Qualcomm, TI (OMAP3), Ericsson, Broadcom
  - Faster development cycle (9-12 months) for OEMs
  - Budget goes to differentiation
Carriers

Slow updates

– Known webkit bugs linger
  • M.J. Keith at Alert Logic
Google's Points of Control

- Access to latest source code
- Control of review process
- Proprietary apps (Market, Maps, …)
- Trademark
- AFA, CTS/CDD
Orphaned Devices

Last Google I/O

- 18 months support for new devices
- Verizon, HTC, Samsung, Sprint, Sony Ericsson, LG, T-Mobile, Vodafone, Motorola, and AT&T
Android: Future Directions

New Devices
  – Tablets
  – Readers
  – PCs / Dockables
Android's Security Model

Linux Kernel
- Process separation
- Access to resources by UID/GID

Android Framework
- Signed packages
- Per-package Permissions
Android UID and GID

Most packages have their own UID
Some share a UID
GID is used for Kernel level resources
   – Camera, bluetooth, display, …
## Android UID and GID

<table>
<thead>
<tr>
<th>app</th>
<th>UID</th>
<th>PID</th>
<th>UID Offset</th>
<th>GID</th>
<th>GID Offset</th>
<th>State</th>
<th>App Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_49</td>
<td>384</td>
<td>86</td>
<td>114796</td>
<td>33796</td>
<td>ffffffff</td>
<td>00000000 S com.android.launcher</td>
<td></td>
</tr>
<tr>
<td>app_37</td>
<td>385</td>
<td>86</td>
<td>94468</td>
<td>16152</td>
<td>ffffffff</td>
<td>00000000 S com.android.voicedialer</td>
<td></td>
</tr>
<tr>
<td>app_10</td>
<td>410</td>
<td>86</td>
<td>97044</td>
<td>19312</td>
<td>ffffffff</td>
<td>00000000 S com.android.vending</td>
<td></td>
</tr>
<tr>
<td>app_8</td>
<td>428</td>
<td>86</td>
<td>119840</td>
<td>23376</td>
<td>ffffffff</td>
<td>00000000 S com.google.process.gapps</td>
<td></td>
</tr>
<tr>
<td>app_27</td>
<td>480</td>
<td>86</td>
<td>97624</td>
<td>20496</td>
<td>ffffffff</td>
<td>00000000 S android.process.media</td>
<td></td>
</tr>
<tr>
<td>app_48</td>
<td>674</td>
<td>86</td>
<td>102452</td>
<td>20256</td>
<td>ffffffff</td>
<td>00000000 S com.google.android.apps.genie.geniewidget</td>
<td></td>
</tr>
<tr>
<td>app_26</td>
<td>686</td>
<td>86</td>
<td>97912</td>
<td>17880</td>
<td>ffffffff</td>
<td>00000000 S com.android.quicksearchbox</td>
<td></td>
</tr>
<tr>
<td>app_36</td>
<td>725</td>
<td>86</td>
<td>96092</td>
<td>18176</td>
<td>ffffffff</td>
<td>00000000 S com.cooliris.media</td>
<td></td>
</tr>
<tr>
<td>app_41</td>
<td>737</td>
<td>86</td>
<td>120740</td>
<td>22184</td>
<td>ffffffff</td>
<td>00000000 S com.google.android.apps.maps</td>
<td></td>
</tr>
<tr>
<td>app_18</td>
<td>764</td>
<td>86</td>
<td>103200</td>
<td>20160</td>
<td>ffffffff</td>
<td>00000000 S com.google.android.voicesearch</td>
<td></td>
</tr>
<tr>
<td>app_12</td>
<td>824</td>
<td>86</td>
<td>94336</td>
<td>15836</td>
<td>ffffffff</td>
<td>00000000 S com.whispersys.updater</td>
<td></td>
</tr>
<tr>
<td>app_9</td>
<td>832</td>
<td>86</td>
<td>97516</td>
<td>16112</td>
<td>ffffffff</td>
<td>00000000 S com.whispersys.monitor</td>
<td></td>
</tr>
</tbody>
</table>
Android Framework Security

Code Signing
  – Links a package to a developer

Permissions
  – Grants a package a capability
Code Signing

Packages are signed when published
– You trust the publisher with the security of their private key
– If the keys don't match, app must be manually removed and reinstalled
– Packages that share keys can share UIDs
Remote Pull and Push

Google can add and remove packages
  – GtalkService
  – Malware may attempt to disable these features
Permissions

Every UID has an associated set of permissions it has been granted:

android.permission.SEND_SMS
android.permission.WRITECALENDAR
android.permission.READ_PHONE_STATE
Permissions

Packages request permissions in their manifest

User is prompted to approve these permissions as a single block

– Only once, at install time
– Permissions not marked 'dangerous' are not displayed
Permissions

Most permissions declared in

– /core/res/AndroidManifest.xml

Not all permissions require user approval

– Signature

– SignatureOrSystem
Permissions: Granularity

Granularity in the permissions themselves
   – Internet is a single permission

Granularity in user control
   – Can't approve a subset of the requested permissions
Permissions: Granularity

Too fine granularity overloads users
Overloaded users stop paying attention
Permissions: Enforcement

Permission checks are performed in PackageManagerService

```java
public int checkUidPermission(String permName, int uid) {
    synchronized (mPackages) {
        Object obj = mSettings.getUserIdLP(uid);
        if (obj != null) {
            GrantedPermissions gp = (GrantedPermissions)obj;
            if (gp.grantedPermissions.contains(permName)) {
                return PackageManager.PERMISSION_GRANTED;
            }
        } else {
            HashSet<String> perms = mSystemPermissions.get(uid);
            if (perms != null && perms.contains(permName)) {
                return PackageManager.PERMISSION_GRANTED;
            }
        }
    }
    return PackageManager.PERMISSION_DENIED;
}
```
Permissions: Services

Services must explicitly check permissions at IPC entry points

```java
public void call(String number) {
    // This is just a wrapper around the ACTION_CALL intent, but we still
    // need to do a permission check since we're calling startActivity()
    // from the context of the phone app.
    enforceCallPermission();

    String url = createTelUrl(number);
    if (url == null) {
        return;
    }

    Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse(url));
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    intent.setClassName(mApp, PhoneApp.getCallScreenClassName());
    mApp.startActivity(intent);
}
```
Permissions: ContentProviders

Read and Write permissions handled by system
Must implement per-URI permission granting
Malware and Exploit Examples

- Getting root
- Remote exploits
- Protocol weaknesses
- Making money
Leaky Apps

Content Providers, SD/Card
Network communication
  – Spoofed http responses
  – Authtokens
Unreliable deputies
GSM Weaknesses

Well publicized attacks on GSM
– See Karsten Nohl
The cost of intercept equipment is marginal
Privilege Elevation

Send an Intent or Binder data to another app that causes unexpected behavior
– Some critical services have very complicated interfaces

Change your own uid or gid
– Kernel, zygote, etc
Android Exploit Examples

Sebastian Krahmer (stealth)
  – Zimperlich
    • Forkbomb to process limit
    • Zygote will fail to change uid from root on fork
  – Gingerbreak
    • Unchecked array index in vold
    • Rewrite GOT entry for strcmp()
Android Remote Exploit Examples

Colin Mulliner
- NFC remote application crash
- NFC remote NFC service crash

Charlie Miller
- PacketVideo media library
Malware Threats

Jon Oberheide

- Rootstrap
- Download and execute exploits as they become available
Malware Threats

Untargeted Monetization
- Premium SMS
- 1-900 Numbers

Persistence
- Remount /system r/w
- Turn off AV tools
Solutions and Best Practices

System Level Changes
Security Applications
Auditing Applications
System Level Changes

- Full disk encryption
- Dynamic egress filtering
- Selective permissions
- Extended code signing
Disk Encryption

Honeycomb
– MTD devices only
– Tied to screen lock

WhisperCore
– yaffs variant supports MTD and block devices
– Enhanced screenlock
Dynamic Information Flow Tracking

DIFT inside the Dalvik VM

TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones

Dynamic Information Flow Tracking

Variable tracking in Dalvik
Message tracking in Binder
Method tracking in system libraries
File tracking via file-system extension
Dynamic Egress Filtering

Monitor outgoing network connections.
Filter connections by:
  – Initiating app.
  – Destination.
  – Network type and location.

Prompts when connections are initiated
Selective Permissions

Remove specific permissions
Create temporary and isolated copies of requested resources
Selective Permissions

Remove specific permissions
Create temporary and isolated copies of requested resources
Exploit Mitigation

Address Space Randomization for Mobile Devices
– Hristo Bojinov, Dan Boneh, Rich Cannings, Iliyan Malchev – WiSec 2011

– Randomizes addresses
  • Even with prelinked libraries
– Android moving to ld.so
– Still forking zygote?
Extended Code Signing

Management of which apps can run
- Whitelist or blacklist
- Installed apps can be blocked

Lets administrators sign, update, install, and remove apps remotely
Security Applications

- Secure backup
- Secure communications
- Secure storage
Secure Backup

Secure incremental backup
Cloud or backend storage
Remote image management
Remote wipe
Secure Communication

Voice calls
  – VoIP solutions: RedPhone, PrivateWave, Cellcrypt

Messaging
  – SMS/MMS/IM

Email
  – Good, MobileIron, TouchDown
Malware Detection

Google
- Can remove malware from Market
- Can remotely disable and update

Lookout
- At the endpoint, limited access
- Can be disabled by malware
Auditing an Application

Examine the Manifest
Decompilers
Other Tools
Auditing an Application

ISEC's Manifest Explorer

Author: Jesse Burns
Auditing an Application

Use adb to pull the apk from the phone

adb pull /data/app/packagename.apk
adb pull /system/app/packagename.apk
Auditing an Application

Use dedex (Nathan Keynes) and jd-gui to inspect DEX code

unzip package.apk
dedex classes.dex
jd-gui classes.jar
public void onResume()
{
    super.onResume();
    redraw();
}

public void onStop()
{
    int k = 0; String str3 = "Allow outgoing connection?"; super.onStop();

    boolean bool = this.verdictRegistered; if (!(bool)) {
        String str1 = "notification"; Object localObject1 = getSystemService(str1); localObject1 = (NotificationManager)localObject1
        Notification localNotification = new android/app/Notification; int i = 2130837513; String str2 = "Allow outgoing co
        Intent localIntent = getIntent(); PendingIntent localPendingIntent = PendingIntent.getActivity(this, k, localIntent
        int j = localNotification.flags; j |= 16; localNotification.flags = j;
        Object localObject2 = "Allow outgoing connection?"; localObject2 = this.label; localObject2 = (TextView)localObject2
        ((NotificationManager)localObjectObject1).notify(k, localNotification);
    }
}

private class DenyClickListener
implements View.OnClickListener
{
    DenyClickListener( DecisionActivity.this param1)
    {
        this(paramDecisionActivity);
    }

    public void onClick() { Object localObject1 = new android/content/Intent; Object localObject2 = this.this$0; Whispers
    localObject2 = "com.whispersys.monitor.VERDICT_ACTION"; ((Intent)localObjectObject1).setAction((String)localObjectObject2);
Other Audit Tools

- Dynamic Information Flow Tracking
  - TaintDroid
- Mandatory Access Control
  - TOMOYO Linux
- Emulator
  - Scott Dunlop's JDWP->JDP method
- Network Monitoring
  - WhisperMonitor
  - Wireshark
Summary: Android Security

Embedded and consumer
Tradeoffs made against security
Divided responsibility for security
System and application layer solutions