CSE 484 / CSE M 584
Computer Security: More Lab 2 review

TA: Adrian Sham
adrsham@cs
Logistics / Reminders

- Homework 2 is due Nov 20 5pm
- Lab 2 out
- Today is last chance to send me encrypted email!
- Upcoming office hours:
  - Friday 9:00am – Gabe and Bo (CSE 220)
  - Wednesday 11:30 am – Yoshi (CSE 558)
  - Tuesdays 2:00pm – Adrian and Kiron (CSE 006)
Javascript

• Important for web development
  – **HTML** to define the content of web pages
  – **CSS** to specify the layout of web pages
  – **JavaScript** to program the behavior of web pages

• CSE 154 slides are helpful if you want to learn **everything** about Web Development (HTML, CSS, PHP, JavaScript)

• For this lab, should be able to learn as you go.

  http://www.w3schools.com/js/
JavaScript vs. Java

- **interpreted**, not compiled
- more relaxed syntax and rules
  - fewer and "looser" data types
  - variables don't need to be declared
  - errors often silent (few exceptions)

From: https://courses.cs.washington.edu/courses/cse154/13sp/lectures/slides/lecture18-javascript.shtml#slide5
XSS Filter Evasion Cheat Sheet

- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
XSS Overview

From: http://excess-xss.com/
# Input handling contexts

<table>
<thead>
<tr>
<th>Context</th>
<th>Example code</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML element content</td>
<td><code>&lt;div&gt;</code><strong>userInput</strong><code>&lt;/div&gt;</code></td>
</tr>
<tr>
<td>HTML attribute value</td>
<td><code>&lt;input value=&quot;userInput&quot;&gt;</code></td>
</tr>
<tr>
<td>URL query value</td>
<td><code>http://example.com/?parameter=userInput</code></td>
</tr>
<tr>
<td>CSS value</td>
<td><code>color: userInput</code></td>
</tr>
<tr>
<td>JavaScript value</td>
<td><code>var name = &quot;userInput&quot;;</code></td>
</tr>
</tbody>
</table>

From: http://excess-xss.com/
Example

<table>
<thead>
<tr>
<th><strong>Application code</strong></th>
<th><code>&lt;input value=&quot;userInput&quot;&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Malicious string</strong></td>
<td>&quot;&lt;script&gt;...&lt;/script&gt;&lt;input value=&quot;</td>
</tr>
<tr>
<td><strong>Resulting code</strong></td>
<td><code>&lt;input value=&quot;&quot;&gt;&quot;&amp;lt;script&amp;gt;...&amp;lt;/script&amp;gt;&amp;lt;input value=&quot;&quot;&gt;</code></td>
</tr>
</tbody>
</table>

From: http://excess-xss.com/
Event Handlers

- JavaScript can be called when an event occurs
- One example is ‘onerror’
- Example:
  - `<img src="badaddress.com" onerror="fireMissiles()"`>  
- Invalid image src will cause onerror to execute
XSS attack process

• Start by typing JavaScript into the “Send me an image link!” box
• See if you can get your browser to execute JavaScript (evade filters)
• Craft JavaScript to steal your own cookie and send it to a server
• Enter your attack JavaScript into “Send me an image link!” box
• You will be redirected
• Copy URL on address bar
• Go back
• Paste URL into “Send me an image link!”
XSS Demo

• Featuring Meowth and Pikachu
CSRF

HTTP Request ①
GET / HTTP/1.1
Host: www.example.org

HTTP Response ②
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1204

<html>
  ...<br />
  <img src="http://stocks.example.org/buy.php?symbol=S00X&shares=1000" />
  ...<br />
</html>

CSRF Attack ③
GET /buy.php?symbol=S00X&shares=1000 HTTP/1.1
Host: stocks.example.org

Victim

www.example.org

stocks.example.org

http://talks.php.net/show/xss-csr-f-apachecon2003/19
CSRF

1. Victim logs into bank
2. Cookie is set
3. Victim visits site/link with hidden img tag
4. Transfer money to attacker’s account
5. Bank validates the session and completes the transaction.

http://opensourceforu.efytimes.com/wp-content/uploads/2010/11/Figure-1-CSRF-attack-on-GET.png
Hack your 4.0!
Click Jacking

Thanks to Vitaly Shmatikov for Clickjacking slides
Click Jacking

• Clickjacking happens when an attacker uses different techniques to hijack clicks meant for their page and routing them to another

• Multiple techniques
  – Transparent UI elements on top of a button or link
  – Timing based attacks

https://www.owasp.org/index.php/Clickjacking
Example

• Video of click jacking

  https://www.youtube.com/watch?v=9V4_emKyAg8

• User is asked to play a game

• Button is quickly switched to a ‘save’ button
• Following slides by Vitaly Shmatikov
• http://www.cs.utexas.edu/~shmat/courses/cs361s/clickjack.ppt
Clickjacking (UI Redressing)

[Hansen and Grossman 2008]

- Attacker overlays multiple transparent or opaque frames to trick a user into clicking on a button or link on another page

- Clicks meant for the visible page are hijacked and routed to another, invisible page
Clickjacking in the Wild

• Google search for “clickjacking” returns 624,000 results... this is not a hypothetical threat!

• Summer 2010: Facebook worm superimposes an invisible iframe over the entire page that links back to the victim's Facebook page
  – If victim is logged in, automatically recommends link to new friends as soon as the page is clicked on

• Many clickjacking attacks against Twitter
  – Users send out tweets against their will
It’s All About iFrame

• Any site can frame any other site
  
  `<iframe src="http://www.google.com/...”>`
  
  `</iframe>`

• HTML attributes
  – Style
  – **Opacity** defines visibility percentage of the iframe
    • 1.0: completely visible
    • 0.0: completely invisible
Hiding the Target Element

[“Clickjacking: Attacks and Defenses”]

- Use CSS `opacity` property and `z-index` property to hide target element and make other element float under the target element.
- Using CSS `pointer-events: none` property to cover other element over the target element.
Partial Overlays and Cropping

[“Clickjacking: Attacks and Defenses”]

- Overlay other elements onto an iframe using CSS `z-index` property or Flash Window Mode `wmode=direct` property
- Wrap target element in a new iframe and choose CSS position offset properties
Drag-and-Drop API

[“Next Generation Clickjacking”]

- Modern browsers support drag-and-drop API
- JavaScript can use it to set data being dragged and read it when it’s dropped
- Not restricted by the same origin policy: data from one origin can be dragged to a frame of another origin
  - Reason: drag-and-drop can only be initiated by user’s mouse gesture, not by JavaScript on its own
Abusing Drag-and-Drop API

["Next Generation Clickjacking"]

1. Bait the user to click and start dragging

2. Invisible iframe with attacker’s text field under mouse cursor, use API to set data being dragged

3. Invisible iframe from another origin with a form field

With two drag-and-drops (simulated scrollbar, etc.), can select and extract arbitrary content from another origin

Frog. Blender. You know what to do.
Fake Cursors

[“Clickjacking: Attacks and Defenses”]

• Use CSS `cursor` property and JavaScript to simulate a fake cursor icon on the screen

`cursor: none`
Keyboard “Strokejacking”

[“Clickjacking: Attacks and Defenses”]

- Simulate an input field getting focus, but actually the keyboard focus is on target element, forcing user to type some unwanted information into target element