WAMP ZAP Analysis and Mitigation

Overview

For this final lab you will use the tools and techniques used throughout the course to analyze, mitigate and document the results of a WAMP applications. The application to analyze uses a prototype UMUC tutoring application that allows a tutor to login, upload their resume, sign a guest book and view and delete tutor assignments. You will need to install the application on the SDEV AMI, run the analysis using ZAP, manual interceptions and code review, fix all vulnerabilities and document the results.

Learning Outcomes:

At the completion of the lab you should be able to:

1. Set-up and run the UMUC tutor application on your VM
2. Conduct automated and manual analysis on a LAMP application
3. Identify, prioritize and repair software vulnerabilities found in the application
4. Document the process, findings and specifics on how you resolved the issues.

Lab Submission Requirements:
After completing this lab, you will submit a word (or PDF) document that meets all of the requirements in the description at the end of this document. In addition, your associated files should be submitted. Those associated files will be the original files as well as a complete set of files with fixes implemented. You should submit multiple files in a winzip file.

Virtual Machine Account Information
Your Virtual Machine has been preconfigured with all of the software you will need for this class. You have connected to this machine in the previous labs. Reconnect again using the Remote Desktop connection, your Administrator username and password.

Tutor Application user accounts:

Username	Password
tutor1	t123
tutor2	t234
tutor3	t345

Part 1 – Set-up and Run the UMUC tutor application on your VM

In this exercise you will create and populate the database tables for the WAMP application and install the PHP and associated files on your VM. The application is fully functional (but definitely not safe or secure). You need to perform a few steps to make sure it is working properly on your VM.

1. Attached to this Project is zip folder. Download the BadTutor.zip file and upload to your SDEV AMI VM. The BadTutor.zip file contains the Web Application as well as SQL folder with SQL scripts to create the databases and tables needed for this application.

2. Once uploaded to the SDEV AMI, unzip the file and place the BadTutor folder in the htdocs folder as shown in figure 1.

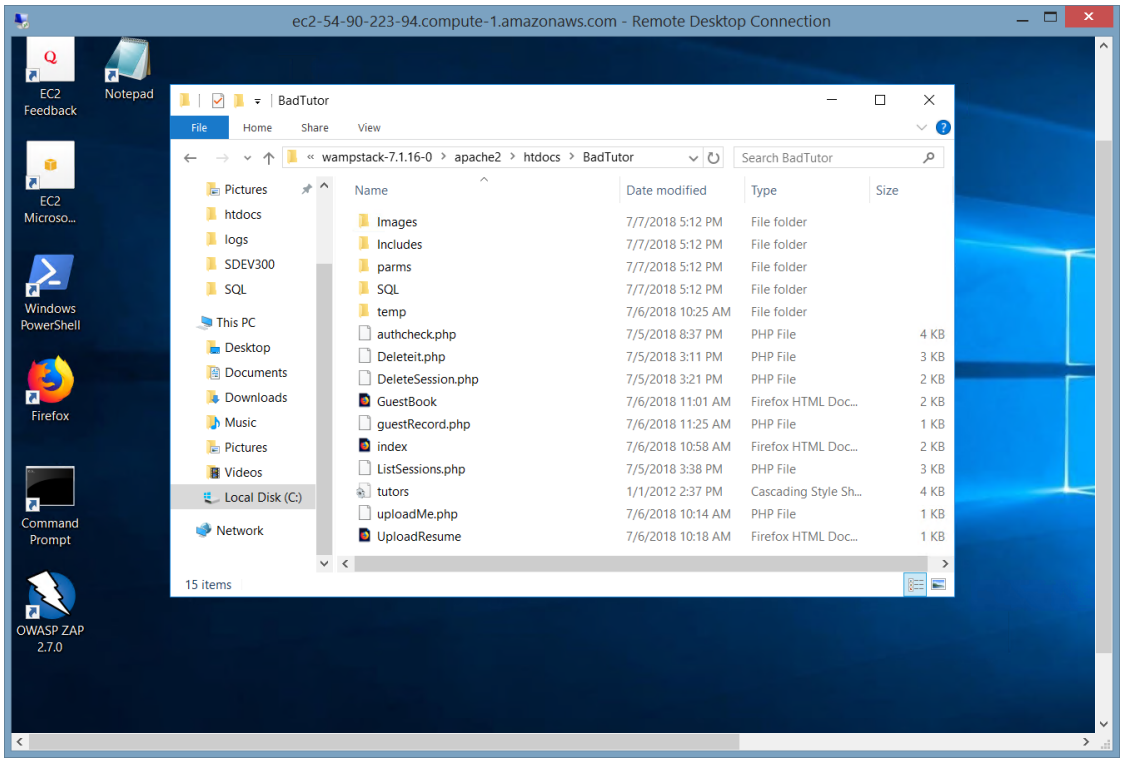

Figure 1 Move BadTutor to the htdocs folder

Next, we will open up the SQL folder and run the database scripts. This is a critical step needed so the Web application can properly communicate with the database.

3. Open your command prompt and change to the Bitnami\WampStack-7.1.16-0\mysql\bin directory and enter the following command:

   ```
   mysql -u root -p
   ```

   When prompted enter the following mysql root password:

   ```
   sdev300vm99
   ```

4. Next, we will run the database scripts in their entirety using the “source” option for MySQL. This will run each of the two SQL scripts without any real human intervention.

To run the scripts you will need to copy and paste or type the following commands at the MySQL prompt:

```
source C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\BadTutor\SQL\DBSetup.sql
```

```
source C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\BadTutor\SQL\SQLCode.sql
```

Note, the source command is used to run any SQL file. You just need to fully define the location (including the path). Figure 2 show the results of running the first Scripts which sets-up the database.

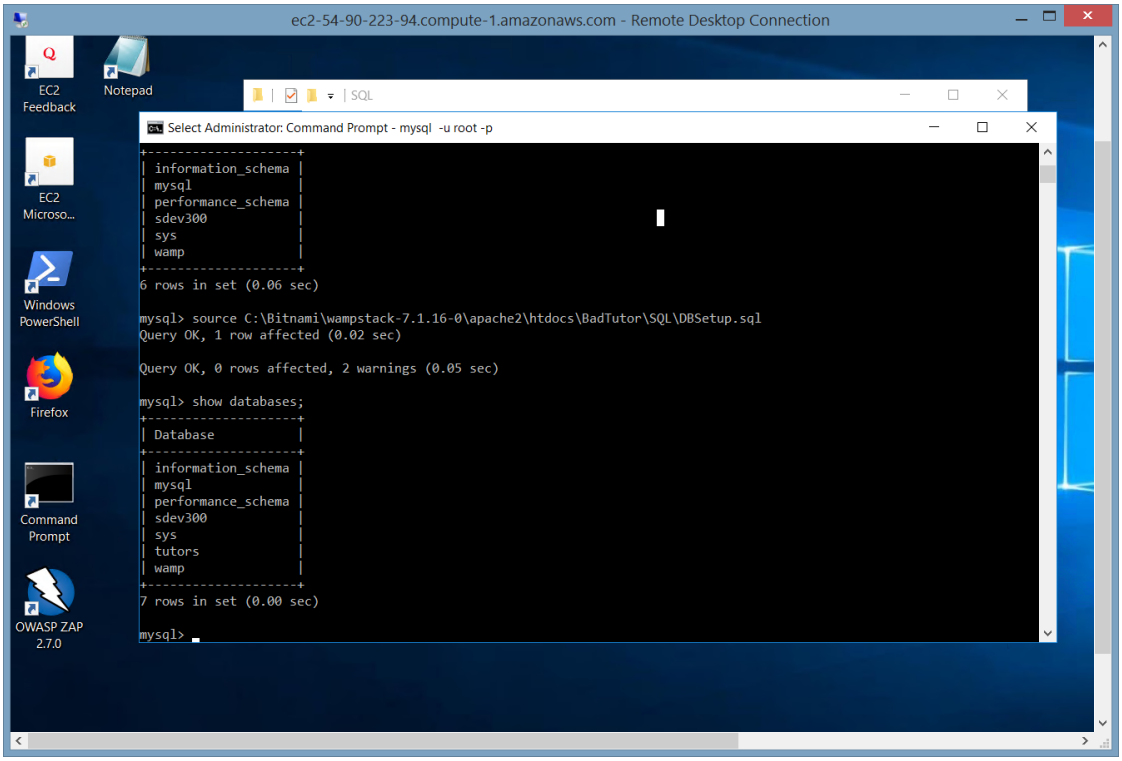

Figure 2 Running the DBSetup.sql script

Use of the `show database;` command before and after the set-up reveals the addition of the new tutors database. Be sure to carefully, review and understand each statement in the SQL script files.

Figure 3 shows the results of running the second SQL script. In this script, the tables are created and an initial set of data is populated for testing purposes.

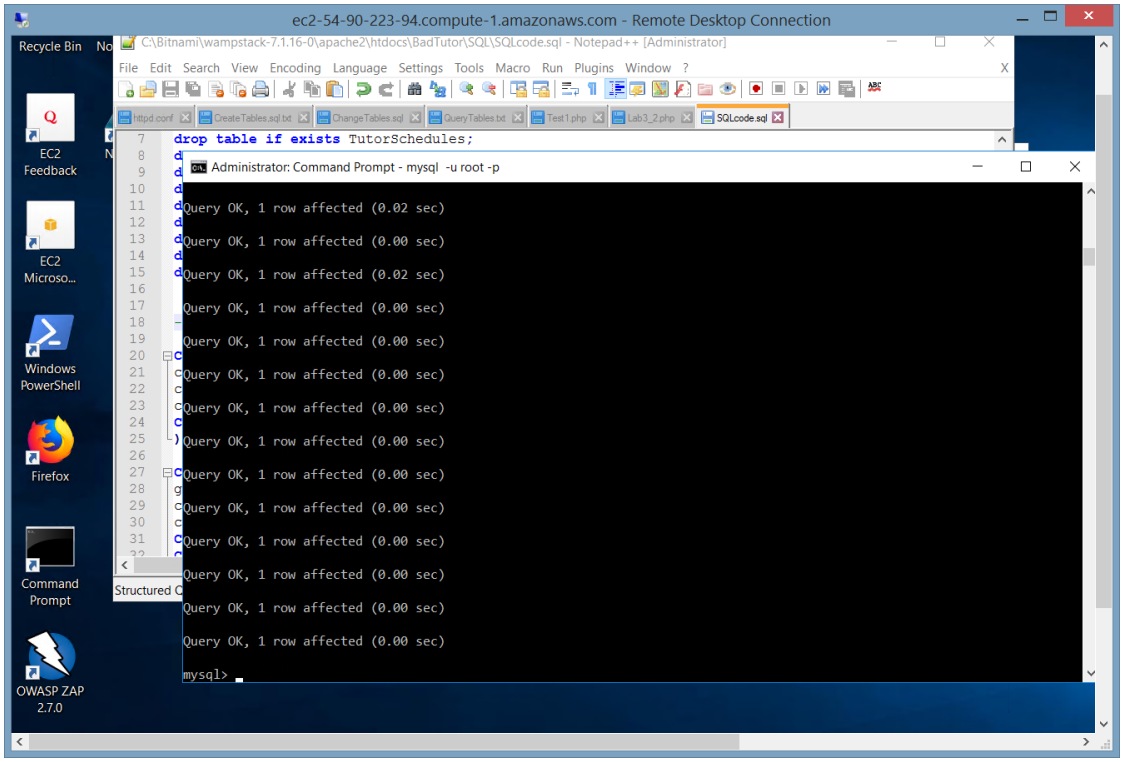

Figure 3 Running the SQLCode.sql file

You should run some queries to make sure your database scripts functioned properly. If they didn't, you can actually, drop the database and start again with the `drop database tutors;` command.

Some simple queries to test your tables are shown below:

- `Use tutors;`
- `Select * from TutorDetails;`
- `Select * from  StudentSchedules;`
- `Select * from TutorSchedules;`
- `Select * from GroupSchedules;`
- `Select * from CourseGroups;`
- `Select * from TutorGroups;`
- `Select * from Students;`
- `Select * from Semesters;`
- `Select * from Courses;`
- `Select * from Tutors;`
- `Select * from GuestBook;`

Compare the results to the SQL files before proceeding to the next steps. If your database data is not correct, you won't be able to test your application.

Next, we will test to make sure the Web application is working.

5. Reset the httponly security option in your httpd.conf file

If you properly corrected the vulnerabilities and environment settings in week 5, you most likely added something similar to this statement in your httpd.conf file

```
Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

For this BadTutor application to function, you will need to comment out that line with a #:

```
#  Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
```

Also, be sure to stop and restart the Apache server for this change to take effect.

Now, the fact that we have to comment out a security protection to get an application to work should speak loudly to you about a security issue in this application. However; these is also some functionality associated with the application that is required. So when you fix the security issues you can't just break the application. You have to provide a better design that allows the desired functionality to be implemented securely. This is not always easy and often requires some risk assessment and trade-offs.

6. Open up your Browser and Launch the BadTutor app (localhost/BadTutor)

As shown in figure 4, you should be able to login using one of the 3 tutor credentials listed earlier in the document, or click on the links to upload a resume or sign the guest book. (Note: Be sure you have modified the FireFox proxy options to "No Proxy")

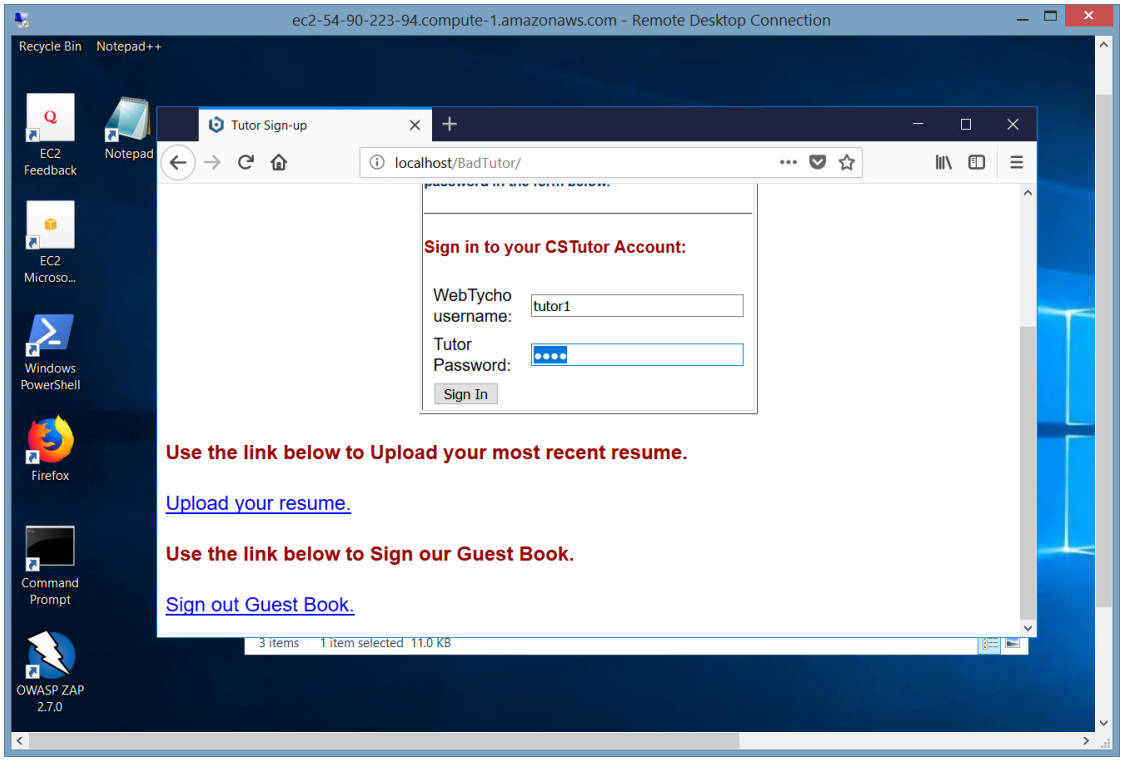

Figure 4 Testing the BadTutor Application

You should sign In with each of the tutor credentials, and experiment with all of the functionality so you understand what the application is designed to do. (Hint: Do you think those passwords are secure?)

7. Go through each link in the application testing the functionality so you fully understand the intent and purpose of the application. Login in as different tutors as you test the application. After you complete your security analysis and fixes, the baselines functionality should still be present but much more secure.

Figures 5 – 17 shows the results of running the application for the tutor1 user and deleting a tutor session, uploading a resume and signing the guest book.

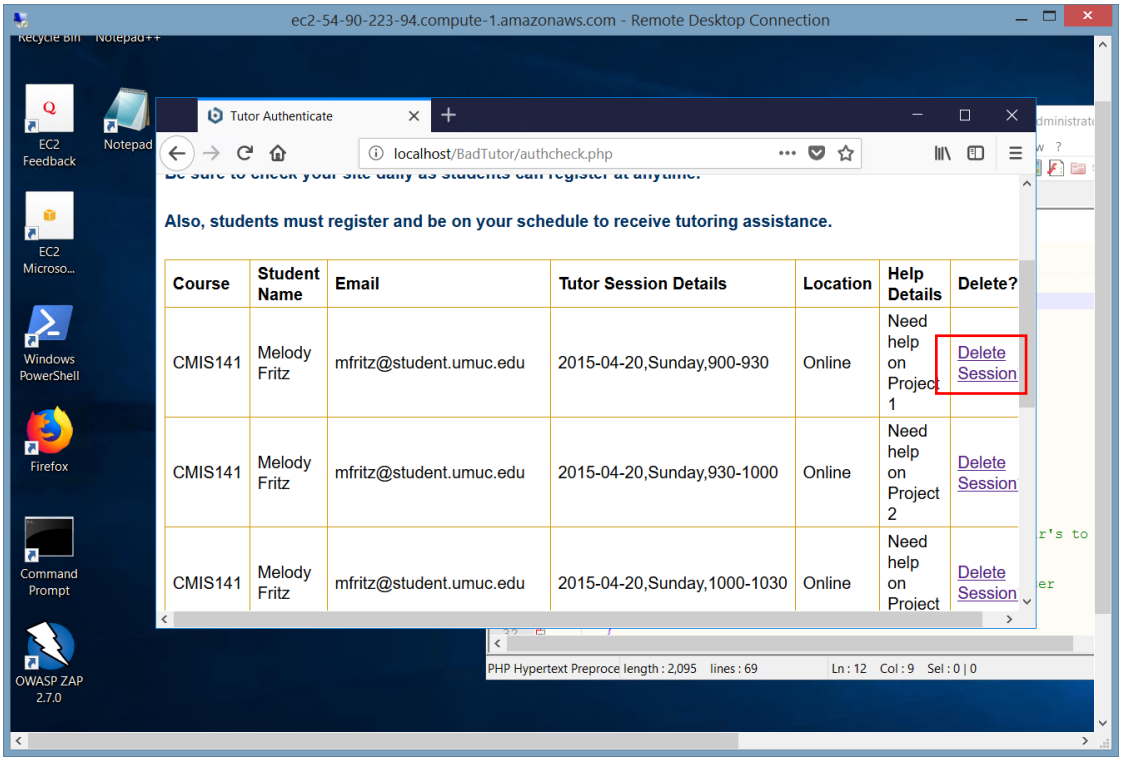

Figure 5 Select Delete Session

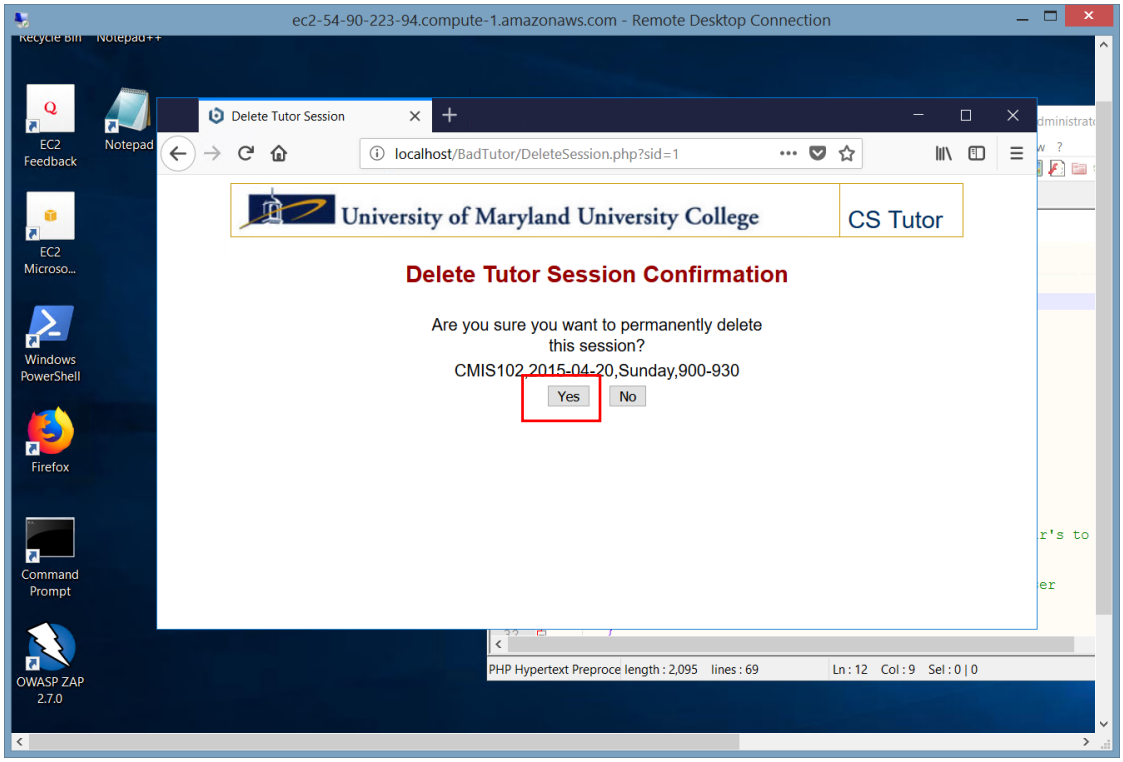

Figure 6 Confirm Deletion of Tutor Session

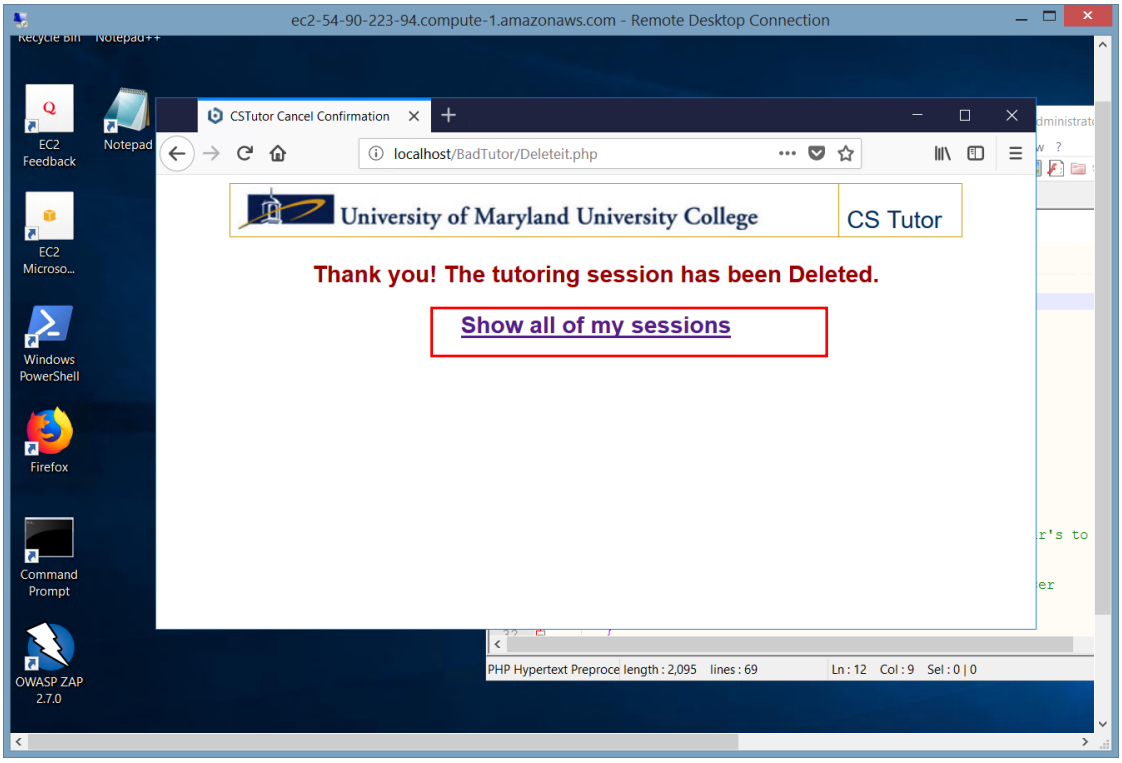

Figure 7 Show All Sessions

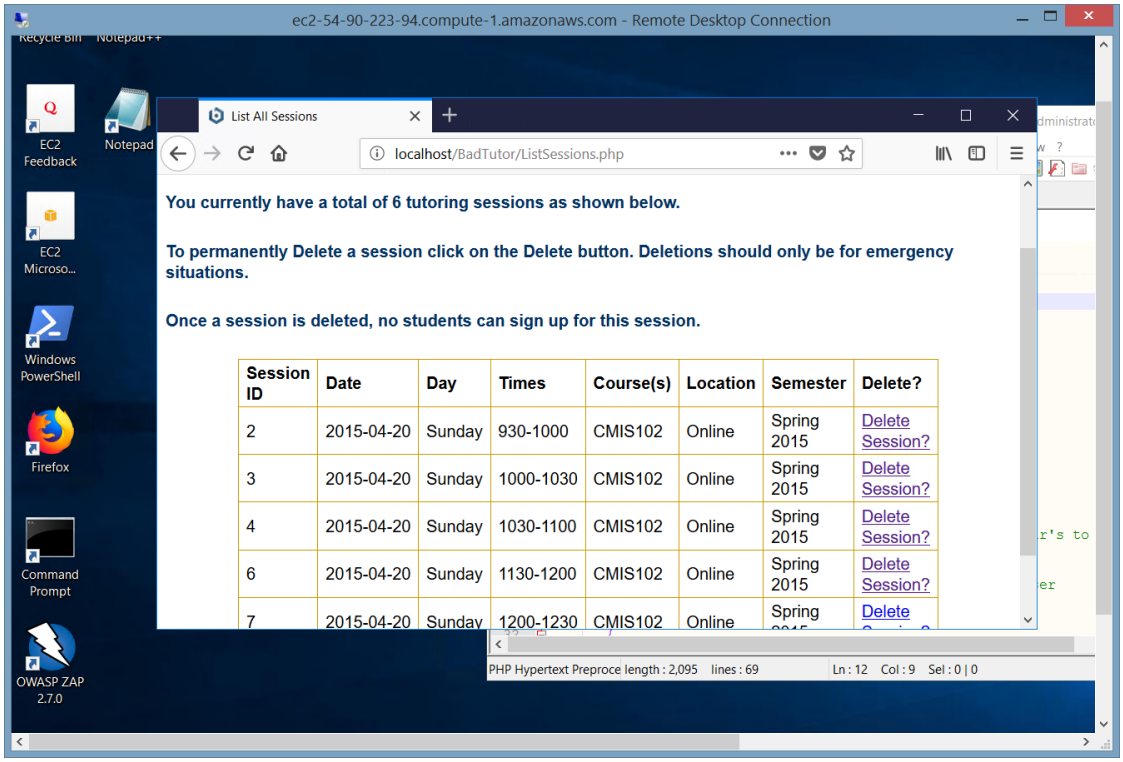

Figure 8 Note that the Tutor Session was Deleted

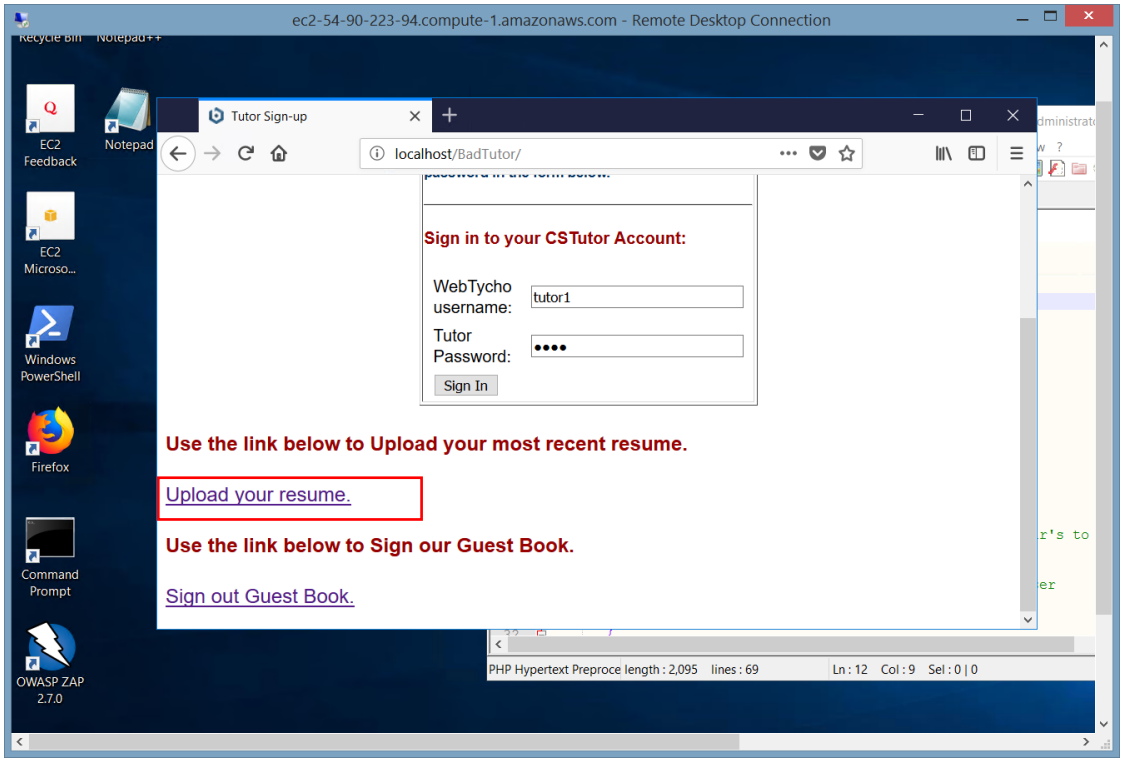

Figure 9 Select upload Resume

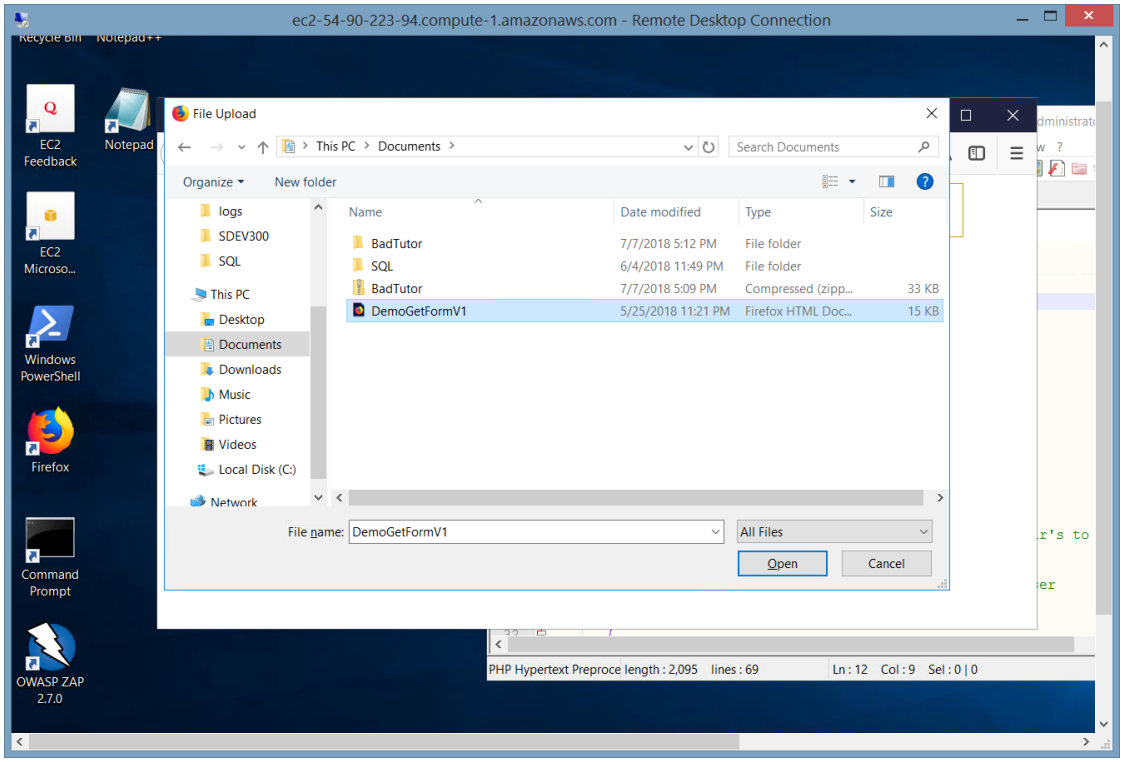

Figure 10 Selecting the Resume to Upload

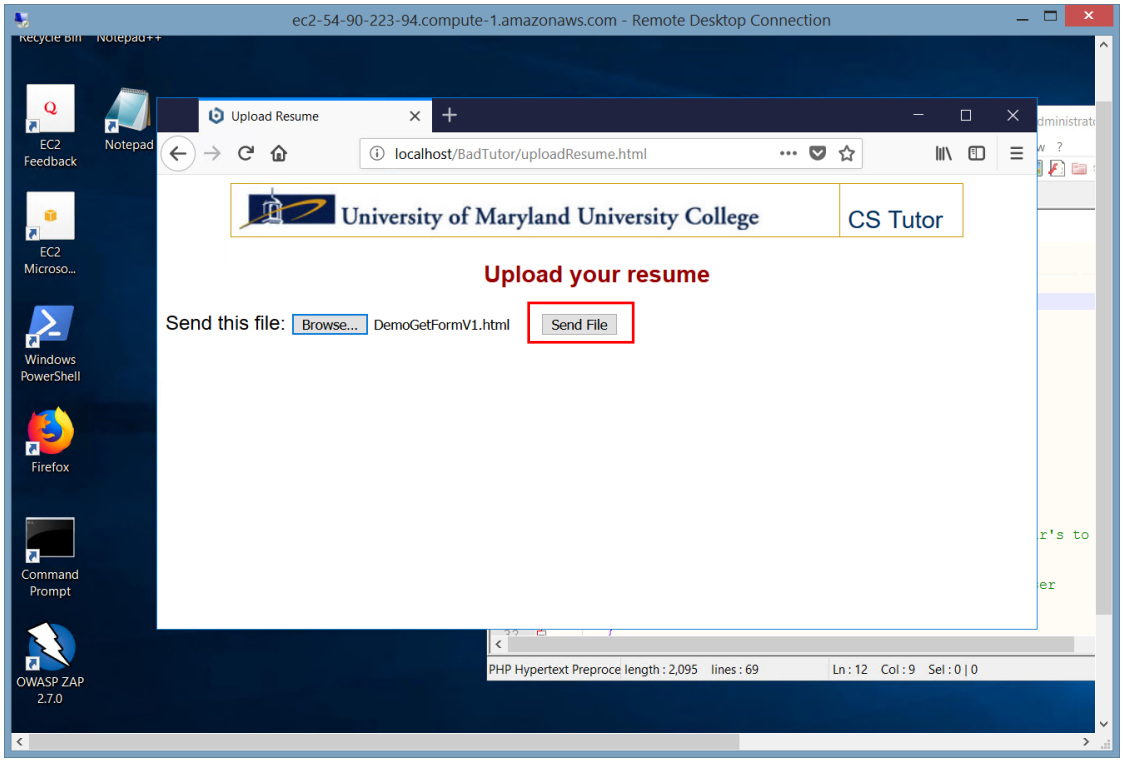

Figure 11 Sending the Resume

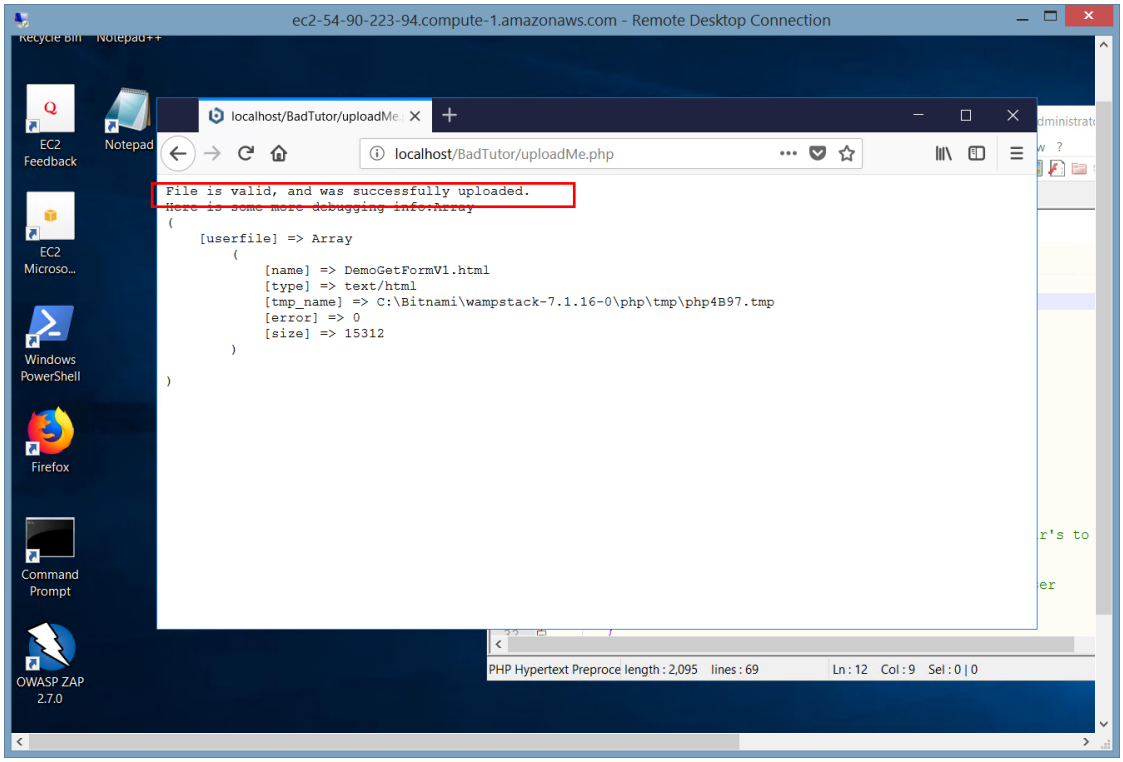

Figure 12 The File Was Sent

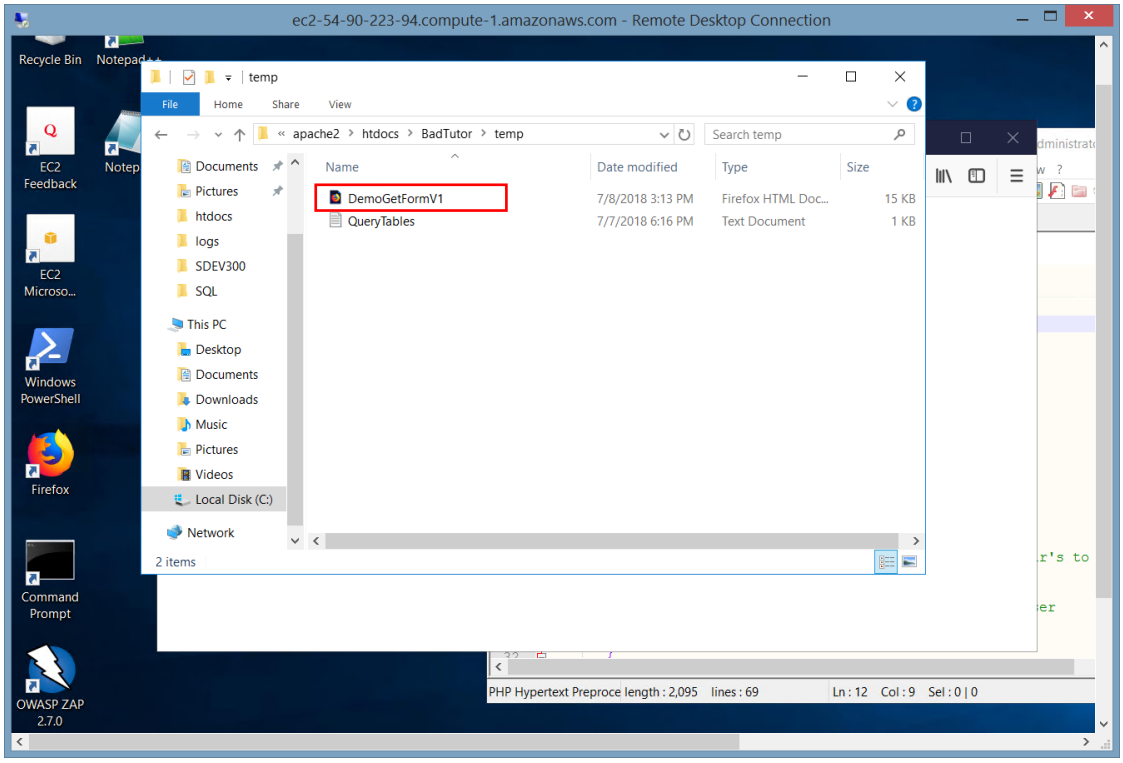

Figure 13 File was sent to BadTutor/Temp folder

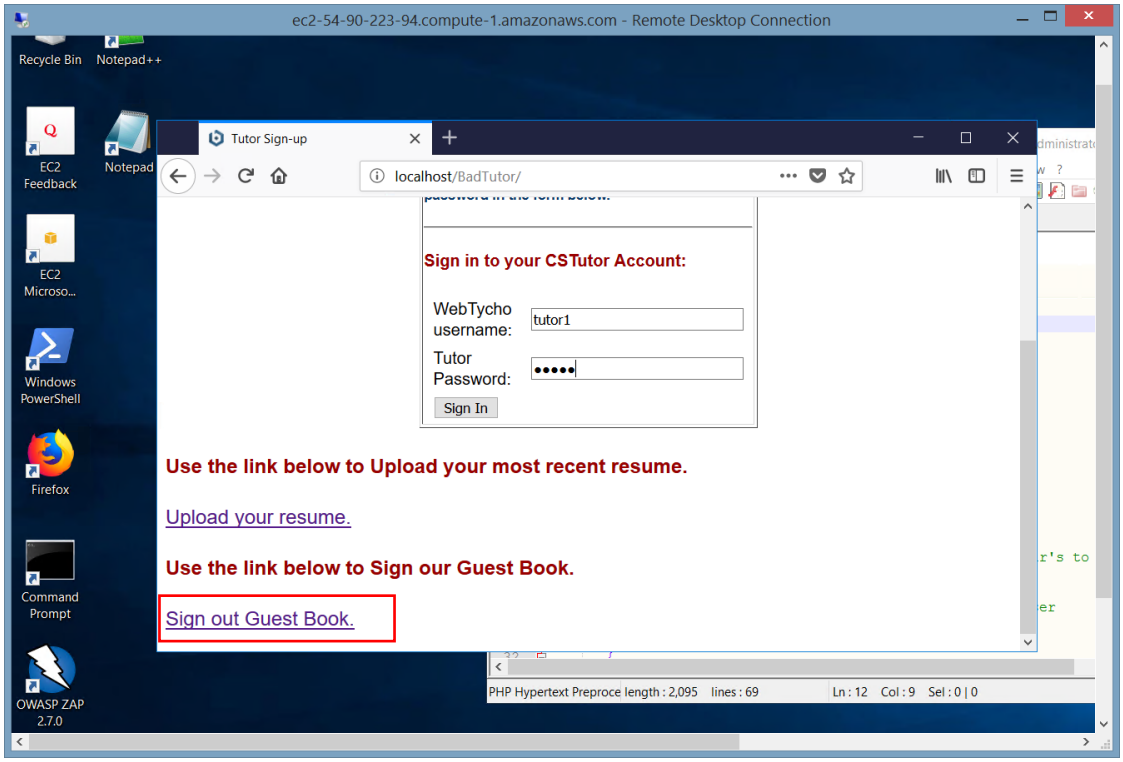

Figure 14 Selecting Link to Sign Guest Book

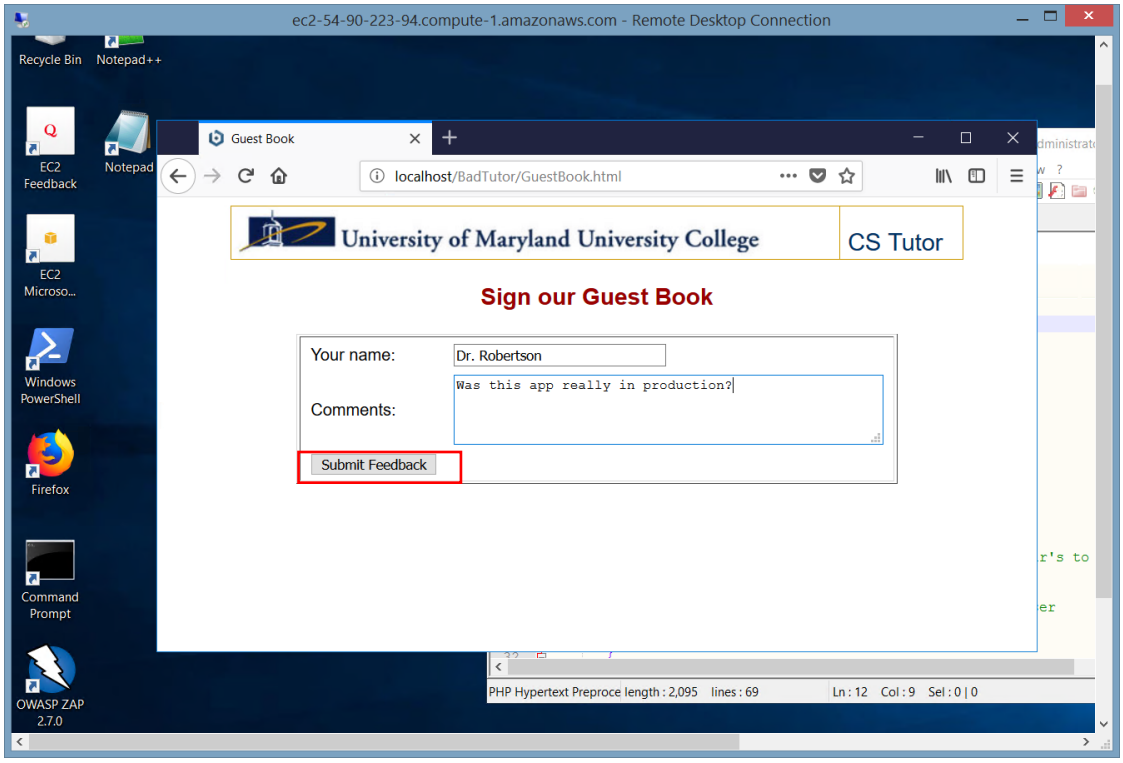

Figure 15 Entering Data in Guest Book

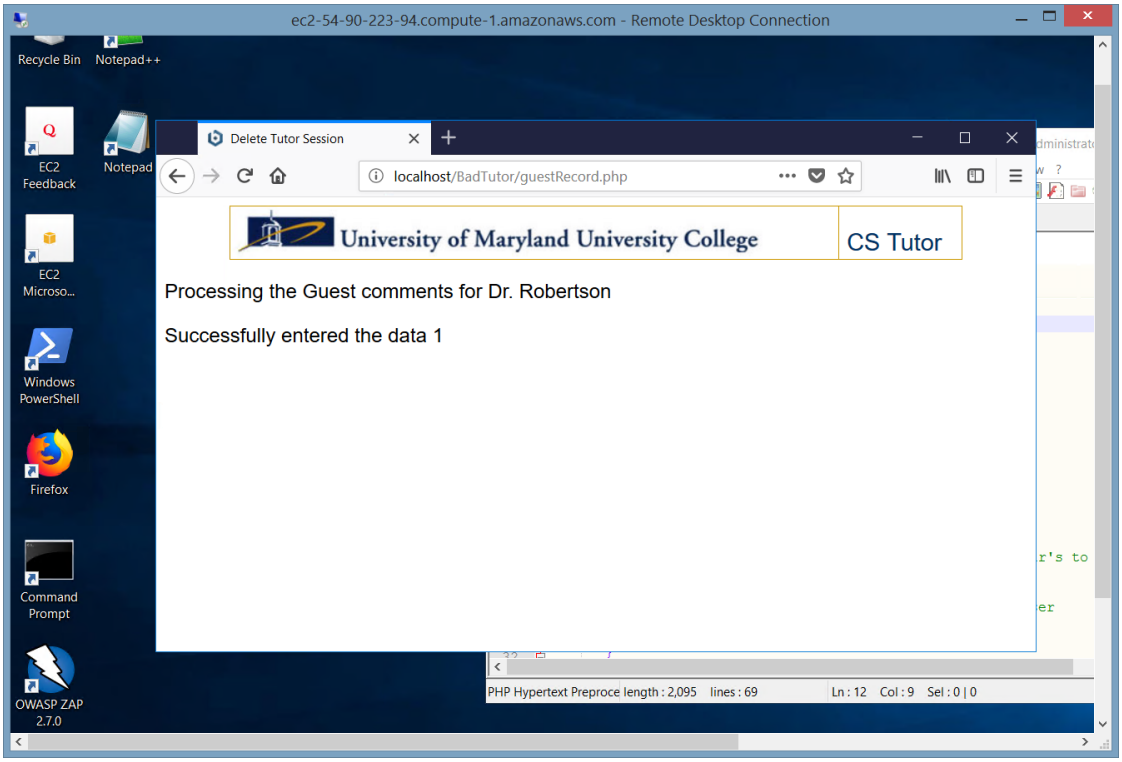

Figure 16 Data Entry Response

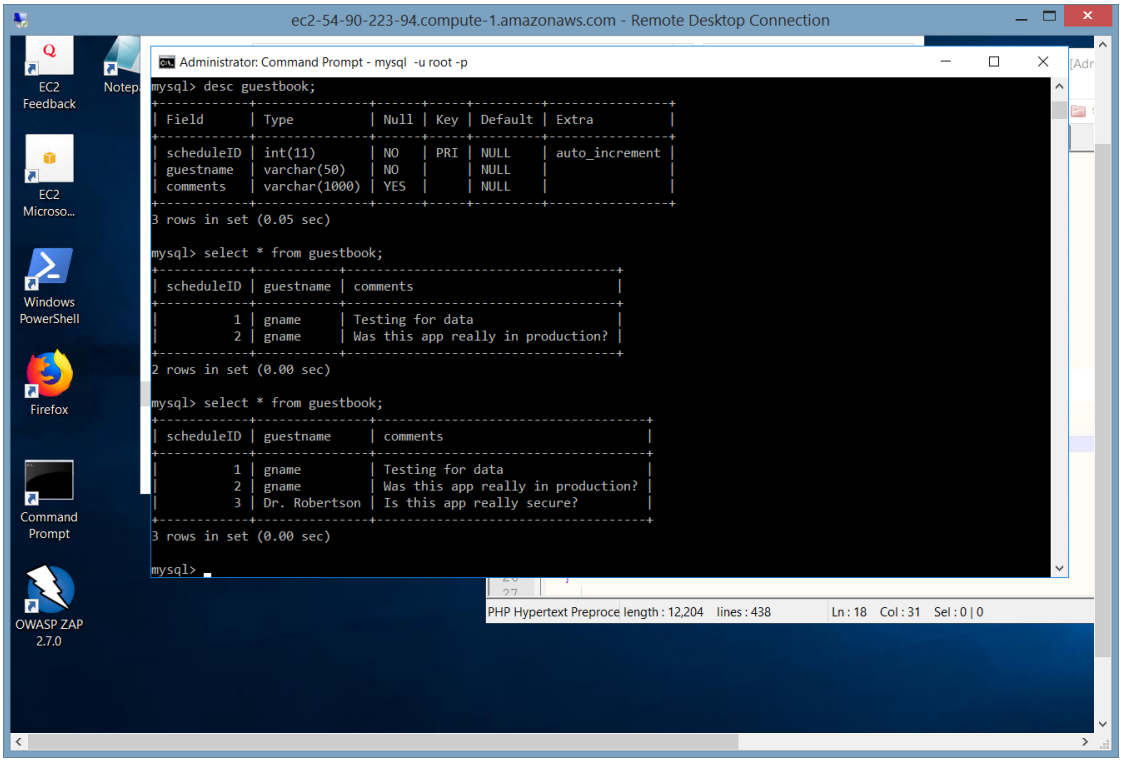

Figure 17 Logging into MySQL – tutors database and verifying data input

You should experiment with the application as needed to learn the functionality.

Note, the application is used to review and delete tutoring sessions. In this case, sessions refer to a 30 minute meeting between a student and the tutor. This is not the same as Web application or HTTP sessions. The HTTP sessions within this PHP application are the heart of the functionality as well as the vulnerabilities for this application.

Lab submission details:

As part of the submission for this Lab, you will run manual and automatic attacks on you're the BadTutor Web application. Be sure to carefully review the code for obvious security issues and information leakage.

You can provide the findings in one well-organized document. You should work to eliminate all alerts in the application and clearly document specifically what you did to mitigate each issue.

Create screen captures demonstrating your process and results. Each screen capture should be fully described. The document should be well-organized and include a table of contents, page numbers, figures, and table numbers. The writing style should be paragraph style with bullets used very sparingly to emphasize specific findings. In other words, this should be a professional report and demonstrate mastery of writing.

When researching your security alerts, be sure to document your references using APA style. You should show both before and after fix vulnerability reports. Your final vulnerability report should show zero alerts and vulnerabilities; or fully describe why an alert was not eliminated.

For your deliverables, you should submit a zip file containing your word document (or PDF file) along with the before and after application files. (including SQL and parameter files) If you made changes to your VM environment (apache2.conf, php.ini) you should provide those files also.

Include your full name, class number, professor name and section and date in the document.

Grading Rubric:

Attribute	**Meets**
Analysis, Scans and Mitigation	**60 points** Provides through and detailed analysis on the application including manual code reviews and comments, manual ZAP interceptions and Automated ZAP analysis (40 points) Fully describes and eliminates all alerts and vulnerabilities or fully describes why an alert was not eliminated. (20 points)
Documentation and submission	**40 points** Submits a word or PDF document that includes screen captures demonstrating your process and results. Screen captures are fully described. Clearly documents specifically what you did to mitigate each issue. (20 points)

	Document is well-organized and includes a table of contents, page numbers, figures and table numbers. The writing style should be paragraph style with bullets used very sparingly to emphasize specific findings. Document your references using APA style. (10 points) Includes all before and after application files in winzip format. (SQL and parameter files, apache2.conf, php.ini) (10 points)

A couple of quick hints:

1. Start early as this will take some time.
2. Be sure your application is working properly before you start running the security analysis.
3. Ask questions if you are stuck or have technical issues.
4. Secure HTTP Sessions are key and they should be hard to guess and randomly generated.
5. SQL injection, SQL injection … SQL Injection. Need I say more?
6. I log in, but shouldn't I be able to logout?
7. Feel free to make the resultant application nicer and more user friendly.