Week of February 8, 2016

Instructions. We will break into groups to discuss the following questions. Please think of as many solutions as you can. Be original! Maybe you will come up with something no one has thought of yet. Be prepared to talk about your solutions with the rest of the section.

Question 1  **Warmup: SOP**  
(5 min)

The Same Origin Policy (SOP) helps browsers maintain a sandboxed model by preventing certain webpages from accessing others. Two resources (can be images, scripts, HTML, etc.) have the same origin if they have the same protocol, port, and host. As an example, the URL http://inst.berkeley.edu/eecs has the protocol HTTP, its port is implicitly 80, the default for HTTP, and the host is inst.berkeley.edu.

Fill in the table below indicating whether the webpages shown can be accessed by http://amazon.com/store/item/83.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Can Access?</th>
<th>Reason if not</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://store.amazon.com/item/83">http://store.amazon.com/item/83</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://amazon.com/user/56">http://amazon.com/user/56</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="https://amazon.com/store/item/345">https://amazon.com/store/item/345</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://amazon.com:2000/store">http://amazon.com:2000/store</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td><a href="http://amazin.com/store">http://amazin.com/store</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 2  **Cross-Site Scripting (XSS)**  
(15 min)

The figure below shows the two different types of XSS.

**Stored XSS**

**Reflected XSS**
As part of your daily routine, you are browsing through the news and status updates of your friends on the social network FaceSpace.

(a) While looking for a particular friend, you notice that the text you entered in the search string is displayed in the result page. Next to you sits a suspicious looking student with a black hat who asks you to try queries such as

```html
<script>alert(42);</script>
```

in the search field. What is this student trying to test?

(b) The student also asks you to post the code snippet to the wall of one of your friends. How is this test different from part (a)?

(c) The student is delighted to see that your browser spawns a JavaScript pop-up in both cases. What are the security implications of this observation? Write down an example of a malicious URL that would exploit the vulnerability in part (a).

(d) Why does an attacker even need to bother with XSS? Wouldn’t it be much easier to just create a malicious page with a script that steals all cookies of all pages from the user’s browser?

(e) FaceSpace finds out about this vulnerability and releases a patch. You find out that they fixed the problem by removing all instances of `<script>` and `</script>`. Why is this approach not sufficient to stop XSS attacks? What’s a better way to fix XSS vulnerabilities?

**Question 3  Session Fixation (15 min)**

Some web application frameworks allow cookies to be set by the URL. For example, visiting the URL

```http
http://foobar.edu/page.html?sessionid=42.
```

will result in the server setting the `sessionid` cookie to the value “42”.

(a) Can you spot an attack on this scheme?

(b) Suppose the problem you spotted has been fixed as follows. `foobar.edu` now establishes new sessions with session IDs based on a hash of the tuple (`username`, `time of connection`). Is this secure? If not, what would be a better approach?