Laboratory assignment 3: Web Application Security[*]

For all the assignments in this course, you are expected to work home and only book a lab slot if you are stuck and require a TA's assistance or you are required to demonstrate your solution.

1 Purpose

In this assignment, you will investigate the impact of two of the top ten most critical web application security risks. Unfortunately, there are lots of websites that are vulnerable to those kinds of security issues. This lab will help you learn how to develop more secure web applications.

2 Reporting

To pass this assignment, you need to work through the exercises and write down your answers to the questions. When you are finished, call for your supervisor to discuss your answers.

3 Preparation at home

Read the material as specified bellow. Then, read the rest of this assignment before you start your work.

3.1 Reading

- Section 11.2, Handling Program Input, in course book.
- The OWASP Top 10 Report: `https://owasp.org/Top10/`

3.2 Optional Reading

- SQL Tutorial; Section 3, 5–7: `http://www.sqlcourse.com`
- OWASP ZAP – Getting Started: `https://www.zaproxy.org/getting-started/`

[*] Special thanks to Dr. Ulf Larson for providing us with the web application platform.

- Sqlmap user's manual https://github.com/sqlmapproject/sqlmap/wiki
- Web Security Academy by PortSwigger: https://portswigger.net/web-security
- Web application hacker's handbook by Dafydd Stuttard and Marcus Pinto (Available online at Chalmers' Library).

4 Introduction

In this assignment, you will investigate a web application against two of the top ten most popular security vulnerabilities, namely, SQL Injection, and Cross Site Scripting. You are going to use the web application at http://localhost:49834, which basically provides you a system to login, post comments, purchase products, etc. This application is vulnerable to the two attacks mentioned earlier. You need to work through the tasks and exploit those vulnerabilities and try to breach the system. Moreover, you are asked to fix the source code for the web application to remove these vulnerabilities, such as the SQL-Injection vulnerability.

This assignment is performed in Windows. You will use Firefox and/or Chrome as web-browsers (not Internet Explorer). There is also the possibility to use one instance of Firefox in normal mode and one in private mode. To start Firefox in private mode go to File → New private window or press Control+Shift+P. Alternatively run `firefox.exe --new-instance --ProfileManager` to start Firefox with a new profile..

4.1 Preparing your system

You will need to do some preparation of the system for running this lab. Two plugins need to be installed into Firefox, and the web application needs to be copied to a directory where you have write permission.

Keep in mind that you may need to install the plugins in both instances if you are using different profiles.

4.1.1 Firefox Plugins

Newer versions of Firefox (Firefox version ≥57) do not require installing a plug-in. Check first if the functionality of creating cookies is supported by the browser.

Please start Firefox and install the *Firebug* plugin:

- Firebug version 2.0.x (by Joe Hewitt)

After the plugin is installed, you might need to restart Firefox.

Hint: To install plugins, go to `Firefox` → `Add-ons`. Then search for the plugin. When you find it, just press `Install`. RESTART Firefox when prompted to do so.

4.1.2 Copying the Web Application

All files needed to perform this lab are stored on Canvas. Please download the source code for the web application, named `cs_lab3.zip` on Canvas. Before you start Visual Studio, you need to unzip the file to a destination you can write like %TMP%.

The files must be placed in a place where you and IIS can write to ensure that the web application works.

4.2 Running the Web Application

We will now have a short review of the system setup. To begin the lab you need to run Visual Studio. To do so, go to `Start Menu` → `All Programs` → `Microsoft Visual Studio` and open `Visual Studio`.

Once you run Visual Studio, the version installed in the labs requires that you login in order to unlock its services. You can choose to do so either using your CID with the username "`[cid]@net.chalmers.se`" or using any other Microsoft account.

You will be asked to choose a view, depending on the type of project. We recommend to use the C# perspective.

To open the project, go to `File` → `Open Project/Solution` and then open the file named `myWebApplication` that you just unpacked. You will get a migration report because this is a project from an older Visual Studio version, but you can simply ignore it.

You should now see the Solution Explorer, which includes all the files related to this web application. One of the interesting files you might want to check is SQLaccess.cs.

Now, to debug/run the program press `F5` or `CTRL+F5`. To stop it you can press `SHIFT+F5`. When the web page is loaded you can point your browser to it at `http://localhost:49834`.

In Task 1, you will login to the web application. After you log in, you can do several activities such as viewing the comments that have been posted earlier, posting comments, purchasing products, etc. You need to investigate where and how you can breach the system with the vulnerabilities mentioned, and get control of the application. Afterwards, you are asked to fix the code so that it is impossible to perform SQL Injection against the web application.

5 Exercises

You will now look at two problems with this web application. First, you will investigate SQL injection against the application. Then, you will look at a cross-site scripting problem. In the last part, you will also see some problems with the cookies.

5.1 Exercise 1: SQL Injection

The number-one security risk, among the top ten according to OWASP, are injection vulnerabilities. In the web application, you are now going to investigate the problem of SQL injection.

One of the critical steps in a web application is the process of authentication. Usually all account information is stored in a database, and when the user authenticates, a query is sent to the database for validating the username and password. This web application is also using a database as back-end storage.

Now, please look at the problems around SQL injection in the OWASP Top 10 Report. Then, look at the following tasks.

Make sure the web application has been started in Visual Studio when performing these tasks.

Task 1: Unauthorized Login
In this task, you will look at the problem of *unauthorized access* to the web application. First, start *Firefox* and browse to the web application at `http://localhost:49834`. This login page is vulnerable to SQL injection. Now, keeping in mind the issues described about SQL injection (in OWASP Top 10), try to login to the web application. What did you write to login? What is the problem? Why can you circumvent the authentication system?

Hint: A comment in SQL is written as "--".

Task 2: Disclosure of Information
As you may have noticed while you were trying to invent an SQL string to exploit the SQL injection vulnerability, the web application has the debugging feature activated.[1] If this web application would have been run in a production system, this would have been a great security risk. What are the security implications of this? Did you find something interesting in the debug output?

Task 3: Unauthorized Modification
The debugging feature has leaked information about the web application. This information could be used by an attacker to modify the database. With the information you found in the debugging output, what can you do? Can you create a new account without having administrative privileges? How did you do it?

You may be tempted to drop the table! We would appreciate if you don't.

Hint: What can the `INSERT` SQL statement be used for?

As you see, the problem of SQL injection (together with a misconfigured webserver) can

[1] If you did not get a debug output in the first task, please submit some SQL statements that has the wrong syntax to the login.

lead to a multitude of security-related problems. Some suggestions on how to protect the web application against SQL injections are given in the OWASP Top 10 Report.

Task 4: Preventing SQL Injection
Using the information you gained in the previous tasks, you should now find the corresponding SQL statement in the web application and update the code so that this type of threats is eliminated. What is making this SQL code vulnerable? How would you fix this vulnerability?

Hint: You can use the search function in Visual Studio to help you pin-point certain strings in the program.

5.2 Exercise 2: Cross-Site Scripting

Cross-site scripting is the number seven problem in the Top 10 OWASP list. In this exercise, we will demonstrate the problems around cross-site scripting, and the potential unwanted outcomes of such an attack.

In the web application, a cross-site scripting vulnerability has been introduced into the feedback form. We are now going to demonstrate the vulnerability with this feedback form.

Task 5: Demonstrating Cross-Site Scripting
This demonstration is shown by having one "attacker" and one "normal user".

Now, make sure that the web application is running. While progressing through this task, please have the following questions in mind:

- Can you suggest any security improvements?
- How can these types of vulnerabilities be prevented?

Then, do the following steps:

Step	"attacker"	"normal user"
1	Start `Firefox` and browse to the first page at `http://localhost:49834`.	
2	The vulnerability is located in the feedback form. Press the Feedback link at the bottom right corner. Then, paste the attack code, which you find in the file `Demo\Attack.txt` in your Visual Studio project, into the feedback form. (Include your group number as part of the name you use to track your response.)	
3		Now, Start the `Chrome` web browser and *login* to the web application at `http://localhost:49834`.

Step	"attacker"	"normal user"
4		Go to the page "view comments". Your browser has now been infected by the cross-site script. Please, use notepad to open the file `cookie.txt` inside the folder where you extracted the zip file. What did you find? What is the vulnerability in the web application that could make this possible? (Make sure you do not logout here.)
5	The attacker now has access to the same `cookie.txt` file inside the folder where you extracted the zip file.. Copy the data to your clipboard (`CTRL-C`). Open the cookie manager in `Firefox` from `Tools`→`Web Developer`→`Firebug`→`Open Firebug`, and then go to the tab `cookies`. Then create a new cookie: `Cookies`→`Create Cookie` (If Cookies have not been enabled, enable them). Fill in the form. The name should be `auth` and in the value, you should paste the authentication id you copied from the cookie file. Also, make sure that the hostname is just `localhost` without any port number, and that the session box is ticked. Now browse to `http://localhost:49834/default.aspx`. What happened? What access do you now have?	
6	Now, we are going to upload a file. Click on the Upload File, and upload the file `backdoor.aspx`, which you will also find under the `Demo` directory in your Visual Studio project.	
7	Now, browse to `http://localhost:49834/backdoor.aspx`. Try to run a command, e.g., `dir`. What happened?	
8		Log out from the web application in Chrome.

Note that you do not necessarily need to use two browsers. You could use one browser (Firefox or Google Chrome) and open one window in normal *mode and one in* incognito *mode.*

Based on this demonstration, can you suggest any security improvements? How can these types of vulnerabilities be prevented?

6 Approval

It is now time to call the TA and discuss your answers.