The Droid Exploitation Saga

Subho Halder @sunnyrockzzs | Aditya Gupta @adi1391

Thursday, 6 December 12
Who are we!

- Information security researcher
- Mobile exploiter
- Creator of afe (Android framework for exploitation)
- Python lovers
- Co-founder of xysec.
- Found bug in some famous websites including Google, Apple, Microsoft, Skype, Adobe and many more
SOME COMPANIES WE’VE FOUND VULNS IN...

And MORE...

Thursday, 6 December 12
Agenda

- Quick Intro to Android
- Android Security Model
- Creating Android malwares
- Android Botnets
- Injecting malwares into another app
- Content Providers
- Creating own modules for AFE
- Fuzzing, Penetration Testing with AFE
Android

- Open-sourced platform by Google Inc
- Generic builds, which can be deployed in any Hardware Configuration.
- Linux Kernel, Webkit Browser, Open-Sourced Applications
# Android Architecture

## 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
- XMPP Service

## Libraries
- Surface Manager
- Media Framework
- SQLite
- Core Libraries
- OpenGL
- FreeType
- LibWebCore
- Dalvik Virtual Machine
- SSL
- Libc
- Display Driver
- Camera Driver
- Bluetooth Driver
- Flash Memory Driver
- USB Driver
- Keypad Driver
- WiFi Driver
- Audio Drivers
- Binder (IPC) Driver
- Power Management

## Linux Kernel
Android Applications

- Written mainly in Java + little XML
- Composed of components:
  - Activities
  - Services
  - Intents
  - Content Providers
  - Broadcast Receivers
Security Architecture

- Apps run in a virtual env/sandbox
- Privilege Separation
- Each app has its own UID n GID
- ASLR in most places (from >4.0)
- DVM
- IPC - Inter Process Communication
- Dalvik VM != Sandbox
## Security Architecture

<table>
<thead>
<tr>
<th>USER</th>
<th>PID</th>
<th>PPID</th>
<th>VSIZE</th>
<th>RSS</th>
<th>WCHAN</th>
<th>PC</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>1</td>
<td>0</td>
<td>368</td>
<td>220</td>
<td>c0077dc0 00000000cc</td>
<td>S/init</td>
<td>/init</td>
</tr>
<tr>
<td>root</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>c009015c 00000000</td>
<td>S kthread</td>
<td>kthread</td>
</tr>
<tr>
<td>root</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>c007aeec 00000000</td>
<td>S ksoftirqd/0</td>
<td>ksoftirqd/0</td>
</tr>
<tr>
<td>root</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>c00aeac4 00000000</td>
<td>S watchdog/0</td>
<td>watchdog/0</td>
</tr>
<tr>
<td>root</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>c008c214 00000000</td>
<td>S events/0</td>
<td>events/0</td>
</tr>
</tbody>
</table>

| system | 19682 | 1304 | 135620 | 15020 | ffffffff ffff0520 | S com.sec.android.providers.drm |
| app_78 | 19770 | 1304 | 146072 | 23376 | ffffffff afd0c5bc | S com.whatsapp |
| radio  | 19788 | 1304 | 138720 | 20488 | ffffffff afd0c5bc | S com.wssyncmlmdm |
| app_41 | 19807 | 1304 | 135888 | 16740 | ffffffff afd0c5bc | S com.sec.android.widgetapp.dualclock |
| app_39 | 19816 | 1304 | 157876 | 23580 | ffffffff afd0c5bc | S com.google.android.apps.maps:GoogleLocat |
Permission Model

- Defined in AndroidManifest.xml
- Displayed to user when installing the app
- Not exactly a XML file
- Defines the package name, version name, min SDK level required and the permission
Which one?

- An app with just **INTERNET** permission
- An app asking for **ALL** permissions
- An app asking for **READ_LOGS** permission
- An app asking for **0** permission
Bypassing the permission model

- INTERNET
- ACCESS FILES FROM SDCARD
- JUICY INFORMATION

- Use Browser, upload using GET
- No Permission Needed!
- Use READ_LOGS

Thursday, 6 December 12
Content Providers

- A content provider manages access to a central repository of data.
- The provider is part of an Android application, which often provides its own UI for working with the data.
- A content provider is identified by a content URI.
Accessing a Content Provider

- Example of getting a list of words from the User Dictionary provider:

```java
// Queries the user dictionary and returns results
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI, // The content URI of the words table
    mProjection, // The columns to return for each row
    mSelectionClause, // Selection criteria
    mSelectionArgs, // Selection criteria
    mSortOrder); // The sort order for the returned rows
```

- The content URI of the words table is: `content://user_dictionary/words`

- Read permission for accessing the content provider is also needed in the manifest file:

```xml
<uses-permission android:name="android.permission.READ_USER_DICTIONARY"/>
```
Content Providers
What about Google Bouncer!

- Virtual Environment to check if a particular app is malicious
- Runs the app in a phone/environment before publishing to the market
- Detects most of the malwares
Android Malware Surges Despite Google’s Efforts To Bounce Dodgy Apps Off Its Platform; F-Secure IDs 51,447 “Unique Samples” In Q3

source: Fsecure
Android Malwares
(common features)

- Send SMS to premium numbers
- Subscribe to premium services
- Dial premium numbers
- Steal messages, contact list, call logs
- Steal SD Card files
- Autorespond to text messages with some predefined format
Creating a malware

- Use **Content Providers** to get all the information
- **Cursors & SQLite** Databases
- Write Java codes like crazy
- Send that data to remote server using **HTTP**
- Set up a **PHP** file to listen to incoming data
- Save it to a **SQL** Database
How can you Automate these?
AFE Internals

Plugin Based Architecture

Modules

Python Based

Libraries
AFE Perspective

Offensive
- Malware Creation
- BotNet Automation
- Crypting
- Injecting

Defensive
- Content Query
- App Assessment
- Fuzzing
- Kernel Assessment

Thursday, 6 December 12
Creating Malwares/Botnets

Set Reverse IP
Change APP Name
Stealer
Build it!
Upload malformed APK
Back

Set your reverse IP, you can either start the listener and listen or send data to your own server by providing full path including http://

Steal Call Logs only
Steal SMS Inbox only
Steal Contacts only
Steal Call Logs and SMS
Steal Call Logs and Contacts
Steal SMS and Contacts
Steal all

Build Completed in 6 seconds. Find the created apk in /output folder
DEMO
(MALWARES WITH AFE)
Botnets

- More popular in PCs
- Gradually coming to mobile
- Already seen some cases
- Harder to detect in mobile phones
- AVs not efficient
- C & C easier in PCs
Botnets

- Can be used to get reverse shell + all malware features
- C & C over HTTP  SMS
- Battery consumption increases suspiciously with HTTP
- Can even execute shell commands with SMS
- Get the output as an SMS
- No notification on the victim’s phone
Botnets

- Operated all over just SMS
- No need of r00t
- Incoming messages won’t show a notification
- Identify each slave with its unique ID
- Remote Shell: `xysec cat /proc/version`
- Further exploitation
- Botnet = $$$

Thursday, 6 December 12
DEMO
(BOTNETS WITH AFE)
Fake legitimate Apps

- Malware services generally injected in legitimate applications
- How to do it?

1. legitimate apk
2. Smali/Java
3. Add malicious services/classes
4. Recompile
5. Final malware
Or Use AFE
Stealing Content Providers

- Catch application
- Over a million downloads
- Saves its notes using vuln content provider
- POC
Bypassing the anti malwares
Bypassing the anti malwares

Matches the signature with its database
Bypassing the anti malwares

Matches the signature with its database

Checks the activity, service and other class names
Bypassing the anti malwares

- Matches the signature with its database
- Checks the activity, service and other class names
- Checks the names of the variables
Bypassing the anti malwares

- Matches the signature with its database
- Checks the activity, service and other class names
- Checks the names of the variables
- Checks the control flow graph
Matches the signature with its database

Checks the activity, service and other class names

cHECKS THE NAMES OF THE VARIABLES

Checks the control flow graph

Rebuild + Zipalign
Matches the signature with its database

Checks the activity, service and other class names

cchecks the names of the variables

Checks the control flow graph

Modifies the classnames and all its references within files
Matches the signature with its database

Checks the activity, service and other class names

checks the names of the variables

Checks the control flow graph

Split variables into two, and append at runtime
Matches the signature with its database

Checks the activity, service and other class names

checks the names of the variables

Checks the control flow graph

Add dummy loops to change CFG
Early Detection

<table>
<thead>
<tr>
<th>SHA256:</th>
<th>718910c7d9ebab4d6a19b7f1be64b6ca7978920ae1c01e6a37df30b017d7221</th>
</tr>
</thead>
<tbody>
<tr>
<td>File name:</td>
<td>file-4832922_apk</td>
</tr>
<tr>
<td>Detection ratio:</td>
<td>30 / 46</td>
</tr>
<tr>
<td>Analysis date:</td>
<td>2012-12-03 18:52:50 UTC (1 day, 11 hours ago)</td>
</tr>
</tbody>
</table>
After Crypting

**virus total**

- **SHA256**: b4dc06304259198a361c180d36b5bfc85c36e4dd10b4cae06f20c3780eeddc99
- **SHA1**: 5ea259c8e1bad5c67c0947e671559d95177ece36
- **MD5**: 9e40e3b02f4e664390c7c6ab3f4022c0
- **File size**: 68.4 KB (69992 bytes)
- **File name**: 1-stringcrypt.apk
- **File type**: Android
- **Detection ratio**: 4 / 46
- **Analysis date**: 2012-12-25 05:29:29 UTC (0 minutes ago)

**Less details**

---

*XY Security Consultancy Services*

*Thursday, 6 December 12*
Checking your apps for vulnerabilities

- Find out leaky content providers
- How to find content providers a particular app is using? (use AFE)
- Try extracting the contents of that Content Provider using another app (use AFE)
- Insecure file storage (use your brain)
- Insecure data transmission (Use Proxy)
- Authentication + Other problems
Being secure

- Use obfuscators such as ProGuard (uses Name obfuscation, optimization, CFG obfuscation) or Dasho
- Check before you publish
- Have your “USB Debugging” turned OFF
- Don’t rely on AV
- Too paranoid? Reverse before you use. :)}
QUESTIONS?

security@xysec.com
http://afe-framework.com
https://github.com/xysec/AFE