Web Security: UI-based attacks

CS 161: Computer Security
Prof. Raluca Ada Popa

February 17, 2016

Contains new slides, slides from past CS 161 offerings and slides from Dan Boneh
Announcements

• Wednesday, Feb 24,
  – 8-9:30pm (in 155 Dwinelle)

• Homework 2 (due Feb 22)
February 16, 2016

A Message to Our Customers

The United States government has demanded that Apple take an unprecedented step which threatens the security of our customers. We oppose this order, which has implications far beyond the legal case at hand.
we have done everything that is both within our power and within the law to help them. But now the U.S. government has asked us for something we simply do not have, and something we consider too dangerous to create. They have asked us to build a backdoor to the iPhone.

Specifically, the FBI wants us to make a new version of the iPhone operating system, circumventing several important security features, and install it on an iPhone recovered during the investigation. In the wrong hands, this software — which does not exist today — would have the potential to unlock any iPhone in someone’s physical possession.

The FBI may use different words to describe this tool, but make no mistake: Building a version of iOS that bypasses security in this way would undeniably create a backdoor. And while the government may argue that its use would be limited to this case, there is no way to guarantee such control.
Clickjacking attacks

• Exploitation where a user’s mouse click is used in a way that was not intended by the user
Talk to your partner

- How can a user’s click be used in a way different than intended?
Simple example

```html
Go to Google</a>
```

What does it do?
- Opens a window to the attacker site

Why include `href` to Google?
- Browser status bar shows URL when hovering over as a means of protection
Frames - background

- A frame is used to embed another document within the current HTML document
- Any site can frame another site
- The `<iframe>` tag specifies an inline frame
Example

**HTML page**

```html
<iframe src="http://www.google.com/">
  This text is ignored by most browsers.
</iframe>
```

**UI rendering**

framed page/inner page

framing page/outer page
Frames

• Outer page can set frame width, height
• But then, only framed site can draw in its own rectangle

• Modularity
  – Brings together code from different sources
What happens in this case?

Funny cats website

JavaScript
Frames: same-origin policy

- Frame inherits origin of its URL
- Same-origin policy: if frame and outer page have different origins, they cannot access each other
  - In particular, malicious JS on outer page cannot access resources of inner page
How to bypass same-origin policy for frames?

Clickjacking
Clickjacking using frames

Evil site frames good site
Evil site covers good site by putting dialogue boxes or other elements on top of parts of framed site to create a different effect
Inner site now looks different to user
Compromise visual integrity – target

- Hiding the target
- Partial overlays
UI Subversion: *Clickjacking*

- An attack application (script) compromises the *context integrity* of another application’s *User Interface* when the user acts on the UI.

**Visual integrity**
- Target is visible
- Pointer is visible

**Temporal integrity**
- Target\_clicked = Target\_checked
- Pointer\_clicked = Pointer\_checked

**Context integrity** consists of visual integrity + temporal integrity

1. Target checked
2. Initiate click
3. Target clicked
Compromise visual integrity – target

- Hiding the target
- Partial overlays
Compromise visual integrity – pointer: cursorjacking

• Can customize cursor!

CSS example:
```css
#mycursor {
cursor: none;
width: 97px;
height: 137px;
background: url("images/custom-cursor.jpg")
}
```

• Javascript can keep updating cursor, can display shifted cursor

Fake cursor, but more visible

Real cursor visible
Compromise visual integrity – pointer: cursorjacking

Cursorjacking deceives a user by using a custom cursor image, where the pointer was displayed with an offset.

Fake, but more visible

real
Clickjacking to Access the User’s Webcam

You will be redirected to the requested page in 60 seconds.

Adobe Flash Player Settings
Camera and Microphone Access
www.webperflab.com is requesting access to your camera and microphone. If you click Allow, you may be recorded.

Allow  Deny
Defeating sitekeys

• Some sites use/used a secret image to identify site to user (e.g., Bank of America)
  • only good site should know the secret image
  • user should check that they receive the correct image

• What is it aimed to protect against?
  • phishing attacks

Invented by Berkeley grad student!

Not really used much now, not considered effective mostly because users ignore these images and don’t remember what the image was for each site.
How can clickjacking subvert sitekeys?

- Phishing sites frame login page to get correct image to appear
- Overlay input box from outer frame at the same location as the password box for the inner frame
- User types password accessible to attacker now
How can we defend against clickjacking?

Discuss with a partner
Defenses

• User confirmation
  - Good site pops dialogue box with information on the action it is about to make and asks for user confirmation
  - Degrades user experience

• UI randomization
  - Good site embeds dialogues at random locations so it is hard to overlay
  - Difficult & unreliable (e.g. multi-click attacks)
Defense 3: Framebusting

Web site includes code on a page that prevents other pages from framing it.
What is framebusting?

Framebusting code is often made up of

• a conditional statement and
• a counter action

Common method:

```java
if (top != self) {
    top.location = self.location;
}
```
Framebusting is very common at the Alexa Top 500 sites

[global traffic rank of a website]

<table>
<thead>
<tr>
<th>Sites</th>
<th>Framebusting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 10</td>
<td>60%</td>
</tr>
<tr>
<td>Top 100</td>
<td>37%</td>
</tr>
<tr>
<td>Top 500</td>
<td>14%</td>
</tr>
</tbody>
</table>

credit: Gustav Rydstedt
Many framebusting methods

<table>
<thead>
<tr>
<th>Conditional Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (top != self)</td>
</tr>
<tr>
<td>if (top.location != self.location)</td>
</tr>
<tr>
<td>if (top.location != location)</td>
</tr>
<tr>
<td>if (parent.frames.length &gt; 0)</td>
</tr>
<tr>
<td>if (window != top)</td>
</tr>
<tr>
<td>if (window.top != window.self)</td>
</tr>
<tr>
<td>if (window.self != window.top)</td>
</tr>
<tr>
<td>if (parent &amp;&amp; parent != window)</td>
</tr>
<tr>
<td>if (parent &amp;&amp; parent.frames &amp;&amp; parent.frames.length &gt; 0)</td>
</tr>
<tr>
<td>if((self.parent &amp;&amp; !(self.parent==self)) &amp;&amp; (self.parent.frames.length!=0))</td>
</tr>
</tbody>
</table>
Many framebusting methods

<table>
<thead>
<tr>
<th>Counter-Action Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>top.location = self.location</code></td>
</tr>
<tr>
<td><code>top.location.href = document.location.href</code></td>
</tr>
<tr>
<td><code>top.location.href = self.location.href</code></td>
</tr>
<tr>
<td><code>top.location.replace(self.location)</code></td>
</tr>
<tr>
<td><code>top.location.href = window.location.href</code></td>
</tr>
<tr>
<td><code>top.location.replace(document.location)</code></td>
</tr>
<tr>
<td><code>top.location.href = window.location.href</code></td>
</tr>
<tr>
<td><code>top.location.href = &quot;URL&quot;</code></td>
</tr>
<tr>
<td><code>document.write('')</code></td>
</tr>
<tr>
<td><code>top.location = location</code></td>
</tr>
<tr>
<td><code>top.location.replace(document.location)</code></td>
</tr>
<tr>
<td><code>top.location.replace('URL')</code></td>
</tr>
<tr>
<td><code>top.location.href = document.location</code></td>
</tr>
</tbody>
</table>
Most current framebusting can be defeated
Easy bugs

Goal: bank.com wants only bank.com’s sites to frame it if (top.location != location) {
  if (document.referrer &&
      document.referrer.indexOf("bank.com") == -1)
  {
    top.location.replace(document.location.href);
  }
}

Bank runs this code to protect itself:

Problem: http://badguy.com?q=bank.com
Abusing the XSS filter

IE8 reflective XSS filters:

On a browser request containing script:

http://www.victim.com?var=<script> alert('xss') … </script>

Server responds

Browser checks

If <script> alert('xss'); appears in rendered page, the IE8 filter will replace it with <sc#pt> alert('xss') … </sc#pt>

How can attacker abuse this?
Abusing the XSS filter

Attacker figures out the framebusting code of victim site
(easy to do, just go to victim site in attacker’s browser and view the source code)

```html
<script>
if(top.location != self.location) //framebust
</script>
```

Framing page does:

```html
<iframe src="http://www.victim.com?var=<script> if (top ... " >
```

XSS filter modifies framebusting script to:

```html
<script>
if(top.location != self.location)
```

XSS filter disables legitimate framebusting code!!
Defense: Ensuring visual integrity of pointer

- Remove cursor customization
  - Attack success: 43% -> 16%
Ensuring visual integrity of pointer

• Freeze screen around target on pointer entry
  – Attack success: 43% -> 15%
  – Attack success (margin=10px): 12%
  – Attack success (margin=20px): 4% (baseline: 5%)
Ensuring visual integrity of pointer

- Lightbox effect around target on pointer entry
  - Attack success (Freezing + lightbox): 2%
Enforcing temporal integrity

- UI delay: after visual changes on target or pointer, invalidate clicks for X ms
  - Attack success (delay=250ms): 47% -> 2% (2/91)
  - Attack success (delay=500ms): 1% (1/89)
Enforcing temporal integrity

• Pointer re-entry: after visual changes on target, invalidate clicks until pointer re-enters target
  – Attack success: 0% (0/88)
Other Forms of UI Sneakiness

• Users might find themselves living in *The Matrix* …
“Browser in Browser”

Apparent browser is just a fully interactive image generated by Javascript running in real browser!
Discussion

• So, how do these lessons apply to desktop applications?

• Compare the security model for desktop apps:
  – Are desktop apps safer against these attacks?
  – Are desktop apps riskier against these attacks?
Is there any hope?
Other defense: X-Frames-Options (IE8, Safari, FF3.7)

- Web server attaches HTTP header to response

  - Two possible values: DENY and SAMEORIGIN
    - DENY: browser will not render page in framed context
    - SAMEORIGIN: browser will only render if top frame is same origin as page giving directive

- Good defense … but poor adoption by sites (4 of top 10,000)

- Coarse policies: no whitelisting of partner sites, which should be allowed to frame our site
Summary

• Clickjacking is an attack on our perception of a page based on the UI

• Framebusting is tricky to get right
  • All currently deployed code can be defeated

• Use X-Frame-Options