CSE 484 / CSE M 584: Computer Security and Privacy

Web Privacy [finish]
Mobile Platform Security [start]

Spring 2017

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
• **Today:** finish web privacy, start mobile security

• **Friday:**
  – Lab #2 due (8pm)
  – **Guest lecture:** Jon McClintock, Amazon Security

• **Monday:**
  – **Guest lecture:** David Aucsmith
  – **Former senior director of Microsoft's Institute for Advanced Technology in Governments** (among many other cool things)
How has this changed over time?

• The web has existed for a while now...
  – What about tracking before 2011? (our first study)
  – What about tracking before 2009? (first academic study)

• Solution: time travel!

[USENIX Security ’16]
The Wayback Machine to the Rescue

Time travel for web tracking: http://trackingexcavator.cs.washington.edu
1996-2016: More & More Tracking

• More trackers of more types
1996-2016: More & More Tracking

- More trackers of more types, more per site
1996-2016: More & More Tracking

- More trackers of more types, more per site, more coverage

Rise And Fall of Historical Champion Trackers

Coverage (of Top 500)

- come.to
- go.com
- v3.com
- doubleclick.net
- allyes.com
- 2o7.net
- google-analytics.com
- google.com
- quantserve.com
- scorecardresearch.com
- gstatic.com

Years

Defenses to Reduce Tracking

• Do Not Track proposal?

Send a ‘Do Not Track’ request with your browsing traffic

Do Not Track is not a technical defense: trackers must honor the request.
Defenses to Reduce Tracking

• Do Not Track proposal?
• Private browsing mode?

Private browsing mode protects against local, not network, attackers.

You've gone incognito. Pages you view in incognito tabs won't stick around in your browser's history, cookie store, or search history after you've closed all of your incognito tabs. Any files you download or bookmarks you create will be kept.

However, you aren't invisible. Going incognito doesn't hide your browsing from your employer, your internet service provider, or the websites you visit.
Defenses to Reduce Tracking

- Do Not Track proposal?
- Private browsing mode?
- Third-party cookie blocking?
Quirks of 3rd Party Cookie Blocking

In some browsers, this option means third-party cookies cannot be set, but they CAN be sent.

So if a third-party cookie is somehow set, it can be used.

How to get a cookie set?
One way: be a first party.

etc.
Defenses to Reduce Tracking

- Do Not Track header?
- Private browsing mode?
- Third-party cookie blocking?
- Browser add-ons?

Often rely on blacklists, which may be incomplete.

“uses algorithmic methods to decide what is and isn't tracking”; incorporates code from UW for handling social media buttons.
MOBILE PLATFORM SECURITY
Roadmap

• Mobile malware
• Mobile platforms vs. traditional platforms
• Deep dive into Android
  – Continued next Wednesday
  – Background for Lab #3
Questions: Mobile Malware

Q1: How might malware authors get malware onto phones?

Q2: What are some goals that mobile device malware authors might have?

Q3: What technical things might malware authors do?
Smartphone (In)Security

Users accidentally install malicious applications.

Over 60% of Android malware steals your money via premium SMS, hides in fake forms of popular apps

By Emil Protalinski, Friday, 5 Oct ‘12, 05:50pm
Smartphone (In)Security

Even legitimate applications exhibit questionable behavior.

Hornyack et al.: 43 of 110 Android applications sent location or phone ID to third-party advertising/analytics servers.

Android flashlight app tracks users via GPS, FTC says hold on
Malware in the Wild

Android malware is growing.

[Zhou et al.]
Mobile Malware Attack Vectors

• Unique to phones:
  – Premium SMS messages
  – Identify location
  – Record phone calls
  – Log SMS

• Similar to desktop/PCs:
  – Connects to botmasters
  – Steal data
  – Phishing
  – Malvertising
Mobile Malware Examples

• **DroidDream** (Android)
  – Over 58 apps uploaded to Google app market
  – Conducts data theft; send credentials to attackers

• **Zitmo** (Symbian, BlackBerry, Windows, Android)
  – Poses as mobile banking application
  – Captures info from SMS – steal banking 2\textsuperscript{nd} factors
  – Works with Zeus botnet

• **Ikee** (iOS)
  – Worm capabilities (targeted default ssh password)
  – Worked only on jailbroken phones with ssh installed
Mobile Malware Examples

“ikee is never going to give you up”
## (Android) Malware in the Wild

### What does it do?

<table>
<thead>
<tr>
<th></th>
<th>Root Exploit</th>
<th>Remote Control</th>
<th>Financial Charges</th>
<th>Information Stealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Net</td>
<td>Phone Call</td>
<td>Block SMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMS</td>
<td>SMS</td>
<td>SMS</td>
</tr>
<tr>
<td># Families</td>
<td>20</td>
<td>27</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Samples</td>
<td>1204</td>
<td>1171</td>
<td>1</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>571</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>315</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>138</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>563</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>43</td>
</tr>
</tbody>
</table>

Why all these problems with mobile malware?
Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Linux) design:
1. There may be multiple users who don’t trust each other.
2. Once an application is installed, it’s (more or less) trusted.
Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Linux) design:
1. There may be multiple users who don’t trust each other.
2. Once an application is installed, it’s (more or less) trusted.

FranziBook:Desktop franzi$ whoami
franzi

FranziBook:Desktop franzi$ id
uid=501(franzi) gid=20(staff) groups=20(staff),401(com.apple.sharepoint.group.1),502(access_bpf),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),33(_appstore),100(_lpoperator),204(_developer),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh)

FranziBook:Desktop franzi$ ls -l hello.txt
-rw-r--r-- 1 franzi staff 0 Nov 29 10:08 hello.txt

FranziBook:Desktop franzi$ chmod 700 hello.txt

FranziBook:Desktop franzi$ ls -l hello.txt
-rwx------ 1 franzi staff 0 Nov 29 10:08 hello.txt
Background: Before Mobile Platforms

Assumptions in traditional OS (e.g., Linux) design:
1. There may be multiple users who don’t trust each other.
2. Once an application is installed, it’s (more or less) trusted.

Apps can do anything the UID they’re running under can do.
What’s Different about Mobile Platforms?

• Applications are isolated
  – Each runs in a separate execution context
  – No default access to file system, devices, etc.
  – Different than traditional OSes where multiple applications run with the same user permissions!

• App Store: approval process for applications
  – Market: Vendor controlled/Open
  – App signing: Vendor-issued/self-signed
  – User approval of permissions
More Details: Android

- Based on Linux
- Application sandboxes
  - Applications run as separate UIDs, in separate processes.
  - Memory corruption errors only lead to arbitrary code execution in the context of the particular application, not complete system compromise!
  - (Can still escape sandbox – but must compromise Linux kernel to do so.) ← allows rooting

Since 5.0: ART (Android runtime) replaces Dalvik VM to run apps natively

[Enck et al.]
Android Applications

- **Activities** provide user interfaces.
- **Services** run in the background.
- **BroadcastReceivers** receive messages sent to multiple applications (e.g., BOOT_COMPLETED).
- **ContentProviders** are databases addressable by their application-defined URIs.

- **AndroidManifest.xml**
  - Specifies application components
  - Specifies required permissions
Rooting and Jailbreaking

• Allows user to **run applications with root privileges**
  – e.g., modify/delete system files, app management, CPU management, network management, etc.

• Done by **exploiting vulnerability in firmware** to install `su` binary.

• Double-edged sword...

• Note: iOS is more restrictive than Android
  – Doesn’t allow “side-loading” apps, etc.