**Analyzing Types of Web Application Attacks**

The purpose of this lab is to provide hands-on experience with web application security using vulnerable web applications such as WebGoat and DVWA. The lab involves accessing WebGoat on port 80 at http://127.0.0.1/Webgoat/attack and exploiting a String SQL Injection vulnerability by submitting the input "Smith" and then "1 or 1=1--" to dump all user information in Stage 1. Tamper Data is used to modify the input to perform a successful SQL injection attack and access employees' personal information.

In the second part of the lab, the Kali machine is used to set up mysql and apache2 and access DVWA Security at http://127.0.0.1/dvwa/login.php. A SQL Injection vulnerability is exploited by submitting the input "always true" and then "' or 'x'='x" to retrieve the version of mysql. The input "' or 1=1 union select null, version() #" is used to access the schema table and retrieve user credentials.

In the third part of the lab, an XSS vulnerability is exploited by submitting the input "<script>alert("Exploiting XSS Vulnerability Test")</script>" to embed a website and steal the cookie session ID. This lab provides practical experience in identifying and exploiting web application vulnerabilities, which is crucial for understanding and improving web application security.

1. **Define WebGoat and DVWA. What are these applications used for?**

**WebGoat and DVWA** are both vulnerable web application environments used for educational purposes to teach web application security.

**WebGoat** is a deliberately insecure web application created by OWASP (Open Web Application Security Project) designed to teach developers and security professionals about common web application vulnerabilities such as SQL injection, cross-site scripting (XSS), and buffer overflows. It provides a hands-on experience for users to exploit and fix vulnerabilities in a safe environment.

**DVWA (Damn Vulnerable Web Application)** is another intentionally vulnerable web application created for security professionals and enthusiasts to practice their skills in a legal and safe environment. It contains various types of security vulnerabilities such as SQL injection, cross-site scripting, and file inclusion vulnerabilities that can be used to test security scanners, crawlers, and vulnerability assessment tools.

**Both WebGoat and DVWA** can be used as training tools for developers and security professionals to learn about web application security vulnerabilities and how to mitigate them. These applications help users to develop the necessary skills and knowledge to identify, exploit and remediate common web application vulnerabilities, which in turn can help to improve the overall security of web applications.

1. **Define SQL injection and Cross-Site Scripting (XSS) attacks.**

**SQL injection and Cross-Site Scripting (XSS)** attacks are two common types of web application security vulnerabilities that can be exploited by attackers to gain unauthorized access to sensitive information or perform malicious actions.

**SQL injection** is a type of vulnerability that occurs when an attacker injects malicious SQL code into an input field on a web application that is not properly validated. This can result in the attacker being able to access, modify, or delete sensitive information stored in a database, or execute commands on the database server.

**Cross-Site Scripting (XSS)** is another type of vulnerability that occurs when an attacker injects malicious scripts into a web page that is viewed by other users. This can allow the attacker to steal session cookies, login credentials, or other sensitive information from the user's browser, or perform actions on behalf of the user.

1. **What is a popular SQL injection command/technique to potentially leak all user information stored in a databases table?**

A popular SQL injection command/technique to potentially leak all user information stored in a database table is known as "union-based SQL injection."

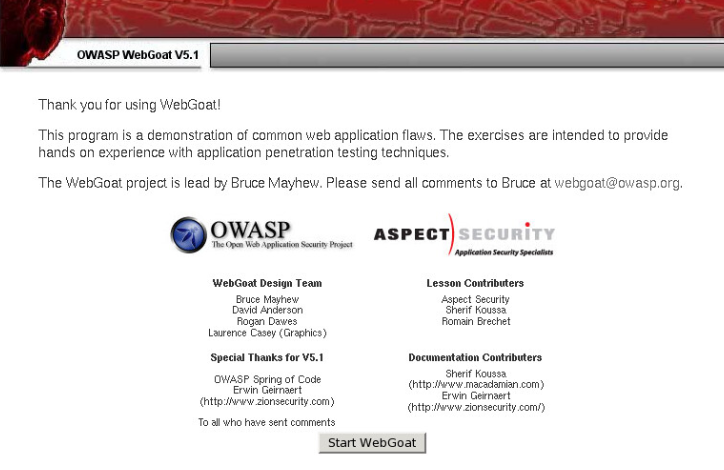
Union-based SQL injection involves a "union select" statement to the end of a maliciously crafted input string, allowing the attacker to retrieve data from another table within the database.

For example, consider a web application that uses a SQL query to retrieve user data based on an input parameter, such as a username. An attacker could submit a malicious input string containing the following code:

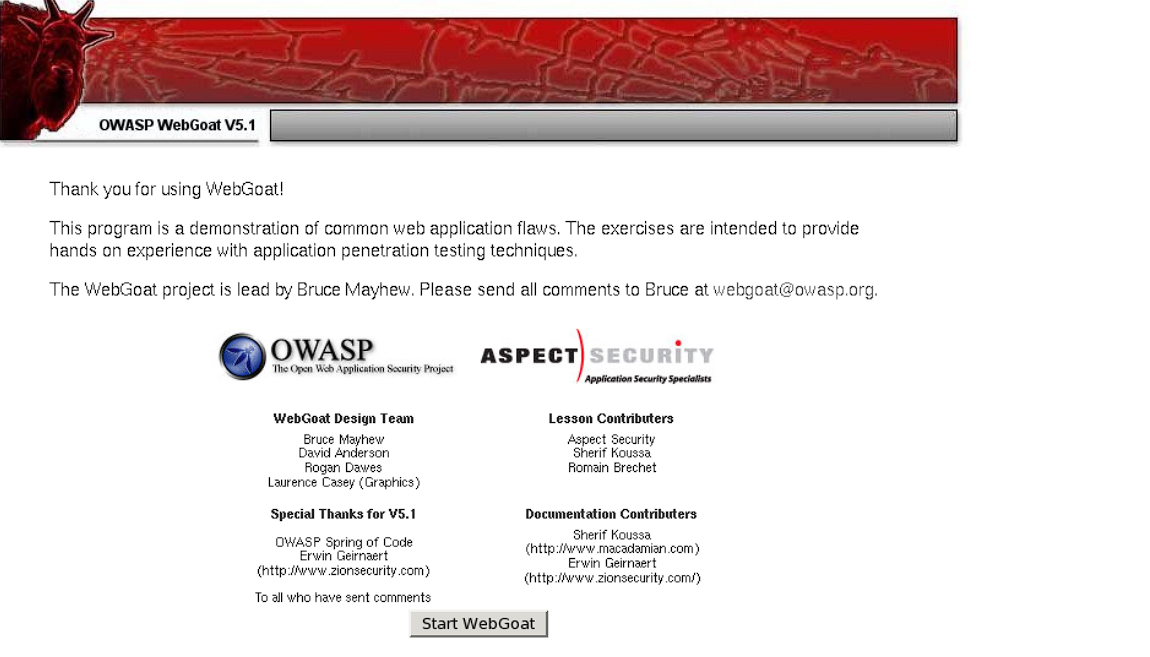
**' union select 1, username, password from users --**

This would modify the original SQL query to include a "union select" statement that retrieves the username and password fields from the "users" table, along with a dummy value for the first column. The double dash "--" is used to comment out the rest of the original SQL query.

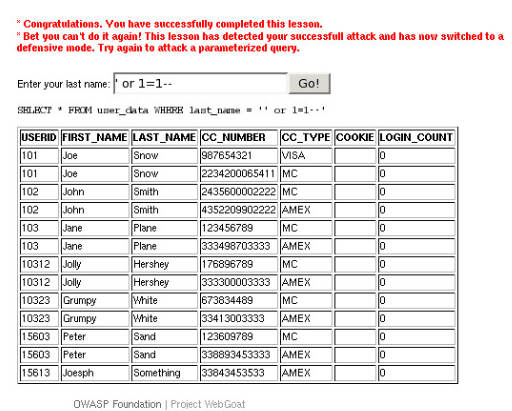
On page 7, import the snapshot equivalent to Figure 10. See the example below:

****

1. **Import your snapshot (similar to example above) here:**

****

On page 10, import the snapshot equivalent to Figure 17. See the example below:

****

1. **Import your snapshot (similar to example above) here:**

Table

Description automatically generated

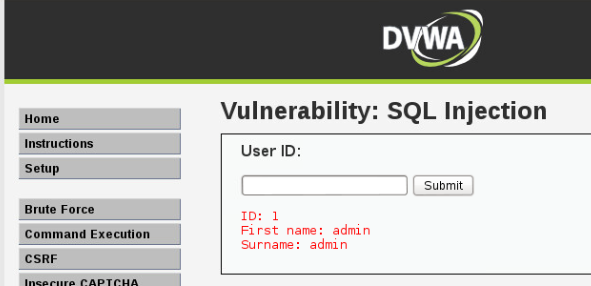
On page 15, import the snapshot equivalent to Figure 26. See the example below:

****

1. **Import your snapshot (similar to example above) here:**

****

On page 19, import the snapshot equivalent to Figure 17. See the example below:

****

1. **Import your snapshot (similar to example above) here:**

**Graphical user interface, text, application

Description automatically generated**

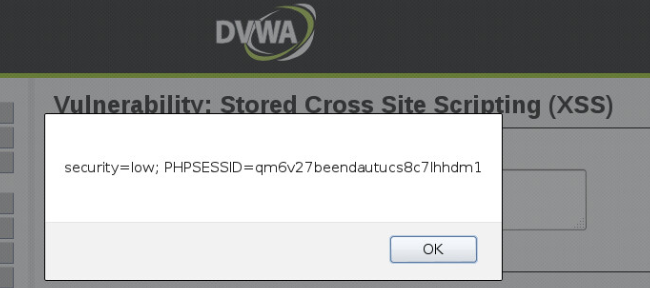
On page 24, import the snapshot equivalent to Figure 27. See the example below:



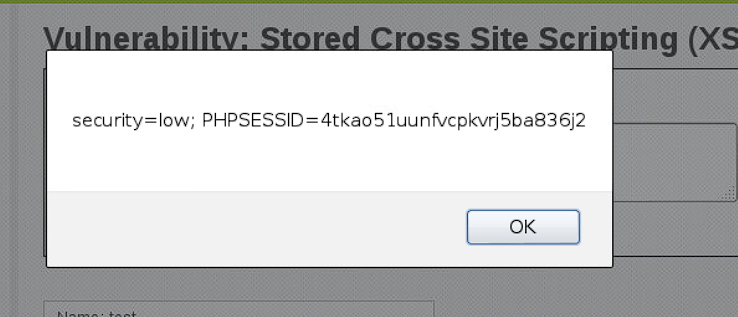
1. **Import your snapshot (similar to example above) here:**



On page 29, import the snapshot equivalent to Figure 13. See the example below:



1. **Import your snapshot (similar to example above) here:**

****