Christopher Bazyani

CYB 552

Advanced Hacking Prevention

Lab 5

Attacking a Vulnerable Web Application and Database

**Section 1: Hands-On Demonstration**

Part 1: Connect to the DVWA

On the vWorkstation desktop, double-click the Connections folder.

In the Connections folder, double-click the DVWA shortcut to launch the Damn Vulnerable Web Application in a browser window.

DVWA login screen

DVWA login screen

At the DVWA login page, type the following credentials and click Login to continue.

Username: admin

Password: password

When prompted to store the password, click Not for this site to dismiss the pop-up.

Graphical user interface, text, application, email

Description automatically generated

On the DVWA Navigation menu, click the DVWA Security button to open the DVWA Security page.



On the DVWA Security page, select low from the Script Security drop-down menu and click Submit to change the security level.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Part 2: Conduct an XSS attack

On the DVWA Navigation menu, click the XSS reflected button.



XSS vulnerabilities are generally found in web forms that send and retrieve data to databases via HTML.

In the What’s your name? box, type Simon and click Submit.

The web form will take the name you entered and repeat it back to you in a friendly welcome.

Expected output from XSS test

Graphical user interface, text, application

Description automatically generated

In the What’s your name? box, type <Simon> and click Submit.

Graphical user interface

Description automatically generated with low confidence

Note: The greater and less-than arrows surrounding “Simon” are referred to as scripting tags in HTML. They are what allow you to add scripts to a web page. By entering <Simon> into a form field you are entering a script that contains only the instruction Simon. The result of this submission can give you clues as to the vulnerability of this form.

A well-designed form should not allow any scripts to be run. While there are many ways for a developer to ensure this is true, the most basic method is to prohibit these greater and less than arrows from being submitted. If this site were not vulnerable, submitting <Simon> might return “Hello <Simon>”, “Hello Simon”, or an error about the use of illegal characters. Each of these would demonstrate that the form is not allowing scripts to be run.

The fact that you see a response, even just the word “Hello,” from the form indicates that this form is vulnerable. The web form accepts the unexpected input without issuing an error, even though it fails to return the expected outcome. Now that you have found a possible vulnerability, you will need to test it further.

Results from the XSS reflected vulnerability

In the What’s your name? box, type <script>alert('yourname found proof of a vulnerability');</script>, replacing yourname with your own first name, and click Submit.

Alert script processed by the web form

Note: In order to test the vulnerability, you need to enter a script that does something. The command “alert” is a scripting function that generates a pop-up alert window to the screen. The command is telling the form to run a script that generates a pop-up window with the message within the quotes. The fact that you see this result, proves that the form will allow scripts to run. Since this simple script was processed correctly, you know that there is a good chance that any type of malicious script can be run.

Make a screen capture showing the exposed XSS vulnerability and paste it into your Lab Report file.

Graphical user interface, text, application

Description automatically generated

Click OK to close the alert window.

On the DVWA Navigation menu, click the DVWA Security button, then select high from the Script Security drop-down menu and click Submit to change the security level of the web application to High.

Repeat steps 1-4.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

In your Lab Report file, briefly describe what the new results tell you about how the form now handles cross-site scripting attacks in the High setting.

On the DVWA Navigation menu, click the DVWA Security button, then select low from the Script Security drop-down menu and click Submit to change the security level of the web application back to Low.

Graphical user interface, text, application, email

Description automatically generated

Part 3: Conduct a SQL Injection Attack

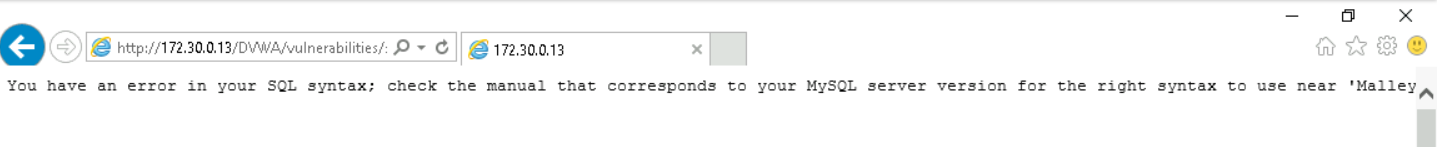
On the DVWA Navigation menu, click the SQL Injection button.



In the User ID box, type O'Malley and click Submit to determine if the web server is vulnerable to SQL injection attacks.

The web form returns a syntax error indicating that there is a problem with the information you typed. The reason for this error is the apostrophe in O’Malley. The script interprets the apostrophe as the end of a command string instead of a part of the name. This leaves “Malley” outside of proper SQL syntax. Often, programmers forget to include script handling for special characters like apostrophes in their data input forms. This type of error can make an application vulnerable to SQL injection.

SQL error



On the browser's navigation bar, click the Back button to return to the SQL Injection form.

In the User ID box, type yourname, replacing yourname with your own last name, and click Submit.

In most cases, the Web form will return no results which likely indicates that a name is not the correct format for the user ID field. If you received an error, as in Step 2, click the browser’s Back button to return to the SQL Injection form.

In the User ID box, type 1 and click Submit.

Graphical user interface, application

Description automatically generated

The form takes your input, 1, and retrieves the first name and surname of the user associated with that user ID. Now you know the format of the User ID field. In addition, from these results, you can surmise that the SQL query that is being run to retrieve the data from this input form is: SELECT \* FROM the database WHERE user\_id = '1' which roughly translates to "retrieve all fields from the database where the User ID is 1".

In the User ID box, type x'='x and click Submit.

A picture containing text

Description automatically generated

This script is commonly used is SQL injection attacks. Instead of interpreting this script as a call for the user “x”, this script returns every user in the application’s database.

Results from the SQL injection test

In the User ID box, type a' ORDER BY 1;# and click Submit.

**Result: No output was displayed**

Another SQL injection method is to enter SQL and use the presence or lack of errors to determine vulnerabilities. Review the output of this script. Here, you are trying to order the output by the first (1) column, or field. If there is no error, it means there is a first column. You now know something about the structure of the database: there is at least one column.

In the User ID box, type a' ORDER BY 2;# and click Submit.

**Result: No output was displayed**

Review the output. If there is no error, you know there are at least two columns. If you see an error statement, it means there isn’t a second column; click the browser’s Back button and proceed to step 10.

In the User ID box, type a' ORDER BY 3;# and click Submit.

Graphical user interface, text

Description automatically generated with medium confidence

Review the output. If there is no error, you know there are at least three columns. If you see an error statement, it means there isn’t a third column; click the browser’s Back button and proceed to step 10.

In your Lab Report file, briefly describe the results of steps 7-9

In the User ID box, type a' OR firstname IS NULL;# and click Submit.

Graphical user interface, text, application

Description automatically generated

Another common SQL injection method is to guess the column names using common spellings. The lack of an error message to this type of query would indicate that you guessed the correct spelling. An error message indicates that firstname is not the correct spelling for the column.

Results from the “firstname” test

On the browser's navigation bar, click the Back button to return to the SQL injection form.

In the User ID box, type a' OR first\_name IS NULL;# and click Submit to try another common spelling for the first name field.

The lack of an error message for this statement indicates that you have guessed the field name correctly as first\_name.

In the User ID box, type a' OR database() LIKE 'DB';# and click Submit.

This script searches for a possible hit on the database’s characters.

In the User ID box, type a' OR database() LIKE 'd%';# and click Submit.

A picture containing text

Description automatically generated

Like the previous script, this one searches for a possible hit on the database’s characters, but the percent sign (%) will split the fields.

In the User ID box, type a' UNION SELECT table\_schema, table\_name FROM information\_Schema.tables;# and click Submit.

A picture containing text

Description automatically generated

You have just used the form to submit another common SQL injection statement against a database structure called information\_schema. The information\_schema is one of a small handful of default schemata in the MySQL database software, so it's a logical and easy place for SQL injection hackers to start. This script will return all of the table and column names in the database so hackers can learn much about the structure and content of the tables in the website's database.

In the User ID box, type a' UNION ALL SELECT 1, @@version;# and click Submit.

Graphical user interface, application

Description automatically generated

This script will return information about the version of SQL being used on the server.

In the User ID box, type a' UNION ALL SELECT system\_user(), user();# and click Submit.

This script will return information about the user name that you are currently using to make queries on the server.

Make a screen capture showing the user account information and paste it into your Lab Report file.

Diagram

Description automatically generated with low confidence

In the User ID box, type a' UNION ALL SELECT user, password FROM mysql.user;# priv;# ' and click Submit.

This script will display a hash for the user to the backend database.

Make a screen capture showing the hash information and paste it into your Lab Report file.

Text

Description automatically generated with low confidence

In the User ID box, type a' UNION SELECT 'test', '123' INTO OUTFILE 'yourname\_S1.txt, replacing yourname with your own name, and click Submit.

This script should insert the phrases test and 123 into a new file called yourname\_S1.txt. You will view the contents of that file in the next part of this lab.

Close the browser window.

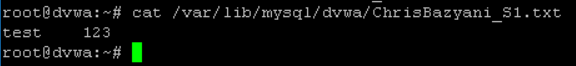
Part 4: Retrieve the yourname\_S1.txt File

In the Connections folder, double-click the PuTTy-DVWA shortcut to open a PuTTY session to the server hosting the DVWA application.

You will be able to use this PuTTY window to view the changes taking place on the DVWA server.

At the command prompt, type cat /var/lib/mysql/dvwa/yourname\_S1.txt, replacing yourname with your own name, and press Enter to display the contents of the file you created.

Make a screen capture showing the contents of the yourname\_S1.txt file and paste it into your Lab Report file.



Close the PuTTY window.

Note: Together with the information you gathered in earlier tests, the ability to write to a file indicates that you have found a user with elevated permissions, user IDs, passwords, table, and column information – in other words, an injectable database.

**Section 2: Applied Learning**

Part 1: Connect to the DVWA

From the Connections folder, double-click the DVWA shortcut and log in using the following credentials:

Username: admin

Password: password

Graphical user interface, text, application, email, website

Description automatically generated

Set the DVWA script security level to low.

Graphical user interface, text, application

Description automatically generated

Part 2: Conduct an XSS attack

Navigate to the XSS reflected page.

XSS vulnerabilities are generally found in web forms that send and retrieve data to databases via HTML.

In the What’s your name? box, submit yourname, replacing yourname with your own first name.

Graphical user interface, text, application

Description automatically generated

The web form will take the name you entered and repeat it back to you in a friendly welcome.

In the What’s your name? box, submit <yourname>, replacing yourname with your own first name.

Graphical user interface, text

Description automatically generated

Note: The greater and less-than arrows surrounding your name are referred to as scripting tags in HTML. They are what allow you to add scripts to a web page. By entering <yourname> into a form field, you are entering a script that contains your name as an instruction. The result of this submission can give you clues as to the vulnerability of this form.

A well-designed form should not allow any scripts to be run. While there are many ways for a developer to ensure this is true, the most basic method is to prohibit these greater and less than arrows from being submitted. If this site were not vulnerable, submitting <Simon> might return “Hello <Simon>”, “Hello Simon”, or an error about the use of illegal characters. Each of these would demonstrate that the form is not allowing scripts to be run.

The fact that you see a response, even just the word “Hello,” from the form indicates that this form is vulnerable. The web form does not complain and it fails to return the expected outcome. Now that you have found a possible vulnerability, you will need to test it further.

In the What’s your name? box, submit <script>alert('Hi, yourname');</script>, replacing yourname with your own first name.

Note: In order to test the vulnerability, you need to enter a script that does something. The command “alert” is a scripting function that generates a pop-up alert window to the screen. The command is telling the form to run a script that generates a pop-up window with the message “Here is proof of a vulnerability”. The fact that you see this result, proves that the form will allow scripts to run. Since this simple script was processed correctly, you know that there is a good chance that any type of malicious script can be run.

Make a screen capture showing the exposed XSS vulnerability and paste it into your Lab Report file.

Graphical user interface, application

Description automatically generated

Close the alert window.

Set the DVWA script security level to high.

Repeat steps 1-4.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Set the DVWA script security level to low.

Graphical user interface, text, application

Description automatically generated

Navigate to the Command Execution page.

In the text box, submit 172.30.0.13 to ping the DVWA server.

The Ping results are displayed on the screen.

A picture containing text

Description automatically generated

Note: In the next steps, you will perform several tasks using the Command Execution vulnerabilities.

In the text box, submit cat /etc/passwd to execute the command that will display the passwd file.

The command does not produce the expected results.

In the text box, submit 172.30.0.13; cat /etc/passwd to add the command that will display the passwd file at the end of the Ping request.

Text, letter

Description automatically generated

The result of this command (the display of the contents of the passwd file) reveals a vulnerability of the code in DVWA. Sites that enable Unix-style utilities may be vulnerable to this type of exploitation if not properly secured or coded correctly.

Minimize the DVWA window, then double-click the PuTTy-DVWA shortcut to open a remote connection to the DVWA server.

At the command prompt, execute cat /var/www/DVWA/vulnerabilities/exec/source/low.php to execute the command that will display the source code for the DVWA’s low security setting.

Text

Description automatically generated

Note: Reading the low.php file, you will see that the code does not check to see if $target matches the IP Address. The code enables users to issue multiple commands by using the semicolon to separate the injection string(s).

The expected action (normal) of the Command Execution page:

The user enters an IP address into the text box.

The system executes a Ping Request on the IP address.

The system displays the Ping Reply below the text box.

The unintended action (malicious) of the Command Execution page:

The user enters an IP address, a semicolon, and then another command into the text box.

The system executes a Ping Request on the IP address, but also executes the second command.

The system displays the Ping Reply, and the result of the second command below the text box.

In the next steps, you will use exploit this vulnerability with a number of new injection commands.

Restore the DVWA window, then submit 172.30.0.13; cat /etc/passwd | tee /tmp/passwd on the Command Execution page to copy the passwd file to the tmp directory, which typically has a lower security setting, if any at all.

Note: The tee command is a command-line interpreter (shell) which will read standard input and writes it to an output file, effectively duplicating the file.

Restore the PuTTY window, then execute ls /tmp at the command prompt to list the tmp directory and confirm that the file was copied.

Graphical user interface

Description automatically generated with medium confidence

Part 3: Conduct a SQL Injection Attack

Restore the DVWA window and click the SQL Injection button.

In the User ID text box, submit 2 to display the first and last name of the user with the ID 2.

Graphical user interface, text, application

Description automatically generated

This is the expected action for this web form. The form takes your input, 2, and retrieves the first name and surname of the user associated with that user ID. Now, you know the format of the User ID field. In addition, from these results, you can surmise that the SQL query that is being run to retrieve the data from this input form is: SELECT \* FROM the database WHERE user\_id = '2' which roughly translates to "retrieve all fields from the database where the User ID is 2."

In the User ID box, submit %' OR '1'='1 to display the first and last names for all users in the database.

Text

Description automatically generated

This a very popular SQL Injection command. This type of injection attack is a string value that returns a statement result. In normal SQL programming language (as opposed to the MySQL variant, which is used in this lab), the form of this command would be: SELECT \* FROM users WHERE (first\_name, last\_name) = %OR '1'='1 ; which translates to "show me everything that is either equal to the wild card or not equal to the wild card." Depending on which type of SQL database you are attacking, there are a wide range of similar queries:

x'='x

' or 'x'='x

' or 1=1--

In the User ID box, submit a' ORDER BY 1;#.

Another SQL injection method is to enter SQL and use the presence or lack of errors to determine vulnerabilities. Review the output of this script. Here, you are trying to order the output by the first (1) column, or field. If there is no error it means there is a first column. You now know something about the structure of the database: there is at least one column.

In the User ID box, submit a' ORDER BY 2;#.

Review the output. If there is no error, you know there are at least two columns. If you see an error statement, it means there isn’t a second column; click the browser’s Back button and proceed to step 7.

In the User ID box, submit a' ORDER BY 3;#.

Graphical user interface, text, application

Description automatically generated

Review the output. If there is no error, you know there are at least three columns. If you see an error statement, it means there isn’t a third column. Click the browser’s Back button and proceed to step 7.

In the User ID box, submit a' OR firstname IS NULL;#.

Another common SQL injection method is to guess the column names using common spellings. The lack of an error message to this type of query would indicate that you guessed the correct spelling. An error message indicates that firstname is not the correct spelling for the column.

Click the browser’s Back button.

In the User ID box, submit a' OR first\_name IS NULL;# to try another common spelling for the first name field.

The lack of an error message for this statement indicates that you have guessed the field name correctly as first\_name.

In the User ID box, submit a' OR database() LIKE 'DB';#.

This script searches for a possible hit on the database’s characters.

In the User ID box, submit a' OR database() LIKE 'd%';#.

Text

Description automatically generated with low confidence

Like the previous script, this one searches for a possible hit on the database’s characters, but the percent sign (%) will split the fields.

In the User ID box, submit a' UNION SELECT table\_schema, table\_name FROM information\_Schema.tables;#.

A picture containing text

Description automatically generated

You have just used the form to submit another common SQL injection statement against a database structure called information\_schema. The information\_schema is one of a small handful of default schemata in the MySQL database software, so it's a logical and easy place for SQL injection hackers to start. This script will return all of the table and column names in the database so hackers can learn much about the structure and content of the tables in the website's database.

In the User ID box, submit a' UNION ALL SELECT 1, @@version;# to return information about the version of SQL being used on the server.

Graphical user interface, text, application, email

Description automatically generated

In the User ID box, submit a' UNION ALL SELECT system\_user(), user();# to return information about the user name that you are currently using to make queries on the server.

Make a screen capture showing the user information and paste it into your Lab Report file.

Graphical user interface, text, application

Description automatically generated

In the User ID box, submit a' UNION ALL SELECT user, password FROM mysql.user;# priv;# ' to return a hash for the user to the backend database.

Make a screen capture showing the hash information and paste it into your Lab Report file.

A picture containing text

Description automatically generated

In the User ID box, submit a' UNION SELECT 'successful', 'hack' INTO OUTFILE 'yourname\_S2.txt, replacing yourname with your own name.

This script should insert the phrase successful hack into a new text file with your own name as the file name. You will view the contents of that file in the next part of this lab.

Close the DVWA window.

Part 4: Retrieve the yourname\_S2.txt File

Restore the PuTTY window.

At the command prompt, execute cat /var/lib/mysql/dvwa/yourname\_S2.txt, replacing yourname with your own name, to display the contents of the file you created.

Make a screen capture showing the contents of the yourname\_S2.txt file and paste it into your Lab Report file.

Text

Description automatically generated

Close the PuTTY window.

Note: Together with the information you gathered in earlier tests, the ability to write to a file indicates that you have found a user with elevated permissions, user IDs, passwords, table, and column information – in other words, an injectable database.