Laboratory Assignment 3: Demonstration of SQL Injection and Cross-Site Scripting

Olof Magnusson and Yu Hsuan Liao

Computer Security EDA263

August 11, 2025

Contents

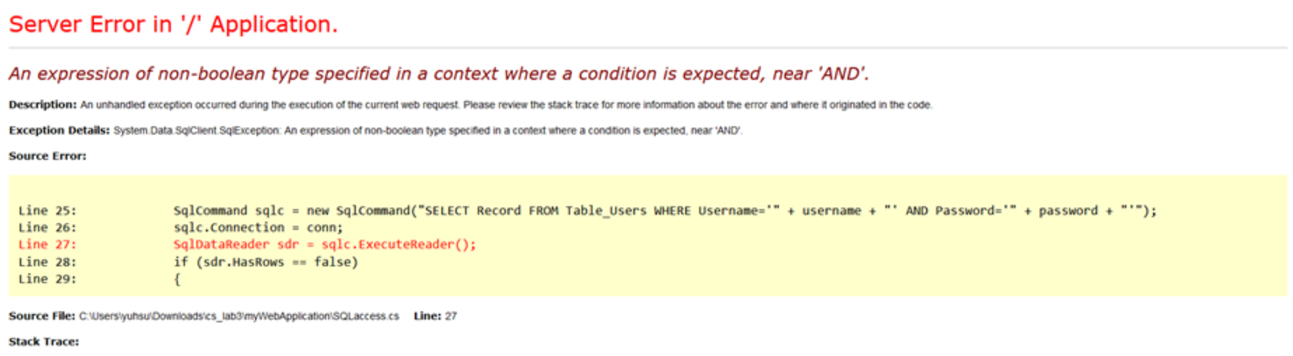
1 Exercise 1: SQL Injection

1. Task 1: Unauthorized Login

   Answer: SELECT UserId, Name, Password FROM Users WHERE 1=1; So this is a big problem, where an attack can get access to all the usernames and passwords in a database by simple inserting 1 = 1 in the input field. We are not associated with the fields (username, password) and will return true. However, most systems drop that specific statement. One could instead, insert 42 = 42 to 'or 1=1. Thus, the rest of the SQL code will be commented (–) and the return value should be true. This leads us to an improper input validation because the system do not sanitize user input.

2. Task 2: Disclosure of Information

   Answer: From the debug information, we know the code and we'll be able to modify and execute SQL injections. When we enter 'OR'= , we get the debug information shown in Figure 1 of An expression of non-boolean type specified in a context where a condition is expected, near 'AND'. And when we input OR'=, it gives out incorrect syntax near "=" which is shown in Figure 2. On line 25, the debugging information shows how the SQL check is going to be done. The attacker could find out this code vulnerability and exploit it in his favor.



Server Error in '/' Application.

An expression of non-boolean type specified in a context where a condition is expected, near 'AND'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: An expression of non-boolean type specified in a context where a condition is expected, near 'AND'.

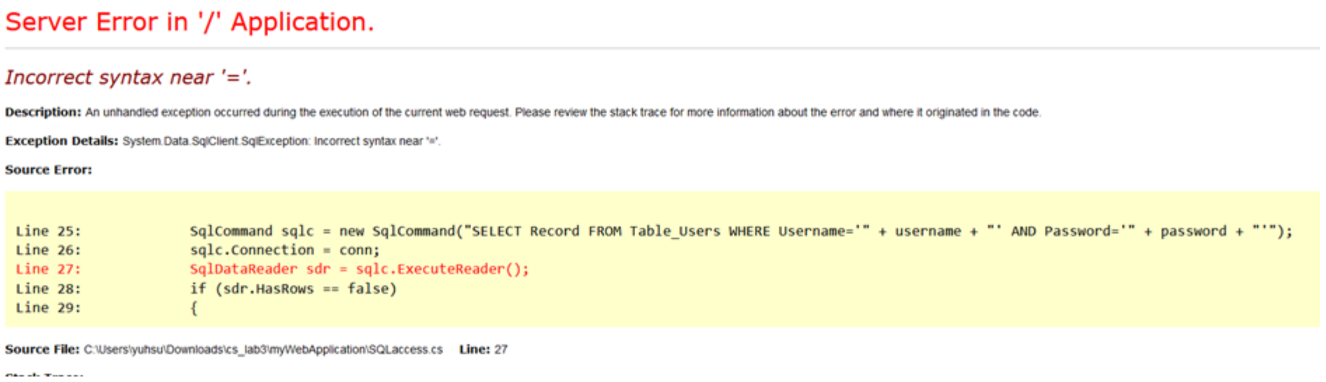
Source Error:

```
Line 25:            SqlCommand sqlc = new SqlCommand("SELECT Record FROM Table_Users WHERE Username='" + username + "' AND Password='" + password + "'");
Line 26:            sqlc.Connection = conn;
Line 27:            SqlDataReader sdr = sqlc.ExecuteReader();
Line 28:            if (sdr.HasRows == false)
Line 29:            {
```

Source File: C:\Users\yuhsu\Downloads\cs_lab3\myWebApplication\SQLaccess.cs **Line:** 27

Stack Trace:

Figure 1: Picture of 'OR'= debug information



```
Server Error in '/' Application.

Incorrect syntax near '='.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Incorrect syntax near '='.

Source Error:

Line 25:            SqlCommand sqlc = new SqlCommand("SELECT Record FROM Table_Users WHERE Username='" + username + "' AND Password='" + password + "'");
Line 26:            sqlc.Connection = conn;
Line 27:            SqlDataReader sdr = sqlc.ExecuteReader();
Line 28:            if (sdr.HasRows == false)
Line 29:            {

Source File: C:\Users\yuhsu\Downloads\cs_lab3\myWebApplication\SQLaccess.cs    Line: 27
```

Figure 2: Picture of OR'= debug information

3. Task 3: : Unauthorized Modification

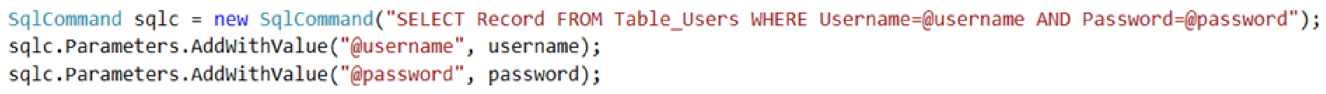
   Answer: Assumed we want to create a new account in the system with the Username: hey, Password: yo. For the Username we input the following:
   $'or\ 1 = 1; INSERT\ INTO\ Table\_Users\ (Username,\ Password)\ VALUES\ ('hey',\ 'yo'); --$

   This means that we are creating an new account in $Table\_Users$ where $(Username, Password)$ are column name and $('hey','yo')$ are the values taking as an input for the execution.

   After creating the account, we can log in as user "hey" to check if we create the account successfully.

4. Task 4: Preventing SQL Injection

   Answer: The previous code will execute anything we input for the Username and insert it into the $Table\_Users$. So, if we type in some SQL code, it will be performed. To fix the problem, we can use SQL parameters for protection. Parameters are represented with a @ marker in the SQL statement. The SQL engine checks each parameter to ensure that it is correct for its column and is treated literally and not as part of the SQL code to be executed. Where Username and password are variables pointing to the associated Username and password. The SQL query is predefined, and the database already knows what this particular query will do. So, in this case, if we are looking for a username of USER OR 1 = 1'– and a blank password, it should return a false result. The revised code is shown in Figure 3



```
SqlCommand sqlc = new SqlCommand("SELECT Record FROM Table_Users WHERE Username=@username AND Password=@password");
sqlc.Parameters.AddWithValue("@username", username);
sqlc.Parameters.AddWithValue("@password", password);
```

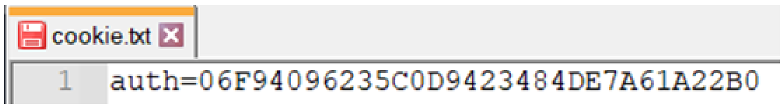
Figure 3: Picture of revised code to prevent SQL Injection

2 Exercise 2: Cross-Site Scripting

1. Task 5: Demonstrating Cross-Site Scripting

(a) Step 4: What did you find? What is the vulnerability in the web application that could make this possible?

Answer: We found out that authorization cookie (= the session ID) was sent to the attackers in file cookie.txt.



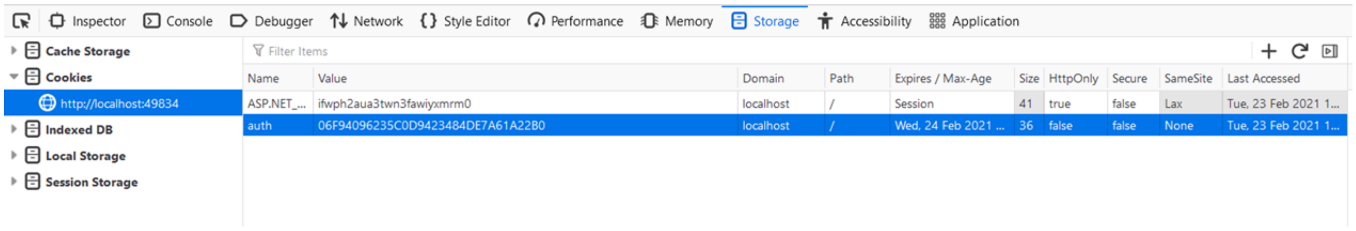
```
auth=06F94096235C0D9423484DE7A61A22B0
```

Figure 4: Picture of content in cookie.txt

It's a stored XSS vulnerability. Stored XSS occurs when a web application gathers input from a user which might be malicious, and then stores that input in a data store for later use. When someone posts a comment, the website will display whatever is entered. If the comment contains HTML tags, they will be added to the webpage's source. So, any script tags will execute whenever the page is loaded.

(b) Step 5: What happened? What access do you now have?

Answer: The attacker hijacks "normal" user's session (shown in Figure 5) and impersonates it. Now, the attacker logins as the "normal" user.

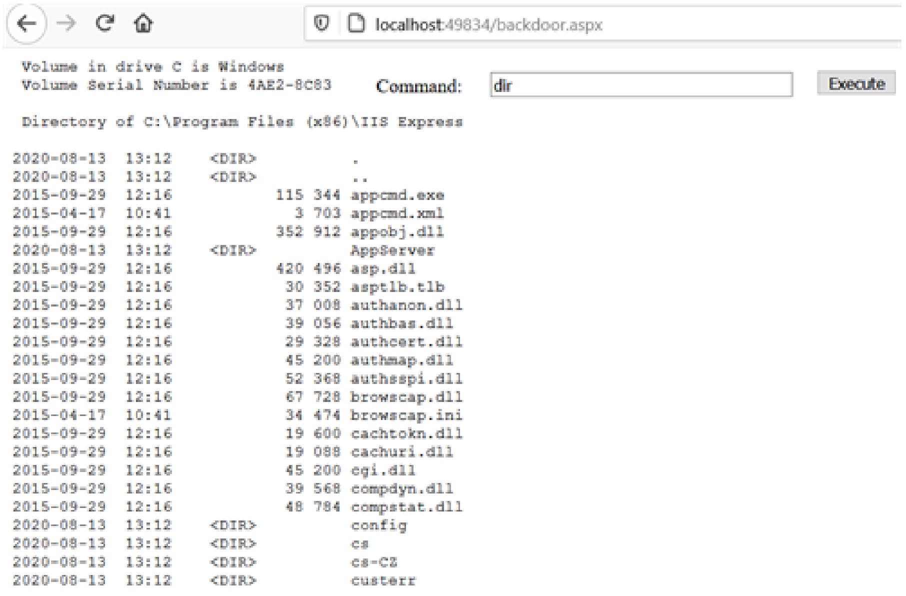


Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
ASP.NET_...	ifwph2aua3twn3fawiyxmrm0	localhost	/	Session	41	true	false	Lax	Tue, 23 Feb 2021 1...
auth	06F94096235C0D9423484DE7A61A22B0	localhost	/	Wed, 24 Feb 2021 ...	36	false	false	None	Tue, 23 Feb 2021 1...

Figure 5: Picture of creating new cookie

(c) Step 7: Now, browse to http://localhost: 49834/backdoor.aspx. Try to run a command, e.g., dir. What happened?

Answer: We run the backdoor program and execute dir command. We get files from C:/Program Files (X86)



```
 Volume in drive C is Windows
 Volume Serial Number is 4AE2-8C83

 Directory of C:\Program Files (x86)\IIS Express

2020-08-13  13:12    <DIR>          .
2020-08-13  13:12    <DIR>          ..
2015-09-29  12:16           115 344 appcmd.exe
2015-04-17  10:41             3 703 appcmd.xml
2015-09-29  12:16           352 912 appobj.dll
2020-08-13  13:12    <DIR>          AppServer
2015-09-29  12:16           420 496 asp.dll
2015-09-29  12:16            30 352 asptlb.tlb
2015-09-29  12:16            37 008 authanon.dll
2015-09-29  12:16            39 056 authbas.dll
2015-09-29  12:16            29 328 authcert.dll
2015-09-29  12:16            45 200 authmap.dll
2015-09-29  12:16            52 368 authsspi.dll
2015-09-29  12:16            67 728 browscap.dll
2015-04-17  10:41            34 474 browscap.ini
2015-09-29  12:16            19 600 cachtokn.dll
2015-09-29  12:16            19 088 cachuri.dll
2015-09-29  12:16            45 200 cgi.dll
2015-09-29  12:16            39 568 compdyn.dll
2015-09-29  12:16            48 784 compstat.dll
2020-08-13  13:12    <DIR>          config
2020-08-13  13:12    <DIR>          cs
2020-08-13  13:12    <DIR>          cs-CZ
2020-08-13  13:12    <DIR>          custerr
```

Figure 6: Picture of information captured from executing backdoor.aspx

2. Suggestions
   Answer:

   - **Forbid input like <script>...</script> tags, <body onLoad=""> and other attributes like: onmouseover, onerror, XSS Attacks may conducted from these tags**
   - **The input could be sanitized which would include proper encoding checking.** Untrusted input should be checked. If the input contains HTMLtags, it should not be added to the webpage's source directly.
     - & —> &amp;
     - < —> &lt;
     - > —> &gt;
     - " —> &quot;
     - ' —> &#x27;
   - **The web server could detect a simultaneous login and invalidate the sessions.**
   - **The web server could detect a simultaneous login from two different IP addresses and invalidate the sessions.**
   - **Using frameworks that automatically escape XSS by design**, such as the latest Ruby on Rails, React JS