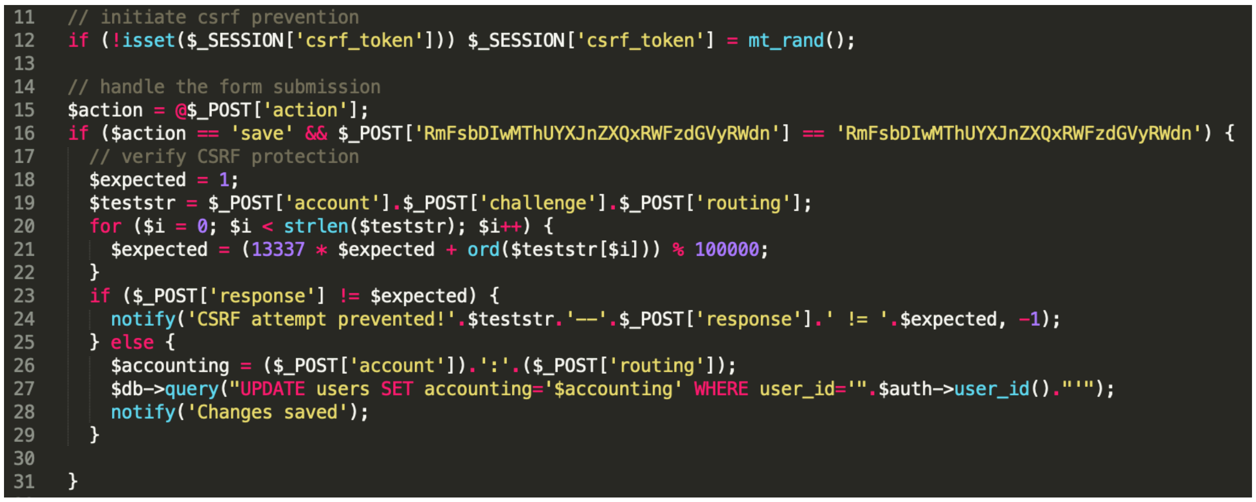
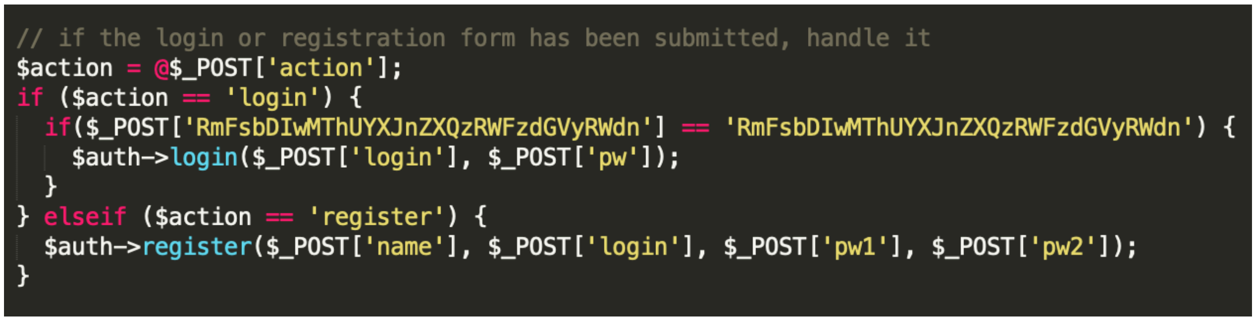
<Joshua Reno>
<jreno3>
CS 4235 Project 4 Epilogue

1. Target 1 Epilogue
1. Accounts.php (line numbers shown below, specifically the {if post[response] != expected} line).
2. The line where the response must (not) equal expected will notify of the attack so we can hardcode the response value as the expected value given the account, routing, and challenge values.
3. One way to avert this vulnerability is to have a random, unique session ID identifier sent with each request that is not hardcoded but generated.



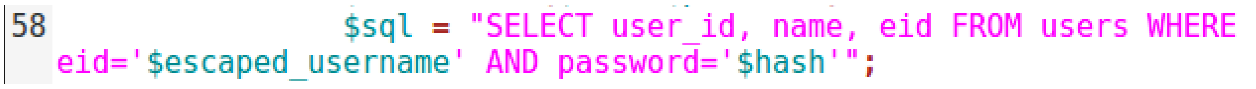
```
11  // initiate csrf prevention
12  if (!isset($_SESSION['csrf_token'])) $_SESSION['csrf_token'] = mt_rand();
13
14  // handle the form submission
15  $action = @$_POST['action'];
16  if ($action == 'save' && $_POST['RmFsbDIwMThUYXJnZXQxRWFzdGVyRWdn'] == 'RmFsbDIwMThUYXJnZXQxRWFzdGVyRWdn') {
17    // verify CSRF protection
18    $expected = 1;
19    $teststr = $_POST['account'].$_POST['challenge'].$_POST['routing'];
20    for ($i = 0; $i < strlen($teststr); $i++) {
21      $expected = (13337 * $expected + ord($teststr[$i])) % 100000;
22    }
23    if ($_POST['response'] != $expected) {
24      notify('CSRF attempt prevented!'.$teststr.'--'.$_POST['response'].' != '.$expected, -1);
25    } else {
26      $accounting = ($_POST['account']).':'.($_POST['routing']);
27      $db->query("UPDATE users SET accounting='$accounting' WHERE user_id='".$auth->user_id()."'");
28      notify('Changes saved');
29    }
30
31  }
```

2. Target 2 Epilogue
1. index.php (line numbers shown below, particularly auth->login).
2. The lines in index.php are not escaped so the attacker can run code in the login field and take the login credentials.
3. Using string escaping can prevent this issue. This is where a string is interpreted as a string and not as code or some other character. Escaping the script tag could prevent execution.



```
// if the login or registration form has been submitted, handle it
$action = @$_POST['action'];
if ($action == 'login') {
  if($_POST['RmFsbDIwMThUYXJnZXQzRWFzdGVyRWdn'] == 'RmFsbDIwMThUYXJnZXQzRWFzdGVyRWdn') {
    $auth->login($_POST['login'], $_POST['pw']);
  }
} elseif ($action == 'register') {
  $auth->register($_POST['name'], $_POST['login'], $_POST['pw1'], $_POST['pw2']);
}
```

3. Target 3 Epilogue
1. auth.php (line number shown below).
2. The line in auth.php escapes the username but not the password. By adding the "' OR’1'='1" string, we injected SQL and prevented the password from being necessary.
3. Prevent user input from being SQL queries. Also, the current sql filter doesn’t prevent the OR value or the apostrophe.



```
58                      $sql = "SELECT user_id, name, eid FROM users WHERE
   eid='$escaped_username' AND password='$hash'";
```