CS590-SWS/527 Software Security

Mobile Security

Asst. Prof. Mathias Payer

Department of Computer Science
Purdue University

TA: Kyriakos Ispoglou

`https://nebelwelt.net/teaching/16-527-SoftSec/`

Spring 2016

Table of Contents

Android statistics

- 1.4 billion users (DMR stats, 9/29/15)
- Android generates $31 billion revenue (Bloomberg, 1/21/16)
- 4,000+ different devices (DMR stats, 9/29/15)
- Android OS with highest crash rate: Gingerbread (Greenbot, 3/27/14)
- Percentage of Android devices running Marshmallow: 1.2% (DMR stats, 2/2/16)
- Percentage of Android devices running Lollipop: 34.1% (DMR stats, 2/2/16)

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)

Android security goals

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

Android security architecture

Architecture

The security of Android devices depends on a three tiered architecture: (i) applications are carefully vetted server-side and only approved applications can be installed from the "market", (ii) 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, and (iii) the system is hardened against local user (app-based) attacks.

Android security fundamentals

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

Android kernel 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 (since 3.0)

Android user-space security for memory safety

- Stack canaries, integer overflow protection, double free protection (using memory allocator), and calloc instead of malloc (1.5)
- Format string protection, NX, and mmap_min_addr (2.3)
- ASLR (4.0), PIE, read only relocations, immediate binding, avoid kernel address leaks (4.1)
- fortify source, init script hardening, certificate pinning, app verification (4.2)
- SELinux, no suid, ADB authentication, capability bounding, more fortification (4.3)
- Stronger SELinux, stronger fortification (4.4)
- More SELinux/fortification, everything PIE, disk encryption (5.0)
- Verified boot, hardware security, system hardening (6.0)
- Each release: security updates, patches, and toolchain updates

Android permissions

- Android has many types of complex permissions.
- Camera functions
- Location data (GPS)
- Bluetooth functions
- Telephony functions
- SMS/MMS functionality
- Network/data connections

Android intents

Intent

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.[a]

[a] This is Google’s definition of an intent.

Android attack vectors

- Unauthorized intent receipt: attacker creates an intent filter and receives intents from other apps that contain privileged information (e.g., intent filter for web service that 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: an attacker may run wireshark to intercept traffic
- Overprivileged app: a bug in an app allows an attacker to leverage these privileges.

Table of Contents

1 Android Security

2 Summary and conclusion

Summary

- Android security evolved over time
- Android systems are hardened against exploits
- Developers sign their app and required permissions
- Applications are vetted centrally and installed from the market

Questions?

?