SAD EXPERIMENT NO: 7

Aim: Implementation of Data Validation Experiment(SQL Injection)

To do:

1. 2What is SQL injection?
2. How and why SQL Injection is performed?
3. How to prevent SQL Injection?
4. SQL Injection Process
    a. SQL Injection vulnerability allowing login bypass
    b. SQL Injection vulnerability allowing retrieval of hidden data

Theory:

1. **What is SQL injection?**

SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input.
SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database. It creates a SELECT statement by adding a variable (txtUserId) to a select string.

An SQL injection attack consists of an insertion or injection of a SQL query via the input data from the client to the application. SQL commands are injected into data-plane input that affect the execution of predefined SQL commands. A successful SQL injection exploit can read sensitive data from the database, modify database data (viz., insert, update, or delete), execute administrative operations on the database, recover the content of a file present in the database management system, and even issue commands to the operating system in some instances. If a web application or website uses SQL databases like Oracle, SQL Server, or MySQL, it is vulnerable to an SQL injection attack. Hackers use SQL injection attacks to access sensitive business or personally identifiable information (PII), which ultimately increases sensitive data exposure.

2. **How and why SQL Injection is performed?**

➢ **Why SQL Injection is performed**

To perform an SQL injection attack, an attacker must locate a vulnerable input in a web application or webpage. When an application or webpage contains a SQL injection vulnerability, it uses user input in the form of an SQL query directly. The hacker can execute a specifically crafted SQL command as a malicious cyber intrusion. Then, leveraging malicious code, a hacker can acquire a response that provides a clear idea about the database construction and thereby access to all the information in the database.

SQL serves as the way of communication to the database. SQL statements are used to retrieve and update data in the database. Attackers use malicious SQL statements in the input box, and in response, the database presents sensitive information. This exploit of security aims at gaining access to the unauthorised data of a website or application. Several websites and web applications store data in SQL databases.

➢ **How an SQL Injection Attack Is Performed**

SQL injection is performed by using a structured query that instigates the desired response. The response is essential for the attacker to understand the database architecture and to access the secured information of the application. An attacker may perform SQL injection with the following approaches:

- SQL statement that is always true: A hacker executes an SQL injection with an SQL statement that is always true. For instance, 1=1; instead of just entering the "wrong" input, the hacker uses a statement that will always be true. Entering "100 OR 1=1" in the query input box will return a response with the details of a table. "OR ""=" This SQL injection approach is similar to the above. A bad actor needs to enter "OR ""=" into the query input box. These two signs serve as the malicious code to break into the application. Consider the following example. An attacker seeks to retrieve user data from an application and can simply type "OR=" in the user ID or password. As this SQL statement is valid and true, it will return the data of the user table in the database.

- Batched SQL injection: Batched SQL injection comprises a set of SQL statements separated by semicolons. The only thing that can make this approach successful is if the SQL statements are true and valid—that is, the statement after the semicolon needs to be true. For example, if the hacker enters "105; DROP TABLE Supplier," the SQL statement after the semicolon will delete the supplier table from the application database.

- **Types of SQL Injection:** SQL injection can be categorized into three categories: in-band, inferential, and out-of-band.

➢ **In-band SQL injection:** In-band SQL injection is the most frequent and commonly used SQL injection attack. The transfer of data used in in-band attacks can either be done through error messages on the web or by using the UNION operator in SQL statements. There are two types of in-band SQL injection: union-based and error-based SQL injection.

➢ **Union-based SQL injection:** When an application is vulnerable to SQL injection and the application's responses return the results for a query, attackers use the UNION keyword to retrieve data from other tables of the application database.

➢ **Error-based SQL injection:** The error-based SQL injection technique relies on error messages thrown by the application database servers. Here, attackers use the error message information to determine the entities of the database.

➢ **Inferential SQL injection:** Inferential SQL injection is also known as a blind SQL injection attack. In a blind SQL injection attack, after sending a data payload, the attacker observes the behavior and responses to determine the data structure of the database.

3. How to prevent SQL Injection?

The only sure way to prevent SQL Injection attacks is input validation and parameterized queries including prepared statements. The application code should never use the input directly. The developer must sanitise all input, not only web form inputs such as login forms. They must remove potential malicious code elements such as single quotes. It is also a good idea to turn off the visibility of database errors on your production sites. Database errors can be used with SQL Injection to gain information about your database.

Step 1: Train and maintain awareness
Step 2: Don't trust any user input
Step 3: Use whitelists, not blacklists
Step 4: Adopt the latest technologies
Step 5: Employ verified mechanisms
Step 6: Scan regularly (with Acunetix)

Most instances of SQL injection can be prevented by using parameterized queries (also known as prepared statements) instead of string concatenation within the query.

The following code is vulnerable to SQL injection because the user input is concatenated directly into the query:

```
String query = "SELECT * FROM products WHERE category = '"+
input + "'";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
```

This code can be easily rewritten in a way that prevents the user input from interfering with the query structure:

```
PreparedStatement statement =
connection.prepareStatement("SELECT * FROM products WHERE
category = ?");
statement.setString(1, input);
ResultSet resultSet = statement.executeQuery();
```

4. SQL Injection Process:

a. SQL injection vulnerability allowing login bypass

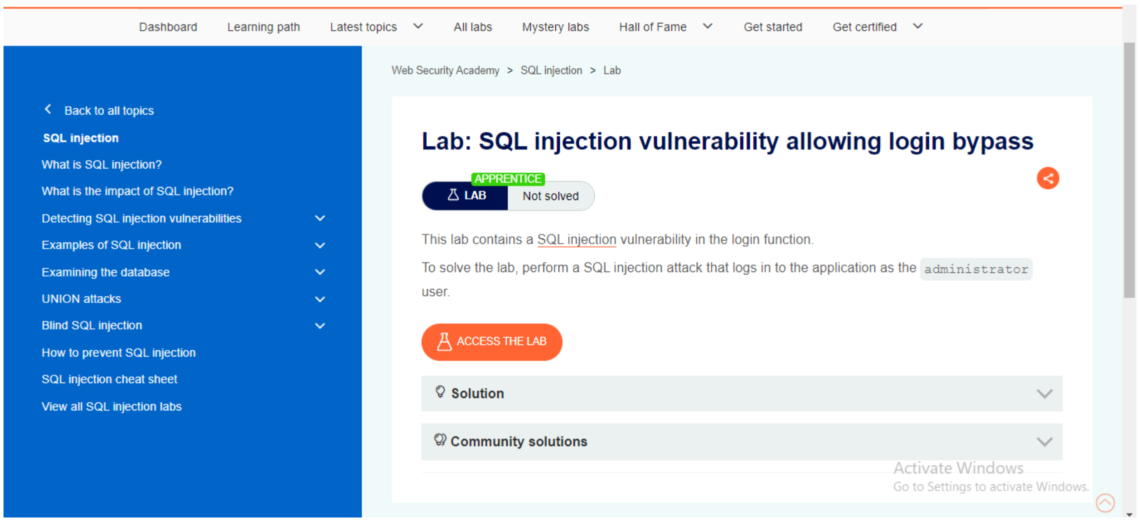
In cases where the results of an SQL query are returned within the application's responses, an attacker can leverage an SQL injection vulnerability to retrieve data from other tables within the database. Here, an attacker can log in as any user without a password simply by using the SQL comment sequence -- to remove the password check from the WHERE clause of the query. For example, submitting the username administrator'-- and a blank password results in the following query:

```
SELECT * FROM users WHERE username = 'administrator'--' AND password = ''
```

This query returns the user whose username is administrator and successfully logs the attacker in as that user. Consider a shopping application that displays products in different categories.

Implementation:

1. Go to Burp Labs Bypass: [Link]
2. Click on Access Lab.

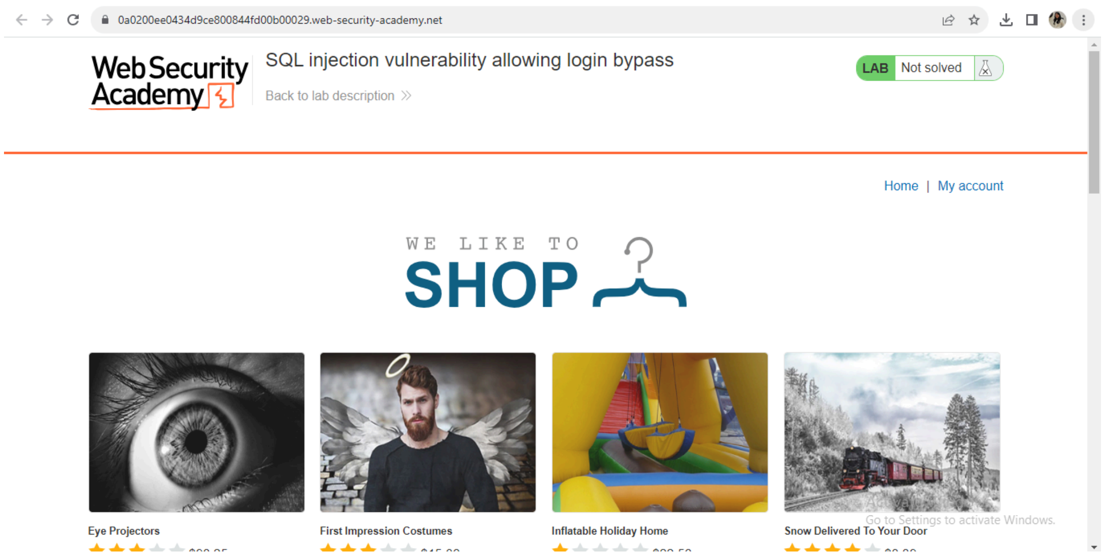


Lab: SQL injection vulnerability allowing login bypass

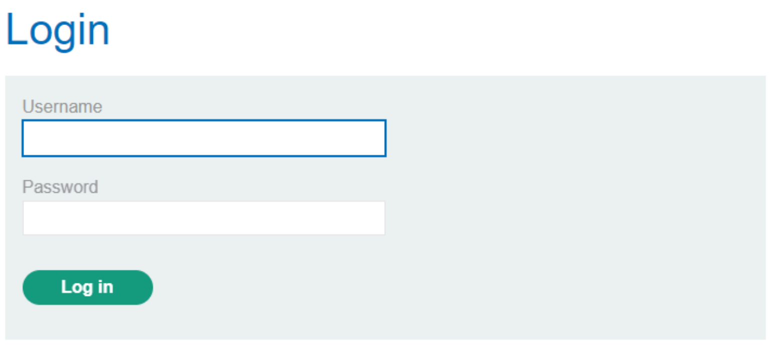
This lab contains a SQL injection vulnerability in the login function.

To solve the lab, perform a SQL injection attack that logs in to the application as the `administrator` user.

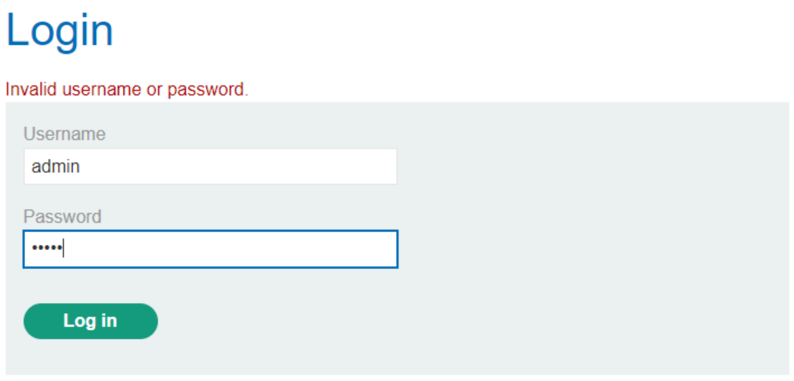
3. Here we can see that there is a My Account button in which we will enter the username and password for login as admin.


0a0200ee0434d9ce800844fd00b00029.web-security-academy.net
Web Security Academy
SQL injection vulnerability allowing login bypass
Back to lab description
LAB
Not solved
Home | My account
WE LIKE TO
SHOP
Eye Projectors
First Impression Costumes
Inflatable Holiday Home
Snow Delivered To Your Door
Go to Settings to activate Windows.

4. After clicking to My Account the screen appears for username and password.



5. After that we will enter username as admin and password as admin and try to get into the website. After clicking the LOGIN button an error will come and show as Invalid username and password.

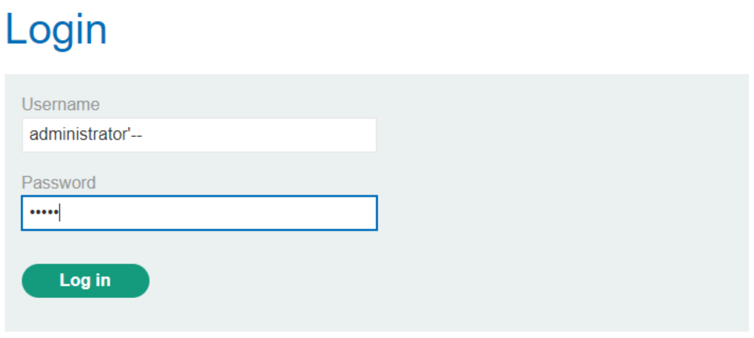


Invalid username or password.

admin

•••••

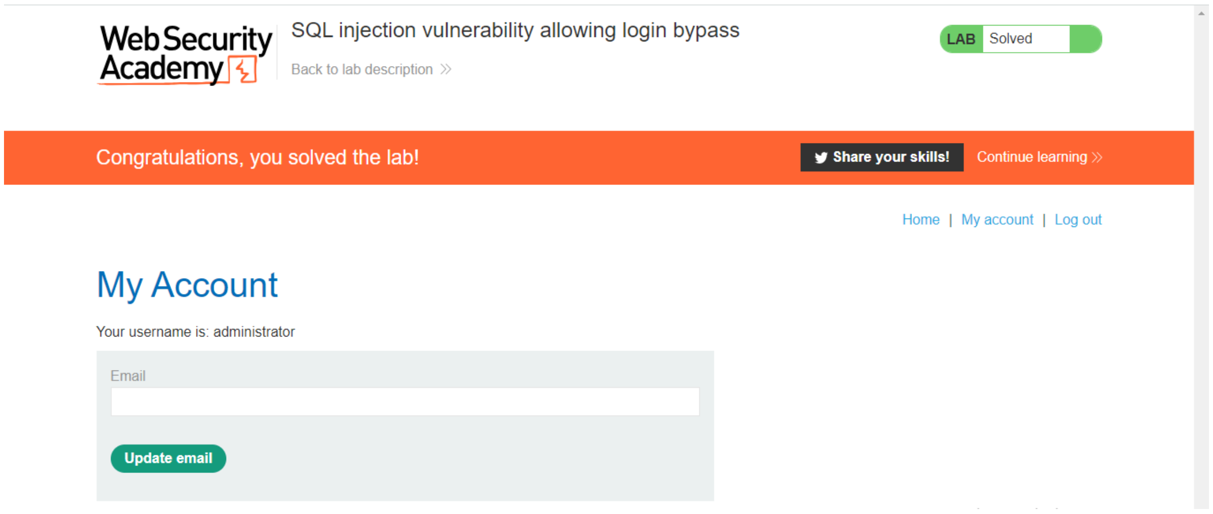
6. After this we will perform an SQL injection attack that logs in to the application as the administrator user.



administrator'--

•••••

7. After then we can see that we got success in LOGIN as Administrator.


SQL injection vulnerability allowing login bypass

Back to lab description »

LAB Solved

Congratulations, you solved the lab!

Share your skills! Continue learning »

Home | My account | Log out

My Account

Your username is: administrator

Email

Update email

b. SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

Consider a shopping application that displays products in different categories. This causes the application to make an SQL query to retrieve details of the relevant products from the database:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

This SQL query asks the database to return all details (*) from the products table where the category is Gifts and released is 1. The restriction released = 1 is being used to hide products that are not released. For unreleased products, presumably released = 0.

The key thing here is that the double-dash sequence -- is a comment indicator in SQL, and means that the rest of the query is interpreted as a comment. This effectively removes the remainder of the query, so it no longer includes AND released = 1. This means that all products are displayed, including unreleased products.

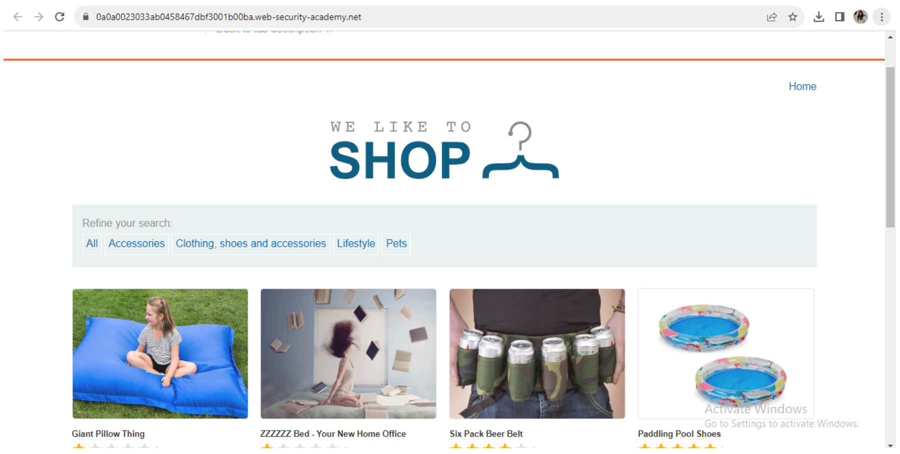
Going further, an attacker can cause the application to display all the products in any category, including categories that they don't know about. This results in the SQL query:

```
SELECT * FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1
```

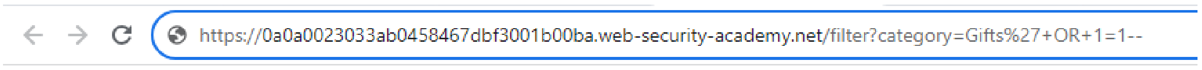
The modified query will return all items where either the category is Gifts, or 1 is equal to 1. Since 1=1 is always true, the query will return all items.

Implementation:

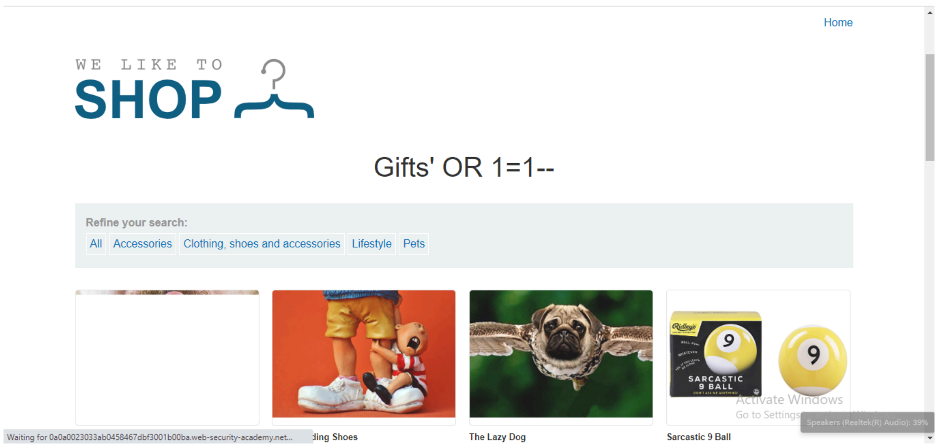
1. Use Burp Suite to intercept and modify the request that sets the product category filter.


0a0a0023033ab0458467dbf3001b00ba.web-security-academy.net
Home
WE LIKE TO SHOP
Refine your search:
All
Accessories
Clothing, shoes and accessories
Lifestyle
Pets
Giant Pillow Thing
ZZZZZZ Bed - Your New Home Office
Six Pack Beer Belt
Paddling Pool Shoes

2. Modify the category parameter, giving it the value filter?category=Gifts%27+OR+1=1--


https://0a0a0023033ab0458467dbf3001b00ba.web-security-academy.net/filter?category=Gifts%27+OR+1=1--

3. Submit the request, and verify that the response now contains additional items.



Conclusion: The SQL Injection experiment underscores the significant threat posed by exploiting input vulnerabilities in web apps. Attackers can manipulate databases by injecting malicious code through input fields. The experiment demonstrated attack methods, such as injecting true statements and leveraging UNION operations, leading to unauthorised data access. The experiment practically exposed login bypass and hidden data retrieval vulnerabilities. In summary, SQL injection remains a critical concern. Understanding and preventing such attacks are crucial for safeguarding applications and data.