The Risk you carry in your Pocket

Nils

Black Hat
Abu Dhabi 2010

MWR InfoSecurity
Who Am I?

- Head of Research @ MWR
- Exploiting stuff before…
  - Microsoft, Google, Adobe, IBM, Mozilla, Sun, Linux, Apple …
- Pwn2Own Winner 2009
  - Safari, IE and Firefox
- Pwn2Own Winner 2010
  - Firefox on Windows 7
• Demo
• Introduction
• Android Sandbox
• Android IPC
• Vulnerabilities
• Demo
• Conclusion
• Q&A
• Demo
• Introduction
• Android Sandbox
• Android IPC
• Vulnerabilities
• Demo
• Conclusion
• Q&A
Introduction

- Prerequisites:
  - I have got a WebKit vulnerability
- Can own:
  - iPhone
  - Palm Web OS
  - Android

- In Android I am limited to the Sandbox
  - Access to Passwords, Cookies, etc…
Introduction

- I want more Privileges
- Record Audio
Introduction

- Research on Android Phone
  - Not emulator
  - HTC Legend
    - Android 2.1
- Some apps
What will you see?

• How to:
  • Audit a Android Handset
  • Additions by Vendors
  • And Carriers
  • Audit Android Applications

• And how to exploit the findings
Android – Previous Research

- Kernel vulnerabilities:
  - E.g. `sock_sendpage()`

- Local vulnerabilities:
  - E.g. `adb root` vulnerability
  - Fork bomb
  - Setuid return value not checked
• Demo
• Introduction
• Android Sandbox
• Android IPC
• Vulnerabilities
• Demo
• Conclusion
• Q&A
Android – Sandbox

- Applications are Sandboxed
- Using Linux User/Group model
- Every Application == 1 User
  - In theory ...
- Communication through IPC
- Permissions
Android – Permissions

- Applications request Permissions
  - AndroidManifest.xml
- Pre-installed apps
  - Set-up by default in phone
- User installed apps
  - Granted by User during installation
  - Limited
Android – Permissions

• Examples:
  • android.permission.CALL_PHONE
  • android.permission.RECORD_AUDIO
  • android.permission.INSTALL_PACKAGE
Android – IPC

• Inter-Process Communication
  • Used by all of the Apps
  • Core feature on Android
  • Protected using Permissions

• Mechanism:
  • Services
  • Content-Providers
  • Broadcasts
  • Activities
Android – IPC

• Supported by /dev/binder
  • Kernel
  • Message routing
  • Permission enforcement

• Messages in “Parcels”
  • Intents special Parcels
Android – Intent

- Serialised Data Structure
- Sent to IPC endpoints
- Contain Extras
  - Strings
  - Primitive Data Types
  - Arrays thereof
  - Serialisable Java Objects (!)
Android – Service

- Similar to RPC
- Class extends Service.class
  - Public methods are exported
  - Called through Intents
- Defined in AndroidManifest.xml:

```xml
<service android:name="BluetoothHeadsetService">
    <intent-filter>
        <action android:name="android.bluetooth.IBluetoothHeadset" />
    </intent-filter>
</service>
```
Android – Activity

• Visual Components of Applications
• Application can instantiate them
  • Sometimes
  • Take arguments in Intents
  • Will run in Implementing Process
  • Permissions!
Android – Content-Providers

• Provide Access to any Data
  • Emails
  • Pictures
• Often backed by SQLite Databases
• Content-Resolver
• URI: content://browser/bookmarks
• Standard Interface using Cursors
• Write and Read Permissions
• Not using Intents
Android – Broadcast Receivers

- Register to Broadcast Messages
  - System and Custom
- Some Messages are protected
  - Others can be forged by anyone
- Arguments in Broadcasts
  - Intents
- AndroidManifest.xml
  - Can register dynamically as well
Android – IPC Exports

- Default IPC exports
- Exported by default
  - Content-Providers
- Export depends on set Filters
  - Services
  - Broadcast Receivers
  - Activities

- Developers aware of that?
Android – Privilege Escalation

- Any vulnerability in any exported:
  - Service, Content-Provider
  - Broadcast Receiver or Activity

- Can lead to privilege Escalation
  - Gaining privileges of vulnerable App
Many Apps on the phone
- All in different Processes (Theoretically)
Default Android apps
- ~ 70 apps
Vendor apps
- HTC: ~ 60 apps
- Plus carrier apps!
User installed apps
- Many more
Android – Processes

- 1 User ⇔ 1 App
- Multiple processes per App
- Not on real phones though
  - Shared User Id’s
    - Across apps
  - Shared processes
    - Across apps

=> Shared Permissions and Access-rights
Android – Shared UIDs

• Applications can Share UserIds
• If signed by same Developer Key
  • Or Pre-installed
• Pro:
  • Performance
• Contra:
  • Security
Android – Shared UIDs

• Example:
  • com.htc.WeatherWidget

• Permissions:
  android.permissions.GET_ACCOUNTS, android.permission.READ_SYNC_SETTINGS
Android – Shared UIDs

• Example:
  • com.htc.WeatherWidget
  • Shares “com.htc.rosie.uid.shared” with:

  com.htc.FriendStreamWidget, com.htc.TwitterWidget,
  com.htc.htcmailwidgets, com.htc.NewsReaderWidget,
  com.htc.StockWidget, com.htc.widget.clockwidget,
  com.htc.htccalendarwidgets, com.htc.footprints.widgets,
  com.htc.htcontactwidgets, com.htc.htcmsgwidgets,
  com.htc.htcsyncwidget, com.htc.launcher, com.htc.WeatherWidget,
  com.htc.htcsettingwidgets, com.htc.photo.widgets,
  com.htc.htcbookmarkwidget, com.htc.MusicWidget,
  com.htc.htcsearchwidgets
Android – Shared UIDs

- Example:
  - com.htc.WeatherWidget

- Permissions:
  - android.permissions.GET_ACCOUNTS, android.permission.READ_SYNC_SETTINGS
Android – Shared UIDs

• Example:
  • com.htc.WeatherWidget

• Shared Permissions:

```java
android.permissionINTERNET, com.htc.htctwitter.permission.useprovider, android.permission.ACCESS_FINE_LOCATION,
android.permission.ACCESS_NETWORK_STATE, android.permission.ACCESS_WIFI_STATE, android.permission.GET_ACCOUNTS,
android.permission.READ_SYNC_SETTINGS, android.permission.READ_CALENDAR, android.permission.WRITE_CALENDAR,
com.google.android.googleapps.permission.GOOGLE_AUTH.mail, android.permission.READ_CONTACTS,
android.permission.CALL_PHONE, android.permission.CALL_PRIVILEGED, android.permission.READ_SMS,
com.htc.socialnetwork.permission.useprovider, android.permission.RECEIVE_BOOT_COMPLETED, android.permission.WRITE_CONTACTS,
android.permission.RECEIVE_SMS, android.permission.RECEIVE_MMS, android.permission.SEND_SMS, android.permission.VIBRATE,
android.permission.WRITE_SMS, android.permission.WRITE_CONTACTS,android.permission.CHANGE_NETWORK_STATE, android.permission.READ_PHONE_STATE,
android.permission.WAKE_LOCK, android.permission.EXPAND_STATUS_BAR, android.permission.GET_TASKS, android.permission.SET_WALLPAPER,
android.permission.WRITE_WALLPAPER_HINTS, android.permission.WRITE_SETTINGS, com.htc.launcher.permission.READ_SETTINGS,
com.htc.launcher.permission.WRITE_SETTINGS, android.permission.WRITE_EXTERNAL_STORAGE, android.permission.BROADCAST_STICKY,
android.permission.WRITE_SECURE_SETTINGS, android.permission.CHANGE_WIFI_STATE,
android.permission.CLEAR_APP_USER_DATA, android.permission.MODIFY_PHONE_STATE, android.permission.ACCESS_COARSE_LOCATION,
android.permission.WRITE_APN_SETTINGS, android.permission.ACCESS_CHECKIN_PROPERTIES, android.permission.BLUETOOTH,
android.permission.BLUETOOTH_ADMIN, android.permission.ACCESS_WIMAX_STATE, android.permission.CHANGE_WIMAX_STATE,
android.permission.ACCESS_LOCATION_EXTRA_COMMANDS, android.permission.ACCESS_LOCATION, android.permission.ACCESS_ASSISTED_GPS,
android.permission.ACCESS_NETWORK_LOCATION, android.permission.ACCESS_GPS,
com.android.browser.permission.READ_HISTORY_BOOKMARKS, com.android.browser.permission.WRITE_HISTORY_BOOKMARKS
```
- Demo
- Introduction
- Android Sandbox
- Android IPC
- Vulnerabilities
- Demo
- Conclusion
- Q&A
Android – Vulnerabilities

- SQL injection in Content Providers
  - When backed by SQLite
- Allows for arbitrary reads in databases
  - Across processes
- Can be filtered by Developer
  - Usually is not
- Not encouraged by Dev Docs
- Have not found instances of writes to DB
- No useful functions (load_extension()…)
final Cursor query(
    Uri uri,
    String[] projection,
    String selection,
    String[] selectionArgs,
    String sortOrder);
```java
final Cursor query(
    "content://settings/system",
    null,
    null,
    null,
    null,
    null);

SELECT * FROM system;
```
final Cursor query("content://settings/system",
null,
"_id=1",
null,
null);

SELECT * FROM system WHERE _id=1;
final Cursor query(
    "content://settings/system",
    null,
    "(select count(*) from secure where \n     name='adb_enabled' and value='0')=0",
    null,
    null);

SELECT * FROM system WHERE "(select count(*) from secure where name='adb_enabled' and value='0')=0;
final Cursor query="content://settings/system",
{"_id"},
null,
nul,
nul,
nul);

SELECT _id FROM system;
final Cursor query(
    "content://settings/system",
    {"" * FROM bluetooth_devices;""},
    null,
    null,
    null);

SELECT * FROM bluetooth_devices; FROM system;
final Cursor query(
    "content://settings/system",
    \{" * FROM sqlite_master;"\},
    null,
    null,
    null);

SELECT * FROM sqlite_master; FROM system;
Android – Vulnerabilities

- Unprotected services
- Example:
  - Introduced by HTC
  - `com.htc.soundrecorder.RecordingService`
  - Not protected
  - Explicitly exported
  - `android.permission.RECORD_AUDIO`
  - Now useless
  - Every HTC Android phone I checked
Android – Native APIs

- Java less prone to Memory Corruptions
- Native APIs more promising for Review
- Services
  - Directly exporting native API’s
- Keep a look out for:
  - `loadLibrary("")`
  - And “ native ” keyword
char mJetFilePath[256];

int JetPlayer::loadFromFile(const char* path) {
    ... 
    strncpy(mJetFilePath, path, strlen(path));

public boolean loadJetFile(String path) {
    return native_loadJetFromFile(path);
}
Android – Others

• Let’s be creative
• Applications do all kinds of stuff
  • Some of which is stupid :P

• Example: Skype - App

# ls -al /data/data/com.skype.raider/files/skypekit
-rwxrwxrwx 1 0 2000 43 /data/data/com.skype.raider/files/skypekit
Permissions:

- android.permission.DISABLE_KEYGUARD
- android.permission.WAKE_LOCK
- android.permission.INTERNET
- android.permission.GET_ACCOUNTS
- android.permission.READ_CONTACTS
- android.permission.ACCESS_NETWORK_STATE
- android.permission.VIBRATE
- android.permission.MODIFY_AUDIO_SETTINGS
- android.permission.RECORD_AUDIO
- android.permission.READ_PHONE_STATE
- android.permission.ACCESS_COARSE_LOCATION
- android.permission.GET_TASKS
- android.permission.AUTHENTICATE_ACCOUNTS
- android.permission.MANAGE_ACCOUNTS
- android.permission.READ_SYNC_SETTINGS
- android.permission.WRITE_SYNC_SETTINGS
- android.permission.GET_ACCOUNTS
- android.permission.USE_CREDENTIALS
- android.permission.WRITE_SETTINGS
- android.permission.WRITE_SECURE_SETTINGS
- android.permission.READ_CONTACTS
- android.permission.WRITE_CONTACTS
- android.permission.READ_SYNC_STATS
- android.permission.WRITE_EXTERNAL_STORAGE
Android – Deserialisation

- Intents contain Extras
  - Can be Serialisable
- Object type is checked after deserialisation
- Arbitrary objects can be deserialised
  - In other Processes
    - Across trust boundaries
    - With other permissions
- Is this exploitable?
  - Sami?
THE FOLLOWING **SLIDE** HAS BEEN APPROVED FOR

ALL SECURITY PROFESSIONALS

THE PRESENTATION HAS BEEN RATED

<table>
<thead>
<tr>
<th>PG-18</th>
<th>VENDORS STRONGLY CAUTIONED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Some Material May Be Inappropriate for Children Under 18</td>
</tr>
<tr>
<td></td>
<td><strong>COARSE PROGRAMMING PRACTICE</strong></td>
</tr>
</tbody>
</table>
Android – Permissions

- Most useful Permission: INSTALL_PACKAGES
- On HTC phones granted to the Browser
  - That’s True!
- Why
  - Flashlite Flash player
  - Installs updates using PackageManager
  - Needs Permissions for that …
Android – Permissions

- INSTALL_PACKAGES in Browser
- Impact
  - Malicious Code in Browser
  - Installs arbitrary Applications
    - Without prompting the User
  - Gains arbitrary Permissions
    - For malicious applications
    - No restricted permissions
Android – Demo

- That should be enough…

Demo Time!
• Demo
• Introduction
• Android Sandbox
• Android IPC
• Vulnerabilities
• Demo
• Conclusion
• Q&A
Android Demo - Vulnerability

- Use-after-free in Browser
- WebKit
  - Android, Chrome, Safari, iPhone, Symbian, Palm Pre and more
- Allows for arbitrary code execution
- HTML5
  - Introduced in Android 2.0
  - 1.5 and 1.6 not vulnerable
- JavaScript
- Patched in 2.2
- No NX, No ASLR
Android - Use-after-free in Browser

Object 1

Object 2

DOM Object
Android - Use-after-free in Browser

Object 2

FREE
Android - Use-after-free in Browser

Object 2

data data data
  data data data
  data data data
Android - Use-after-free in Browser

Object 2

(*data)()

data data data data data data data data
Android - Shellcode

- Steps:
  1. Connect back to Attacker
  2. Upload malicious APK
  3. Install from Browser
  4. Pwnage!
Android - Demo
THE FOLLOWING DEMO HAS BEEN APPROVED FOR

ALL SECURITY PROFESSIONALS

THE PRESENTATION HAS BEEN RATED

<table>
<thead>
<tr>
<th>PG-18</th>
<th>VENDORS STRONGLY CAUTIONED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Some Material May Be Inappropriate for Children Under 18</td>
</tr>
<tr>
<td></td>
<td>0-DAY, SHELLCODE and MOBILE ROOTKIT</td>
</tr>
</tbody>
</table>
Android Proof-of-Concept

• Reported the vulnerability to vendors
  • Patched in 2.2
• However
  • Any WebKit vulnerability will do
• Not patched in most Phones
• Demo
• Introduction
• Android Sandbox
• Android IPC
• Vulnerabilities
• Demo
• Conclusion
• Q&A
Conclusion

- Understand the Threats
- Android Sandbox
  - Fairly Reasonable
- Many bugs introduced by:
  - Vendors, Carriers
  - 3rd Party Apps
- Testing and Assurance
  - For Phones
  - Not just OS
• Demo
• Introduction
• Android Sandbox
• Android IPC
• Vulnerabilities
• Demo
• Conclusion
• Q&A
Questions?