Section 8
SQL Injection, XSS, XSRF, exec()
Download the Code

git clone http://bitbucket.org/gerbercj/section8.git
SQL Injection

What’s the issue here?

$$\text{stmt} = \text{dbh} \rightarrow \text{query}( $$
   
   $$\text{sprintf(} \text{‘SELECT 1 FROM users}$$
   
   $$\text{ WHERE id=‘%s’ AND pwd=‘%s’}$$
   
   $$\text{), $user, $pwd}$$
   
$$\text{});}$$

What if $$\text{pwd} = ‘’ \text{ OR ‘’=’’}$$
What if $$\text{user} = ‘’ \text{; DROP TABLE users;}’’$$
Cross-Site Scripting

What is the issue here?

```php
$stmt = $dbh->prepare(
    'INSERT INTO comments (comment),
    VALUES (:comment));
$stmt->bindValue(':comment', $comment);
$stmt->execute();
...
foreach($dbh->query('SELECT *
    FROM comments') as $comment)
    print "<tr><td>{$comment[1]}</td></tr>";
```

What if the comment is:
<script>alert("Hacked!")</script>
Cross-Site Request Forgery

What is the issue here?

$stmt = $dbh->prepare(
  ‘INSERT INTO portfolios (id,symbol, shares),
   VALUES (:id,:symbol,:shares));
$stmt->bindValue(‘:id’, $_SESSION[‘id’]);
$stmt->bindValue(‘:symbol’, $_GET[‘symbol’]);
$stmt->bindValue(‘:shares’, $_GET[‘shares’]);
$stmt->execute();

What if I then visit a page with this:
exec()

What is the issue here?

if (isset($_POST['cmd']))
    exec($_POST['cmd'],$output);

What if run this:
wget -q0- http://section8/exec.php
--post-data="cmd=whoami"
Things to Remember...

• Escape your SQL parameters
  – mysql_real_escape_string()
  – PDO::prepare() / PDO::bindValue() / PDO::exec()

• Escape strings being displayed as HTML
  – htmlspecialchars()

• Confirm important transactions
  – Malicious sites can trick users into performing actions with their session cookies

• POST parameters are not safer than GET
  – Users can construct POST requests manually