CSE 484 / CSE M 584: Computer Security and Privacy

Mobile Platform Security [finish]

Fall 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 ...
Admin

• Project checkpoint #2 due tonight
• Keep letting us know of any fuzzing issues with HW3
  – Double check that you followed the instructions
  – Make sure to put things in C: instead of D:
  – It’s possible to run into issues on the MS side, so let us know and we’ll loop them in if needed (in the meantime you can relax 😊)
This Week: Mac OS X High Sierra Issue

• Given physical access, if root password not set, can login (creating a root account?) without a password
• Manual fix: set root password

Security Update 2017-001
Released November 29, 2017

Directory Utility

Available for: macOS High Sierra 10.13 and macOS High Sierra 10.13.1
Not impacted: macOS Sierra 10.12.6 and earlier

Impact: An attacker may be able to bypass administrator authentication without supplying the administrator’s password

Description: A logic error existed in the validation of credentials. This was addressed with improved credential validation.

CVE-2017-13872

Entry updated November 29, 2017
Back to Mobile Security: Challenges with Isolated Apps

So mobile platforms isolate applications for security, but...

1. **Permissions**: How can applications access sensitive resources?
2. **Communication**: How can applications communicate with each other?
State of the Art

Prompts (time-of-use)

Manifests (install-time)
Are Manifests Usable?

Do users pay attention to permissions?

24 observed installations

- Looked at permissions: 42%
- Didn’t look, but aware: 42%
- Unaware of permissions: 17%

... but 88% of users looked at reviews.
Do users understand the warnings?

<table>
<thead>
<tr>
<th>Permission</th>
<th>n</th>
<th>Correct Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Choice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ_CALENDAR</td>
<td>101</td>
<td>46</td>
</tr>
<tr>
<td>CHANGE_NETWORK_STATE</td>
<td>66</td>
<td>26</td>
</tr>
<tr>
<td>READ_SMS₁</td>
<td>77</td>
<td>24</td>
</tr>
<tr>
<td>CALL_PHONE</td>
<td>83</td>
<td>16</td>
</tr>
<tr>
<td>WAKE_LOCK</td>
<td>81</td>
<td>27</td>
</tr>
<tr>
<td>WRITE_EXTERNAL_STORAGE</td>
<td>92</td>
<td>14</td>
</tr>
<tr>
<td>READ_CONTACTS</td>
<td>86</td>
<td>11</td>
</tr>
<tr>
<td>INTERNET</td>
<td>109</td>
<td>12</td>
</tr>
<tr>
<td>READ_PHONE_STATE</td>
<td>85</td>
<td>4</td>
</tr>
<tr>
<td>READ_SMS₂</td>
<td>54</td>
<td>12</td>
</tr>
<tr>
<td>2 Choices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Choices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAMERA</td>
<td>72</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4: The number of people who correctly answered a question. Questions are grouped by the number of correct choices. n is the number of respondents. (Internet Survey, n = 302)
Are Manifests Usable?

Do users act on permission information?

“Have you ever not installed an app because of permissions?”

- Yes: 72%
- No: 20%
- Probably: 8%

25 interview responses
Android 6.0: Prompts!

- **First-use prompts** for sensitive permission (like iOS).
- **Big change!** Now app developers need to check for permissions or catch exceptions.
Over-Permissioning

- Android permissions are badly documented.
- Researchers have mapped APIs $\rightarrow$ permissions.

www.android-permissions.org (Felt et al.), http://pscout.csl.toronto.edu (Au et al.)
Improving Permissions: AppFence

Today, ultimatums give app developers an unfair edge in obtaining permissions.

AppFence can enable new interfaces that give users control over the use of their info.

- Network communication
  - full Internet access
- Your personal information
  - read contact data
- Phone calls
  - read phone state and identity

I'd rather not share all that information just to try this app, but it looks like I have no choice.

I'll start by giving out only the information I think this app actually needs.
Improving Permissions: User-Driven Access Control

Let this application access my location now.

Insight:
A user’s natural UI actions within an application implicitly carry permission-granting semantics.
Let this application access my location now.

**Our study shows:**
Many users already believe (52% of 186) – and/or desire (68%) – that resource access follows the user-driven access control model.
New OS Primitive: Access Control Gadgets (ACGs)

Approach: Make resource-related UI elements first-class operating system objects (access control gadgets).

• To receive resource access, applications must embed a system-provided ACG.
• ACGs allow the OS to capture the user’s permission granting intent in application-agnostic way.
(2) Inter-Process Communication

• Primary mechanism in Android: **Intents**
  – Sent between application components
    • e.g., with `startActivity(intent)`

  – **Explicit**: specify component name
    • e.g., `com.example.testApp.MainActivity`

  – **Implicit**: specify action (e.g., `ACTION_VIEW`) and/or data (URI and MIME type)
    • Apps specify **Intent Filters** for their components.
Unauthorized Intent Receipt

Attack #1: Eavesdropping / Broadcast Theft

com.example.goodapp1

com.example.goodapp2

com.example.badapp
Unauthorized Intent Receipt

Attack #2: Service/Activity Hijacking

“Caution: To ensure that your app is secure, always use an explicit intent when starting a Servier. Using an implicit intent to start a service is a security hazard because you can't be certain what service will respond to the intent, and the user can't see which service starts.”

Figure 1. How an implicit intent is delivered through the system to start another activity: [1] Activity A creates an Intent with an action description and passes it to startActivity(). [2] The Android System searches all apps for an intent filter that matches the intent. When a match is found, [3] the system starts the matching activity (Activity B) by invoking its onCreate() method and passing it the Intent.
Unauthorized Intent Receipt

• **Attack #1:** Eavesdropping / Broadcast Thefts
  – Implicit intents make intra-app messages public.

• **Attack #2:** Activity Hijacking
  – May not always work:

• **Attack #3:** Service Hijacking
  – Android picks one at random upon conflict!
Intent Spoofing

• **Attack #1:** General intent spoofing
  – Receiving implicit intents makes component public.
  – Allows data injection.

• **Attack #2:** System intent spoofing
  – Can’t directly spoof, but victim apps often don’t check specific “action” in intent.

```
com.example.goodapp1

com.example.goodapp2

com.example.badapp
```
Permission Re-Delegation

• An application without a permission gains additional privileges through another application.

• **Demo video**

• Settings application is **deputy**: has permissions, and accidentally exposes APIs that use those permissions.
More on Android...
Incomplete Isolation

Embedded UIs and libraries always run with the host application’s permissions! (No same-origin policy here…)

[Shekhar et al.]
Android Application Signing

• Apps are signed
  – Often with self-signed certificates
  – Signed application certificate defines which user ID is associated with which applications
  – Different apps run under different UIDs

• Shared UID feature
  – Shared Application Sandbox possible, where two or more apps signed with same developer key can declare a shared UID in their manifest
Shared UIDs

- App 1: Requests GPS / camera access
- App 2: Requests Network capabilities

Generally:
- First app can’t exfiltrate information
- Second app can’t exfiltrate anything interesting

With Shared UIDs (signed with same private key):
- Permissions are a superset of permissions for each app
- App 1 can now exfiltrate; App 2 can now access GPS / camera
File Permissions

• Files written by one application cannot be read by other applications
  – Previously, this wasn’t true for files stored on the SD card (world readable!) – Android cracked down on this

• It is possible to do full file system encryption
  – Key = Password/PIN combined with salt, hashed
Memory Management

- Address Space Layout Randomization to randomize addresses on stack
- Hardware-based No eXecute (NX) to prevent code execution on stack/heap
- Stack guard derivative
- Some defenses against double free bugs (based on OpenBSD’s dmalloc() function)
- etc.

[See http://source.android.com/tech/security/index.html]
Android Fragmentation

• Many different variants of Android (unlike iOS)
  – Motorola, HTC, Samsung, ...
• Less secure ecosystem
  – Inconsistent or incorrect implementations
  – Slow to propagate kernel updates and new versions

[https://developer.android.com/about/dashboards/index.html]
What about iOS?

- Apps are sandboxed
- Encrypted user data
  - See recent news...
- App Store review process is (maybe) stricter
  - But not infallible: e.g., see Wang et al. “Jekyll on iOS: When Benign Apps Become Evil” (USENIX Security 2013)

- No “sideloading” apps
  - Unless you jailbreak

Figure 1: High Level Intuition
CSE 484 / CSE M 584: Computer Security and Privacy

Usable Security [start]

Fall 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 ...
Poor Usability Causes Problems
Importance in Security

• Why is usability important?
  – People are the critical element of any computer system
    • People are the real reason computers exist in the first place
  – Even if it is possible for a system to protect against an adversary, people may use the system in other, less secure ways
Usable Security Roadmap

• 2 case studies
  – Phishing
  – SSL warnings

• Step back: root causes of usability problems, and how to address
Case Study #1: Phishing

• Design question: How do you help users avoid falling for phishing sites?
A Typical Phishing Page

Weird URL
http instead of https
Safe to Type Your Password?
Safe to Type Your Password?

![Phishing website](image-url)
Safe to Type Your Password?
Safe to Type Your Password?

“Picture-in-picture attacks”

Trained users are more likely to fall victim to this!
Experiments at Indiana University

• Reconstructed the social network by crawling sites like Facebook, MySpace, LinkedIn and Friendster
• Sent 921 Indiana University students a spoofed email that appeared to come from their friend
• Email redirected to a spoofed site inviting the user to enter his/her secure university credentials
  – Domain name clearly distinct from indiana.edu
• 72% of students entered their real credentials into the spoofed site
More Details

• Control group: 15 of 94 (16%) entered personal information
• Social group: 349 of 487 (72%) entered personal information

• 70% of responses within first 12 hours
• Adversary wins by gaining users’ trust

• Also: If a site looks “professional”, people likely to believe that it is legitimate
Phishing Warnings

Passive (IE)

Active (Firefox)

Active (IE)
Are Phishing Warnings Effective?

• CMU study of 60 users
• Asked to make eBay and Amazon purchases
• All were sent phishing messages in addition to the real purchase confirmations
• Goal: compare active and passive warnings
Active vs. Passive Warnings

- Active warnings significantly more effective
  - Passive (IE): 100% clicked, 90% phished
  - Active (IE): 95% clicked, 45% phished
  - Active (Firefox): 100% clicked, 0% phished
User Response to Warnings

• Some fail to notice warnings entirely
  – Passive warning takes a couple of seconds to appear; if user starts typing, his keystrokes dismiss the warning
• Some saw the warning, closed the window, went back to email, clicked links again, were presented with the same warnings… repeated 4-5 times
  – Conclusion: “website is not working”
  – Users never bothered to read the warnings, but were still prevented from visiting the phishing site
  – Active warnings work!
Why Do Users Ignore Warnings?

• Don’t trust the warning
  – “Since it gave me the option of still proceeding to the website, I figured it couldn’t be that bad”

• Ignore warning because it’s familiar (IE users)
  – “Oh, I always ignore those”
  – “Looked like warnings I see at work which I know to ignore”
  – “I thought that the warnings were some usual ones displayed by IE”
  – “My own PC constantly bombards me with similar messages”
Case Study #2: Browser SSL Warnings

- **Design question 1:** How to indicate encrypted connections to users?
- **Design question 2:** How to alert the user if a site’s SSL certificate is untrusted?
The Lock Icon

- Goal: identify secure connection
  - SSL/TLS is being used between client and server to protect against active network attacker
- Lock icon should only be shown when the page is secure against network attacker
  - Semantics subtle and not widely understood by users
  - Whose certificate is it??
  - Problem in user interface design
Will You Notice?

Clever favicon inserted by network attacker
Do These Indicators Help?

• “The Emperor’s New Security Indicators”

<table>
<thead>
<tr>
<th>Score</th>
<th>First chose not to enter password...</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>1 ∪ 2</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>upon noticing HTTPS absent</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>1</td>
<td>after site-authentication image removed</td>
<td>0%</td>
<td>0%</td>
<td>2%</td>
<td>0%</td>
<td>2%</td>
</tr>
<tr>
<td>2</td>
<td>after warning page</td>
<td>8%</td>
<td>5%</td>
<td>12%</td>
<td>13%</td>
<td>25%</td>
</tr>
<tr>
<td>3</td>
<td>never (always logged in)</td>
<td>10%</td>
<td>12%</td>
<td>8%</td>
<td>22%</td>
<td>30%</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>18%</td>
<td>17%</td>
<td>22%</td>
<td>35%</td>
<td>57%</td>
</tr>
</tbody>
</table>

Users don’t notice the absence of indicators!
Aside (re: Phishing): Site Authentication Image (SiteKey)

If you don’t recognize your personalized SiteKey, don’t enter your Passcode.
Latest Design in Chrome

Secure https://mail.google.com/mail/u/0/#inbox

Not Secure http-password.badssl.com

⚠️ This page includes a password or credit card input in a non-secure context. A warning has been added to the URL bar. For more information, see https://goo.gl/zmWq3m.
Firefox vs. Chrome Warning

33% vs. 70% clickthrough rate
### Experimenting w/ Warning Design

<table>
<thead>
<tr>
<th>#</th>
<th>Condition</th>
<th>CTR</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Control (default Chrome warning)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Chrome warning with policeman</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Chrome warning with criminal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Chrome warning with traffic light</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Mock Firefox</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Mock Firefox, no image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Mock Firefox with corporate styling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Click-through rates and sample size for conditions.
Experimenting w/ Warning Design

<table>
<thead>
<tr>
<th>#</th>
<th>Condition</th>
<th>CTR</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Control (default Chrome warning)</td>
<td>67.9%</td>
<td>17,479</td>
</tr>
<tr>
<td>2</td>
<td>Chrome warning with policeman</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Chrome warning with criminal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Chrome warning with traffic light</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Mock Firefox</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Mock Firefox, no image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Mock Firefox with corporate styling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Click-through rates and sample size for conditions.

Figure 1. The default Chrome SSL warning (Condition 1).
Experimenting w/ Warning Design

<table>
<thead>
<tr>
<th>#</th>
<th>Condition</th>
<th>CTR</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Control (default Chrome warning)</td>
<td>67.9%</td>
<td>17,479</td>
</tr>
<tr>
<td>2</td>
<td>Chrome warning with policeman</td>
<td>68.9%</td>
<td>17,977</td>
</tr>
<tr>
<td>3</td>
<td>Chrome warning with criminal</td>
<td>66.5%</td>
<td>18,049</td>
</tr>
<tr>
<td>4</td>
<td>Chrome warning with traffic light</td>
<td>68.8%</td>
<td>18,084</td>
</tr>
<tr>
<td>5</td>
<td>Mock Firefox</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Mock Firefox, no image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Mock Firefox with corporate styling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Click-through rates and sample size for conditions.

Figure 1. The default Chrome SSL warning (Condition 1).

Figure 4. The three images used in Conditions 2-4.
Experimenting w/ Warning Design

<table>
<thead>
<tr>
<th>#</th>
<th>Condition</th>
<th>CTR</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Control (default Chrome warning)</td>
<td>67.9%</td>
<td>17,479</td>
</tr>
<tr>
<td>2</td>
<td>Chrome warning with policeman</td>
<td>68.9%</td>
<td>17,977</td>
</tr>
<tr>
<td>3</td>
<td>Chrome warning with criminal</td>
<td>66.5%</td>
<td>18,049</td>
</tr>
<tr>
<td>4</td>
<td>Chrome warning with traffic light</td>
<td>68.8%</td>
<td>18,084</td>
</tr>
<tr>
<td>5</td>
<td>Mock Firefox</td>
<td>56.1%</td>
<td>20,023</td>
</tr>
<tr>
<td>6</td>
<td>Mock Firefox, no image</td>
<td>55.9%</td>
<td>19,297</td>
</tr>
<tr>
<td>7</td>
<td>Mock Firefox with corporate styling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Click-through rates and sample size for conditions.

![Mock Firefox SSL warning](image)

Figure 2. The mock Firefox SSL warning (Condition 5).
Experimenting w/ Warning Design

<table>
<thead>
<tr>
<th>#</th>
<th>Condition</th>
<th>CTR</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Control (default Chrome warning)</td>
<td>67.9%</td>
<td>17,479</td>
</tr>
<tr>
<td>2</td>
<td>Chrome warning with policeman</td>
<td>68.9%</td>
<td>17,977</td>
</tr>
<tr>
<td>3</td>
<td>Chrome warning with criminal</td>
<td>66.5%</td>
<td>18,049</td>
</tr>
<tr>
<td>4</td>
<td>Chrome warning with traffic light</td>
<td>68.8%</td>
<td>18,084</td>
</tr>
<tr>
<td>5</td>
<td>Mock Firefox</td>
<td>56.1%</td>
<td>20,023</td>
</tr>
<tr>
<td>6</td>
<td>Mock Firefox, no image</td>
<td>55.9%</td>
<td>19,297</td>
</tr>
<tr>
<td>7</td>
<td>Mock Firefox with corporate styling</td>
<td>55.8%</td>
<td>19,845</td>
</tr>
</tbody>
</table>

Table 1. Click-through rates and sample size for conditions.

![Firefox SSL warning with Google styling](image)

Figure 3. The Firefox SSL warning with Google styling (Condition 7).
Opinionated Design Helps!

The site's security certificate is not trusted!

You attempted to reach 192.168.17.129, but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications.

You should not proceed, especially if you have never seen this warning before for this site.

<table>
<thead>
<tr>
<th>Adherence</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.9%</td>
<td>4,551</td>
</tr>
</tbody>
</table>
Opinionated Design Helps!

Adherence | N
---|---
30.9% | 4,551
32.1% | 4,075
58.3% | 4,644
Challenge: Meaningful Warnings

- a248.e.akamai.net
- Client missing root certificate
- Captive portal
- Expired certificate
- Client clock wrong
- Anti-virus software
- School or employer
- Gov’t content filter
- ISP adding advertisements
- Malware
- State attacks
- Certificate mis-issuance
- REAL ATTACK
Stepping Back: Root Causes?

- Computer systems are complex; users lack intuition
- Users in charge of managing own devices
  - Unlike other complex systems, like healthcare or cars.
- Hard to gauge risks
  - “It won’t happen to me!”
- Annoying, awkward, difficult
- Social issues
  - Send encrypted emails about lunch?...
How to Improve?

- Security education and training
- Help users build accurate mental models
- Make security invisible
- Make security the least-resistance path
- ...?