Paxson
Spring 2017

CS 161 Computer Security

Discussion 3

Week of February 6, 2017

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*** **(15 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`.

Origin	Can Access?	Reason if not
http://store.amazon.com/item/83		
http://amazon.com/user/56		
https://amazon.com/store/item/345		
http://amazon.com:2000/store		
http://amazin.com/store		

Question 2 ***Session Fixation*** **(10 min)**

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

`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?

Question 3 ***SQL Injection*** **(15 min)**

(a) Explain the bug in this PHP code. How would you exploit it? Write what you would need to do to delete all of the tables in the database.

```
$query = "SELECT name FROM users WHERE uid = $UID";
// Then execute the query.
```

(Here, `$UID` represents a URL parameter named `UID` supplied in the HTTP request. The actual representation of such a value in PHP is a bit messier than we've shown here. We leave out the syntactic details so we can focus on the functionality.)

(b) How does blacklisting work as a defense? What are some difficulties with blacklisting?

(c) What is the best way to fix this bug?

Question 4 ***Cross Site Request Forgery (CSRF)*** **(10 min)**

In a CSRF attack, a malicious user is able to take action on behalf of the victim. Consider the following example. Mallory posts the following in a comment on a chat forum:

```
<img src="http://patsy-bank.com/transfer?amt=1000&to=mallory"/>
```

Of course, Patsy-Bank won't let just anyone request a transaction on behalf of any given account name. Users first need to authenticate with a password. However, once a user has authenticated, Patsy-Bank associates their session ID with an authenticated session state.

(a) Explain what could happen when Victim Vern visits the chat forum and views Mallory's comment.

(b) What are possible defenses against this attack?