Reviewing Attacks on Android

Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology

Acknowledgments: Some of the slides are fully or partially obtained from other sources. Reference is noted on the bottom of each slide, a full list of references is provided on the last slide.
History

• First commercial hand held cell phone 1983 (1362)
  • used embedded systems
The first smartphone

• IBM Simon 1993 (1372)
  • touchscreen, email
  • Based on ROM-DOS

• After ROM-DOS
NEW MOBILE THREAT FAMILIES AND VARIANTS, Q1-Q3 2013

- Q1 2013: 153 Android, 13 Symbian, 0 Others (Blackberry, iOS, Windows Phone)
- Q2 2013: 205 Android, 3 Symbian, 0 Others (Blackberry, iOS, Windows Phone)
- Q3 2013: 252 Android, 7 Symbian, 0 Others (Blackberry, iOS, Windows Phone)
Questions we investigate

- People at google are smart, latest security measures are being used, could there be any problems?

- Wouldn’t upgrading my android definitely improve my security?

- No microphone permission, so would there be any risk of eavesdropping?

- I have no private info on my smartphone, would there be any privacy risks?
Security Hardening on Android

Underlying Operating System
- Address Space Layout Randomization (ASLR)
- Data Execution Prevention (DEP)

Mobile Platform
- Permission Model
- App code signing

Permission Model and App code signing are enhancements built on top of the underlying operating system, which includes Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP).

References:
- [Lee2014]
ASLR (Address Space Layout Randomization)

- To implement many of the attacks, location of loaded codes in the memory should be known.
- ASLR randomized the layout for each process.
- Implemented in many OSes:
  - Linux
  - Android 4.1 implements ASLR
  - Mac OS
  - Windows
  - . . . . .
Performance Prioritized Designs of Android

- Multi-layered architectures
  - Android Applications run on Dalvik VM
  - with additional runtime libraries

- \(\rightarrow\) Slow app launch time
Process creation module

- Application
  - Android Runtime Library
  - Dalvik VM

- Application
  - Android Runtime Library
  - Dalvik VM

The zygote process: a template process hosting apps

Fast app launch time!
Weakened ASLR effectiveness

- All apps have the same memory layouts
  - For shared libraries loaded by the Zygote process

Weakens Android ASLR security

[Lee2014]
Attacking weakened ASLR: Remote Coordination Attack

1. Chrome is tricked into loading a malicious JavaScript.
2. The malicious JavaScript exploits the information leak vulnerability (CVE-2013-0912).
3. A URI Intent is used to initiate communication and install a crafted video file.
4. The crafted video file exploits the control-flow hijack vulnerability with leaked memory layout information.

[Lee2014]
Attacking weakened ASLR: Local Trojan Attack

- Zero---permission trojan app
  - Asks (almost) no permissions to the system
  - Scanning memory spaces using the native code
  - Layout information can be exported
    - Intent
    - Internet
- Once the trojan app is installed, ASLR can be easily bypassed
Questions we investigate

• People at google are smart, latest security measures are being used, could there be any problems?

• Wouldn’t upgrading my android definitely improve my security?

• No microphone permission, so would there be any risk of eavesdropping?

• I have no private info on my smartphone, would there be any privacy risks?
Mobile OS Updating (Android)

- More complex
  - Sandboxed apps
  - Lots of sensitive user data
  - Updating live system
- More often
- More files
  - 15,525 files from 4.0.4 to 4.1.2
- Less steps (for user)
  - Press one button

[Xing2014]
Android Updating

- Download upgrading image through OTA (Over the Air)
- Reboot to recovery mode
- Replace some system files, such as bootloader, Package Manager Service (PMS), and APKs under /system directory
- Reboot to the new OS
- Update other components
What PMS does when upgrading Android OS

• Install or reinstall all system apps under `/system`, and then 3rd-party apps under `/data/app`
• Register an app’s permissions, shared UID, activities, intent filters, ……
• Decide what to do when a conflict occurs (duplicated attr. or prop.)
  • Build a structure `mSettings` for existing apps, and include:
    • `mPackages`
    • `mUserIds`
    • `mSharedUsers`
    • `mPermissions`
    • etc.
  • Check the `mSettings` when installing a new system package
• If having conflicts, decide case by case.
Pileup Exploits

- Assume that an attacker has a malicious app installed through google play or 3rd part market
- App requests permission not available in current version
- Possible exploits:
  - Permission Harvesting and Preempting
  - Shared UID Grabbing
  - Data Contamination
  - Denial of Services
Pileup Exploits – Permission Harvesting and Preempting

Before updating
- Installed malware
  - Claimed for permissions of new OS or apps
  - Old OS can not recognize these permissions
  - No report

Updating to new OS
- Installing System apps
  - Declare new permissions
- Installing 3rd-party apps
  - Reinstalling the old malware
    - Automatically grant the permissions
  - Without user’s consent

These permissions are restricted below “dangerous” level

[Shahrivar 1393]

ISCISC2014
Reviewing Attacks on Android

[Xing2014]
Pileup Exploits – Permission Harvesting and Preempting

Before updating:
- Installed malware
  - Declared and defined the permissions the same as those of new system apps
  - Old OS lets the malware declare them
  - Without user’s intervention

Updating to new OS:
- Building mSettings for old apps
- Installing System apps
  - Check
  - Skip if conflicts
- Declare new permissions
- Reinstalling the old malware
  - Automatically declare and grant permissions
  - Without user’s consent

“signature” - OK
“system” - OK
Lower to “normal” – OK
Change the description – OK

Shahrivar 1393
ISCISC2014
Reviewing Attacks on Android
[Xing2014]
Questions we investigate

- People at google are smart, latest security measures are being used, could there be any problems?

- Wouldn’t upgrading my android definitely improve my security?

- No microphone permission, so would there be any risk of eavesdropping?

- I have no private info on my smartphone, would there be any privacy risks?
MICROPHONE ACCESS

**REQUIRES PERMISSIONS**

**DOES NOT REQUIRE PERMISSIONS**

GYROSCOPE ACCESS

Shahrivar 1393

ISCISC2014

Reviewing Attacks on Android

[Michalevsky2014]
Gyrosopes

STM Microelectronics
Samsung Galaxy

InvenSense
Google Nexus

Shahrivar 1393
Reviewing Attacks on Android

ISCISC2014

[Michalevsky2014]
Gyroscopes are susceptible to sound
Gyroscopes are (lousy, but still) microphones

- Hardware sampling frequency:
  - InvenSense: up to 8000 Hz
  - STM Microelectronics: 800 Hz
- Software sampling frequency:
  - Android: 200 Hz
  - iOS: 100 Hz
- Very low SNR (Signal-to-Noise Ratio)
  - Acoustic sensitivity threshold: ~70 dB
    - Comparable to a loud conversation.
How do we look into higher frequencies?

- Speech range:
  - Adult male 85-180 HZ
  - Adult female 165 - 255 HZ

- Make use of aliasing
Accuracy

• Gender identification
  • Nexus 4 84%
  • Galaxy S III 82%
  • Random guess probability 50%

• Speaker identification
  • Random guess probability is 20% for one gender and 10% for a mixed set
  • Isolated word recognition (speaker dependent)
    • 65% (random guess probability 9%)
What if OS is patched?

- Hardware sampling frequency:
  - InvenSense: up to 8000 Hz
  - STM Microelectronics: 800 Hz
- Software sampling frequency:
  - Android: 200 Hz
  - iOS: 100 Hz
Questions we investigate

• People at google are smart, latest security measures are being used, could there be any problems?

• Wouldn’t upgrading my android definitely improve my security?

• No microphone permission, so would there be any risk of eavesdropping?

• I have no private info on my smartphone, would there be any privacy risks?
Android Public Resources

Application Framework

Public APIs
(Audio Usage, CPU Usage, Running application list)

Linux Kernel

Public files
(procfs, sysfs)

[Zhou2013]
Location inference

- /proc/net/arp contains Address Resolution Protocol (ARP) information!
- /proc/net/arp contains BSSID (i.e. MAC address of the wireless interface) of the access point phone is connected to
  - ARP information wasn’t considered sensitive in original Linux design
- Databases such as Navizon collect MAC to GPS locations
- zero permission app could collect MAC information from /proc/net/arp

[Zhou2013]
Transmitting out information

- Using URI ACTION_VIEW an app could transmit a GET request through the browser
  - A payload could be transmitted with the GET request
- User will observe this on the screen
  - When screen is off, the browser will be “paused”
- Therefore, an app will continuously check the lcd status indicator (/lcd_power)
  - When indicator becomes zero, the screen dims out
  - the app will submit the request to the browser at that point
  - after transmission, it redirect the browser to google to cover its tracks
Driving route interference

- Speaker status (i.e. On/Off), could be check by AudioManager.isMusicActive

![Diagram showing ON and OFF states with timings]

- Segment 1: Turn left onto N Goodwin Ave
- Segment 2: Head west on W Clark St toward N Busey Ave

[Zhou2013]
Driving route interference

- Check if GPS navigation app is running
- Collect speaker on/off periods
- Create Fingerprint
  - 10 | 30 | 60 | 10 | 40
- Find the matching fingerprint in the database
Questions we investigated

- People at google are smart, latest security measures are being used, could there be any problems?

- Wouldn’t upgrading my android definitely improve my security?

- No microphone permission, so would there be any risk of eavesdropping?

- I have no private info on my smartphone, would there be any privacy risks?
Acknowledgments/References