

# BAD TUTOR SECURITY AND CODE ANALYSIS

## LAB 8

SDEV<sub>300</sub>

Submitted by: Daniel Graham

## CONTENTS

Overview and goals	2
Summary of Findings	3
Findings	4
Findings	Guestbook 5
	Logging In 6
	Authentication 8
	Uploading your resume 9

## OVERVIEW AND GOALS

The objective of this report is to analyze, document and supply code fixes for the security vulnerabilities for the Bad Tutor web application. The types of security problems that I will cover are authentication, cross site scripting and SQL injections. The application will be scanned using the ZAP tool, and manually reviewed for exploits.

Bad tutor is an application designed to manage tutor sessions.

Bad tutor does the following functions:

- tutor login
- tutor resume upload
- guest book signing
- tutor view assignments
- tutor delete assignments

## SUMMARY OF FINDINGS

The app "Bad Tutor" is chalk full of dead code, and SQL injection and cross-site scripting possibilities. The major areas of concern is places where users provide input, and data leakage. The guest book was a major area for SQL injection and XSS. Another major area was the resume file uploading as it enabled any user (as its anonymous) to upload any file type, this means that a user could potentially run code on the server and potentially other users' machines. The database was of a concern too as it openly stored user passwords in plain text.

# FINDINGS

The following are the details of the research from code review

## Application installation

The database user has root level access. This poses a high security threat not only to the application but to any application or user of the database. Best practice is to create a user for each web application and only give the permissions required.

## Database

The database user has root level access. This poses a high security threat not only to the application but to any application or user of the database. Best practice is to create a user for each web application and only give the permissions required.

Through inspection of the database the design seems sound. There is an issue with password storage. The tutor passwords are stored in plain text. This is extraordinarily dangerous. The passwords have now been hashed and verified when the user logs in.

## GUEST BOOK

The page Guestbook.html submits to guestRecord.php. The first form input (name) is open to Cross-site scripting and SQL injection. The second (comments) is open to SQL injection. Wrapping \$gname in check\_input prevents XSS. Converted the insert guest function into a prepared statement.

I also check for the return value and insert the appropriate sentence instead of just outputting the variable. Also added a go back to homepage link to both the guest book and the confirmation page.

## LOGGING IN

The system appears to be checking usernames and passwords , and rejecting empty inputs. Once logged in there is no way to log out. repaired this broken functionality by adding a logout link the listSessions page.

The auth check is repeating code, and doing too many things at once. The actual checking for the user is a plain text check, and the app is only checking if the combination exists.

## Schedules

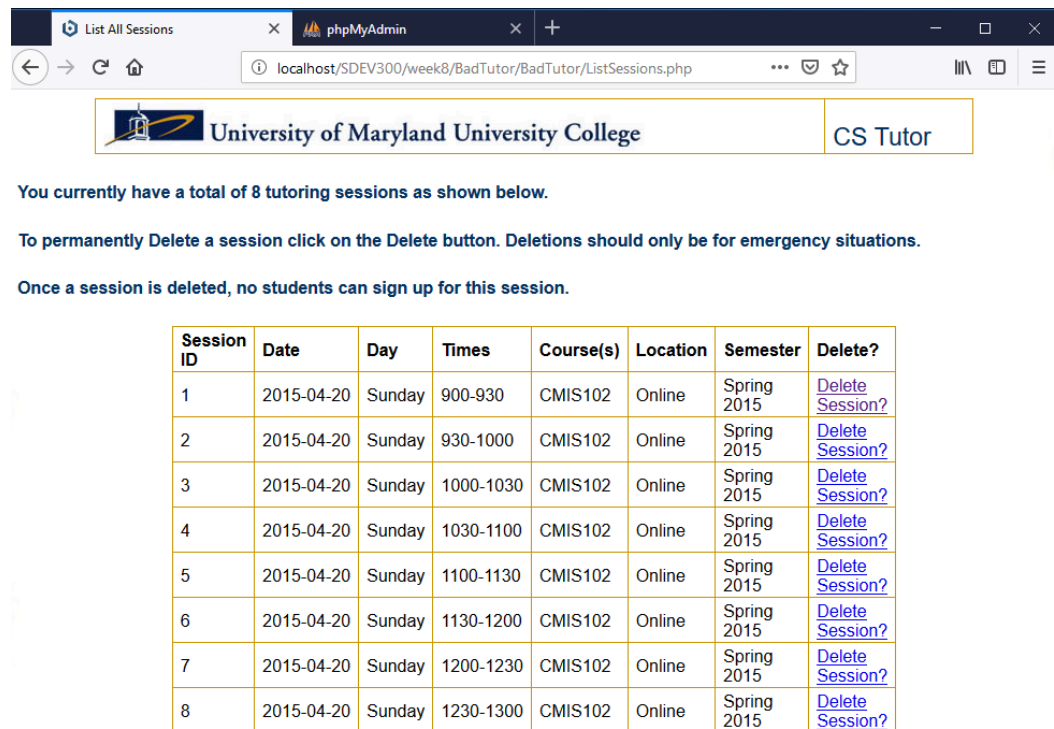
This is resulting in TWO lists of the sessions for a tutor, both of which are in a different format. The list show on the authCheck.php page:



The screenshot shows a web browser window with the URL `localhost/SDEV300/week8/BadTutor/BadTutor/authcheck.php`. The page header includes the University of Maryland University College logo and the text "CS Tutor". Below the header, there are three lines of text: "You currently have 8 tutoring sessions.", "Be sure to check your site daily as students can register at anytime.", and "Also, students must register and be on your schedule to receive tutoring assistance." The main content is a table with 7 columns: Course, Student Name, Email, Tutor Session Details, Location, Help Details, and Delete?. The table contains 8 rows of data, each representing a tutoring session for a specific student.

Course	Student Name	Email	Tutor Session Details	Location	Help Details	Delete?
CMIS141	Melody Fritz	mfritz@student.umuc.edu	2015-04-20,Sunday,900-930	Online	Need help on Project 1	<a href="#">Delete Session?</a>
CMIS141	Melody Fritz	mfritz@student.umuc.edu	2015-04-20,Sunday,930-1000	Online	Need help on Project 2	<a href="#">Delete Session?</a>
CMIS141	Melody Fritz	mfritz@student.umuc.edu	2015-04-20,Sunday,1000-1030	Online	Need help on Project 3	<a href="#">Delete Session?</a>
CMIS141	Melody Fritz	mfritz@student.umuc.edu	2015-04-20,Sunday,1030-1100	Online	Need help on Project 4	<a href="#">Delete Session?</a>
CMIS242	Jon Robbins	jrobbins@student.umuc.edu	2015-04-20,Sunday,1100-1130	Online	Need help on Project 1	<a href="#">Delete Session?</a>
CMIS242	Jon Robbins	jrobbins@student.umuc.edu	2015-04-20,Sunday,1130-1200	Online	Need help on Project 1	<a href="#">Delete Session?</a>
CMIS242	Jon Robbins	jrobbins@student.umuc.edu	2015-04-20,Sunday,1200-1230	Online	Need help on Project 1	<a href="#">Delete Session?</a>
CMIS242	Jon Robbins	jrobbins@student.umuc.edu	2015-04-20,Sunday,1230-1300	Online	Need help on Project 1	<a href="#">Delete Session?</a>

The tutoring list shown after you delete a session or choose not to delete a session:



You currently have a total of 8 tutoring sessions as shown below.

To permanently Delete a session click on the Delete button. Deletions should only be for emergency situations.

Once a session is deleted, no students can sign up for this session.

Session ID	Date	Day	Times	Course(s)	Location	Semester	Delete?
1	2015-04-20	Sunday	900-930	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>
2	2015-04-20	Sunday	930-1000	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>
3	2015-04-20	Sunday	1000-1030	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>
4	2015-04-20	Sunday	1030-1100	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>
5	2015-04-20	Sunday	1100-1130	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>
6	2015-04-20	Sunday	1130-1200	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>
7	2015-04-20	Sunday	1200-1230	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>
8	2015-04-20	Sunday	1230-1300	CMIS102	Online	Spring 2015	<a href="#">Delete Session?</a>

from the logic of the it appears the list on authCheck.php is the tutor schedule for 2 weeks, and the listSessions.php is the tutor's group schedule. This is a user flow issue. It may pose a security threat if these two page's authentication scheme differ.



## AUTHENTICATION

the script `authcheck.php` calls the `findTutor` function which is open to SQL injection because it simply sends the user input off to the database without sanitizing it. A fix was deployed using `mysqli` prepared statements.

Another major issue with the authentication is that there are not comparisons of hashes, as the password is stored in plain text in the database. `password_verify` function was added. the SQL call was modified such that the password hash was returned.

Had to change the database for tutor details to hold a `varchar 255` so that the hash could be properly stored. The `SQLcode.sql` was updated to reflect. `insert tutor details` has been commented out, and to a `insert-tutors.php` was created. in order to hash the passwords before sending them off to the database.

If you refresh the page on `authcheck.php` you are asked to log in even if you just logged in. fixed by refactoring the page to start the session immediately, and checked thw `'wsuser'` is set and if it is set, grab the username value.

## UPLOAD YOUR RESUME

The first issue is in UploadResume.html as it has a input that is hidden that sets the max file size. this is not recommended as a client can change the value on the client therefore a server-side check for size is required.

A big risk factor is that a user can upload ANY file of any type. To secure the file upload restrictions must be placed on the type of file that can be uploaded. You can do this on the client-side in HTML to let the user know what filetype you expect and will accept and do an check on the server to enforce the rule. I added both of these features for docx (word documents) and pdf.

For the file type confirmation a php.ini edit was required for the file\_info module.

```
;extension=php_fileinfo.dll
```

Had to be enabled.

The debugging info print\_r was also removed, This is a function that should only be in a development environment and not exposed to the user as it can provide useful information to a hacker.