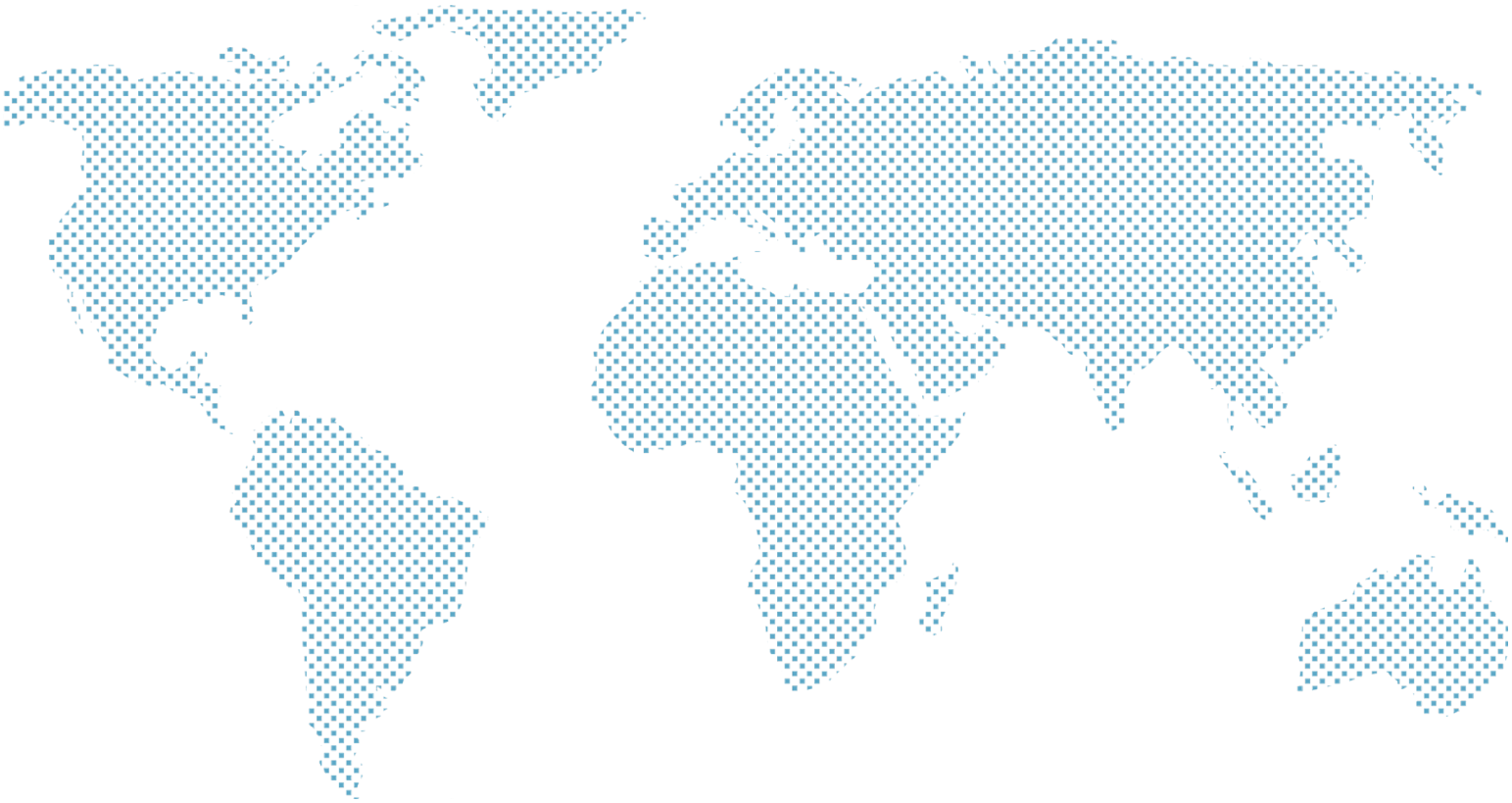
BAD TUTOR

Vulnerability Analysis Report



MARK WAGNER

SDEV300-7983 May 7, 2019 Supervisor: Abrom Cooper

BAD TUTOR

Table of Contents:

SDEV300-7983 May 7, 2019 Supervisor: Abrom Cooper

BAD TUTOR

Review of site:

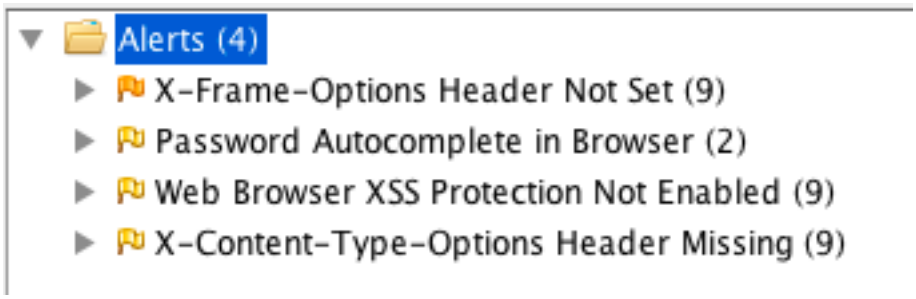
The site is a tutoring page where users can log in, read and delete sessions created by students stored on a MySQL database. Without logging in, users can sign a guest book or upload a resume which is stored in the /temp/ folder of the application.

Unfortunately this site has numerous security flaws outlines below:

Manual Attack Findings:

I was able to delete a session and after modification, upload my resume.

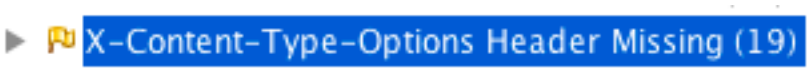
The error generated from the failed resume upload (due to write-protected temp folder) was causing an application error disclosure. This can be prevented in the future by setting parameter on move_uploaded_file to quiet.



The X Frame issue I attempted to correct by adding to httpd.conf:

```
Header set X-Frame-Options SAMEORIGIN
```

Now, Instead of a medium error, I get a low level error that NOSNIFF is not on which is OKAY. The remaining threats are also not issues as I will explain after auto-scanning.



BAD TUTOR

Automated Attack Findings:

- Alerts (7)
  - Cross Site Scripting (Reflected)
  - Directory Browsing
  - Parameter Tampering (4)
  - X-Frame-Options Header Not Set (15)
  - Password Autocomplete in Browser (2)
  - Web Browser XSS Protection Not Enabled (17)
  - X-Content-Type-Options Header Missing (19)

Major Issues:

Insecure HTTP Session:

```
This XSS notice could be mitigated overall with the following:
```

```
class SessionManager
{
   static function sessionStart($name, $limit = 0, $path = '/', $domain = null, $secure = n
ull)
   {
      // Set the cookie name before we start.
      session_name($name . '_Session');

      // Set the domain to default to the current domain.
      $domain = isset($domain) ? $domain : isset($_SERVER['SERVER_NAME']);

      // Set the default secure value to whether the site is being accessed with SSL
      $https = isset($secure) ? $secure : isset($_SERVER['HTTPS']);

      // Set the cookie settings and start the session
      session_set_cookie_params($limit, $path, $domain, $secure, true);
      session_start();
   }
```

But instead, I used php escape character sanitation on the specific input handlers so that the application can continue to run without major modification or re-enabling of the response header, which would break the program. Source: TeamTreeHouse

BAD TUTOR

Major Issues Continued:

Reflected XSS: The following escapes were not enough to stop a ZED fuzzer from pushing code through to action. Corrected with escaping as shown in part 6.

```
// Get input
$message = trim( $_POST[ 'Message' ] )
$gname   = trim( $_POST[ 'guestname'


$message = stripslashes( $message );
$gname = stripslashes( $gname );
```

SQL Injection:

I ran into some trouble trying to eliminate check_inputs() from the authcheck.php in an attempt to eliminate the parameter tampering – display of error code information due to improper form type. Basically, before querying to SQL, the program checks the input to make sure that if it was submitted as type “text”, it meets the criteria of text. If email, then it should have an ‘@’, etc.. The low level issue is that if those criteria are NOT met, the error is displayed to the user, which is a mild security risk.

```
<br />
<b>Notice</b>:  Undefined index: wsuser in <b>/Applications/XAMPP/xamppfiles/htdocs/badtutor/authcheck.php</b> on li
b>27</b><br />
<html>
```

Description:

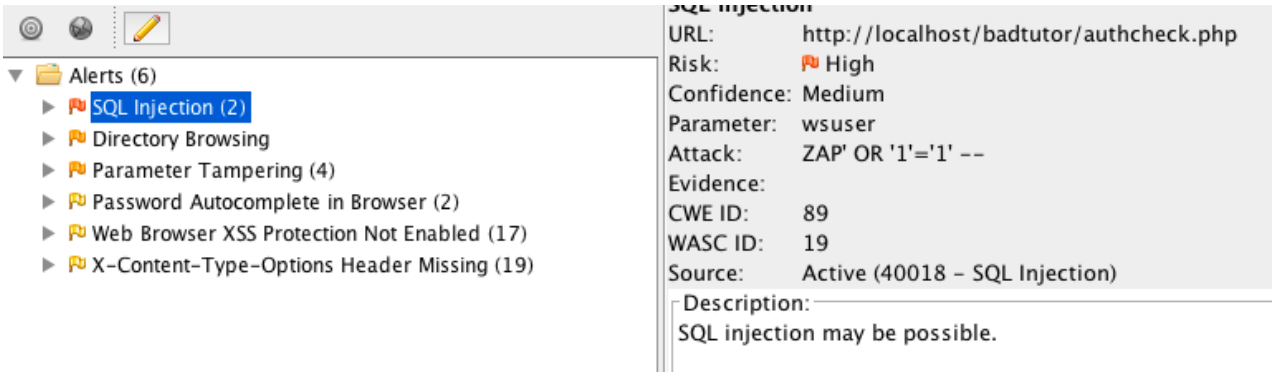
Parameter manipulation caused an error page or Java stack trace to be displayed. This indicated lack of exception handling and potential areas for further exp

```
// Retrieve Post Data - causing error display problem
//$wsuser = check_input($_POST['wsuser']);
//$wspass = check_input($_POST['wspass']);

//if (check_input($_POST['wsuser']) != )

// Authenticate User
//$count = findTutor($wsuser,$wspass);
$count = findTutor($_POST['wsuser'],$_POST['wspass']);
```

BAD TUTOR

Major Issues Continued: SQL Injection

By eliminating the check_inputs(), it opens the database up to unchecked input or SQL



Injection.

The only alternative to leaving the check_inputs() in place is to write my own sanitization script but leaving the check_inputs() function active presents no real risk so I will allow the Parameter Tampering flag to remain. Another way is to override the CI method and make it use a try / catch internally, or put one around it which has not worked as shown:

```
// Retrieve Post Data - causing error display problem
try {
    check_input($_POST['wsuser']);
} catch (Exception $e) {
    echo 'Caught exception of input: ',  $e->getMessage(), "\n";
} $wsuser = check_input($_POST['wsuser']);
```

Outside of this, I did not see any actual SQL Injection despite making a few manual attempts:

’ and password=‘ against: but it still fails the username thanks to count comparison.

```
// Define the Query
// For Windows MYSQL String is case insensitive
 $Myquery = "SELECT count(*) cnt from TutorDetails
        where tychoName='$tname' and password='$pass'";
```

BAD TUTOR

Medium Issues:

Directory Browsing: There was a published index to the site that I removed by changing the httpd.conf file to:

```
Options Indexes FollowSymLinks
AllowOverride All
Order allow,deny
Allow from all
```

```
<Directory "... /htdocs">
   Options FollowSymLinks
   AllowOverride None
   Require all granted
   Order allow,deny
   Allow from all
</Directory>
```

Unfortunately, this did not work. So I reverted it and added a .htaccess file to the directory with the following:

Options - Indexes

That too, did not work, so I removed the work “Indexes” from Options line in conf file based on advice from stack overflow. This also failed to help, so I changed index.html to _index.html

Application Error Disclosure (56) ı. The alert could be a false positive if tl

This led to 56 minor errors basically relating to errors regarding the now broken index reference. The issue of indexing really isn’t so major but it is up to the Administrators and developers to rebuild these references or keep the light risk.

SDEV300-7983 May 7, 2019 Supervisor: Abrom Cooper

BAD TUTOR

Minor Issues:

X-Content Options Header Missing: This is due a server-side setting that allows the browser to open new frames and needed for the application to work.

Web Browser XSS Protection is not enabled: This is the same problem as above.

Other Code Error & Vulnerability Analysis:

No Logout Function – Once leaving the site, the sessions would remain open, so if you revisited the authcheck.php file, the sensitive data would still be shown. I corrected this with a logout.php file that unsets and destroys the session and then redirects to the index.

```
<?php
session_start();
session_unset();
session_destroy();
header( string: "location:index.html");
exit();
?>
```

Login Error

Sorry, the username and password do not match any current accou
Try again, or contact the Tutor account administrator.

HTTP Only Cookie Protection disabled:

When uploading resume, temp folder had drop box/read/write permissions tuned off.

I now get:

```
File is valid, and was successfully uploaded.
Here is some more debugging info:Array
```

BAD TUTOR

Corrected Code Summary:

- Added logout button to pages that cancel http session.
- Minor Improvements to UI/UX with <br/> inserts.
- Reduced need to secure HTTP Sessions by random generation now that session terminates if host is changed or logout button is clicked.
- Also made name of session a random number beginning with uid instead of the $wsuser username which could be captured by the post request.

```
//$_SESSION['wsuser'] = $wsuser;
  $_SESSION['wsuser'] = uniqid( prefix: "uid");
```

I had to change later references for wsuser to wsusername which may have defeated the purpose. I am not sure how to get around using the session variable to hold the key reference for the tutors.

```
//$_SESSION['wsuser'] = $wsuser;
  $_SESSION['wsuser'] = uniqid();
  $_SESSION['wsusername'] = $wsuser;
```

I may have been barking up the wrong tree as the actual underlying session ID generated by php is already an md5 hash:

BAD TUTOR

If you want to know how PHP generates a session ID by default check out the source code on Github. **It is certainly not random** and is based on a hash (default: md5) of these ingredients (see line 310 of code snippet):

1. **IP address** of the client
2. **Current time**
3. **PHP Linear Congruence Generator** - a pseudo random number generator (PRNG)
4. **OS-specific random source** - if the OS has a random source available (e.g. /dev/urandom)

Corrected Code Summary Continued:

- Corrected XSS vulnerability by properly escaping input on guestrecord.php for $message and $gname.

```
$message = htmlentities($message,  quote_style: E
$gname = htmlentities($gname, quote_style: ENT_QU

$message = htmlspecialchars($message);
$gname = htmlspecialchars($gname);

$message = strip_tags($message);
$gname = strip_tags($gname);
```

Conclusion:

These remaining vulnerabilities have been determined to be benign.

- Alerts (3)
  - Password Autocomplete in Browser (2)
  - Web Browser XSS Protection Not Enabled (17)
  - X-Content-Type-Options Header Missing (19)

I hid the images and temp folder but the one remaining Directory Browsing vulnerability relates to a nonexistent directory. There is no icons folder. This does not appear if I attack localhost/badtutor specifically.

BAD TUTOR

Directory Browsing	
URL:	http://localhost/icons/
Risk:	Medium

This application has a long way to go as far as functionality but as a noticeboard for tutors it is relatively safe.

The best way to reduce many of these low level errors and vulnerabilities is to use a sophisticated, modern and supported DOM such as Angular.js which can support OAUTH, allowing UMUC users to directly make the switch over to this tutor page from the main site.

Sources:

[Edge], E. (n.d.). Combating ClickJacking With X-Frame-Options. Retrieved May 7, 2019, from https://blogs.msdn.microsoft.com/ieinternals/2010/03/30/combating-clickjacking-with-x-frame-options/

How to Create Bulletproof Sessions. (2017, July 11). Retrieved from https://blog.teamtreehouse.com/how-to-create-bulletproof-sessions

ArthurPrsar 2. (n.d.). How do I disable directory browsing? Retrieved from https://stackoverflow.com/questions/2530372/how-do-i-disable-directory-browsing

Kavinda, S. (2018, April 11). Input Validation with PHP. Retrieved from https://developer.hyvor.com/php/input-validation-with-php

Trott, T., & Trott, A. T. (2019, February 19). Parameter Tampering and How to Protect Against It. Retrieved from https://lonewolfonline.net/parameter-tampering-protect/