CS527 Software Security

Mobile Security

Mathias Payer

Purdue University, Spring 2018

Android statistics

- 1.4 billion users (1)
- Android generates $31 billion revenue (2)
- 4,000+ different devices (1)
- Android OS with highest crash rate: Gingerbread (3)
- Percentage of Android devices running Marshmallow: 1.2% (1)
- Percentage of Android devices running Lollipop: 34.1% (1)

(1 DMR stats, 9/29/15, 2 Bloomberg, 1/21/16, 3 Greenbot, 3/27/14)

Android history

- 2005: Google buys Android
- 2007: Initial SDK released
- 2008: First devices announced
- 2009: Cupcake (1.5, 3), Donut (1.6, 4), Eclair (2.0, 5)
- 2010: Froyo (2.2, 8), Gingerbread (2.3, 9)
- 2011: Honeycomb (3.0, 11), Ice Cream Sandwich (4.0.1, 14)
- 2012: Jelly Been (4.1.1, 16)
- 2013: KitKat (4.4, 19)
- 2014: Lollipop (5.0, 21)
- 2015: Marshmallow (6.0, 23)
- 2016: Nougat (7.0, 24-25)
- 2017: Oreo (8.0 - 8.1, 26-27)
- 2018: Android P?

Android security goals

- Isolate individual applications
- Protect system resources from applications
- Vet applications "online"
- Protect data of *the* user (until 5.0 single user)

Android security architecture

- Applications are carefully vetted server-side and only approved applications can be installed from the "market"
- Each application runs in a Java-like sandbox and is restricted to user-granted permissions and can therefore only communicate through well-defined API channels with other applications
- The system is hardened against local user (app-based) attacks.

Android app security

- Each apps runs in its own secure context/sandbox
- Interactions between apps are restricted through the API
- Each app has an associated policy, encoding the permissions
- Apps are signed by the developer, vetted, and installed from a central market

Android system security

- Hardened Linux kernel protects applications
- Each application runs as its independent user
- Stringent permissions on file systems (except sdcard)
- SELinux to apply access control policies on processes
- File system encryption
- User-space: stack canaries, integer overflow mitigation, double free protection (through allocator), fortify source, NX, `mmap_min_addr`, ASLR, PIE, relro, immediate binding
- Each release: security updates, patches, toolchain updates, tigther security defaults

Android permissions

Complex permission system on a per-app basis.

- Camera
- Location
- Bluetooth
- Telephony
- SMS/MMS functionality
- Network/data connections

Android intents

Google's idea of mobile IPC

> *An Intent is a simple message object that represents an "intention" to do something. For example, if your application wants to display a web page, it expresses its "Intent" to view the URL by creating an Intent instance and handing it off to the system. The system locates some other piece of code (in this case, the Browser) that knows how to handle that Intent, and runs it. Intents can also be used to broadcast interesting events (such as a notification) system-wide. (From Google's Android website.)*

(Some) Android attack vectors

- Unauthorized intent receipt: attacker creates an intent filter, receives other apps' intents that contain privileged information (e.g., intent filter for web service intercepts online payment process)
- Intent spoofing: attacker sends a malicious intent to an intent processor (e.g., flooding the network with malicious messages)
- Insecure storage: there are no access restrictions on the SD card (why?), an attacker may read/write any data on the SD card
- Insecure communication: run Wireshark to intercept traffic
- Overprivileged app: confused deputy, bugs in application can be leveraged by attacker to gain privileges
- Unsafe privileges: there's only one Bluetooth privilege, privileges are per *app*, not *per connected device*

Summary and conclusion

- Android security evolved over time
- Android systems hardened against exploits
- Developers sign apps which identifies them (comes with a cost)
- Applications are vetted centrally and installed from the market