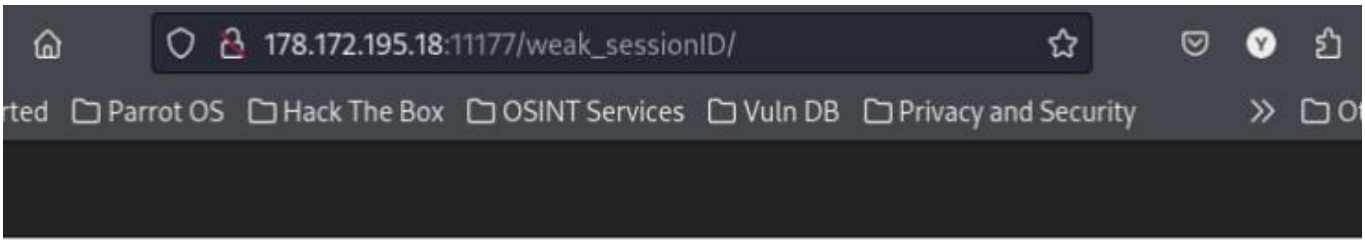
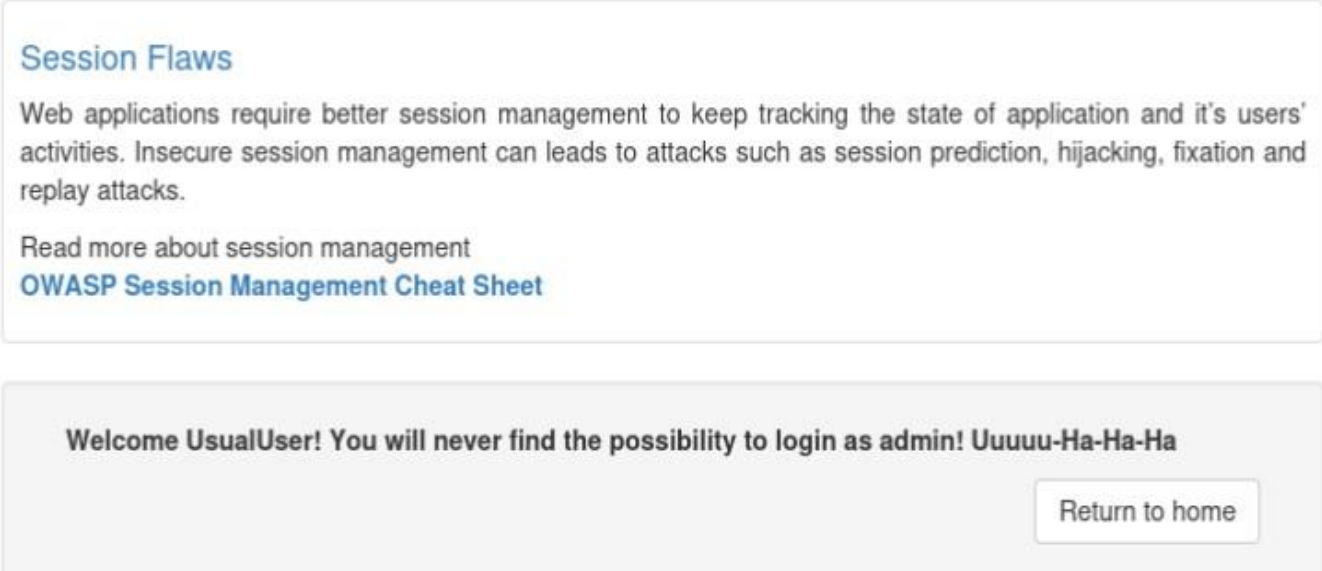
Web Application Security Testing -> **Weak Session ID. Session Fixation. CSRF**

Weak Session ID





Solution

1. Launch a terminal (`[Ctrl]`+`[Alt]`+`[T]`)
2. Type `curl -v http://178.172.195.18:11177/weak_sessionID/`.

   > **NOTE:** `-v` - to display additional information (*headers, cookies, etc.*)

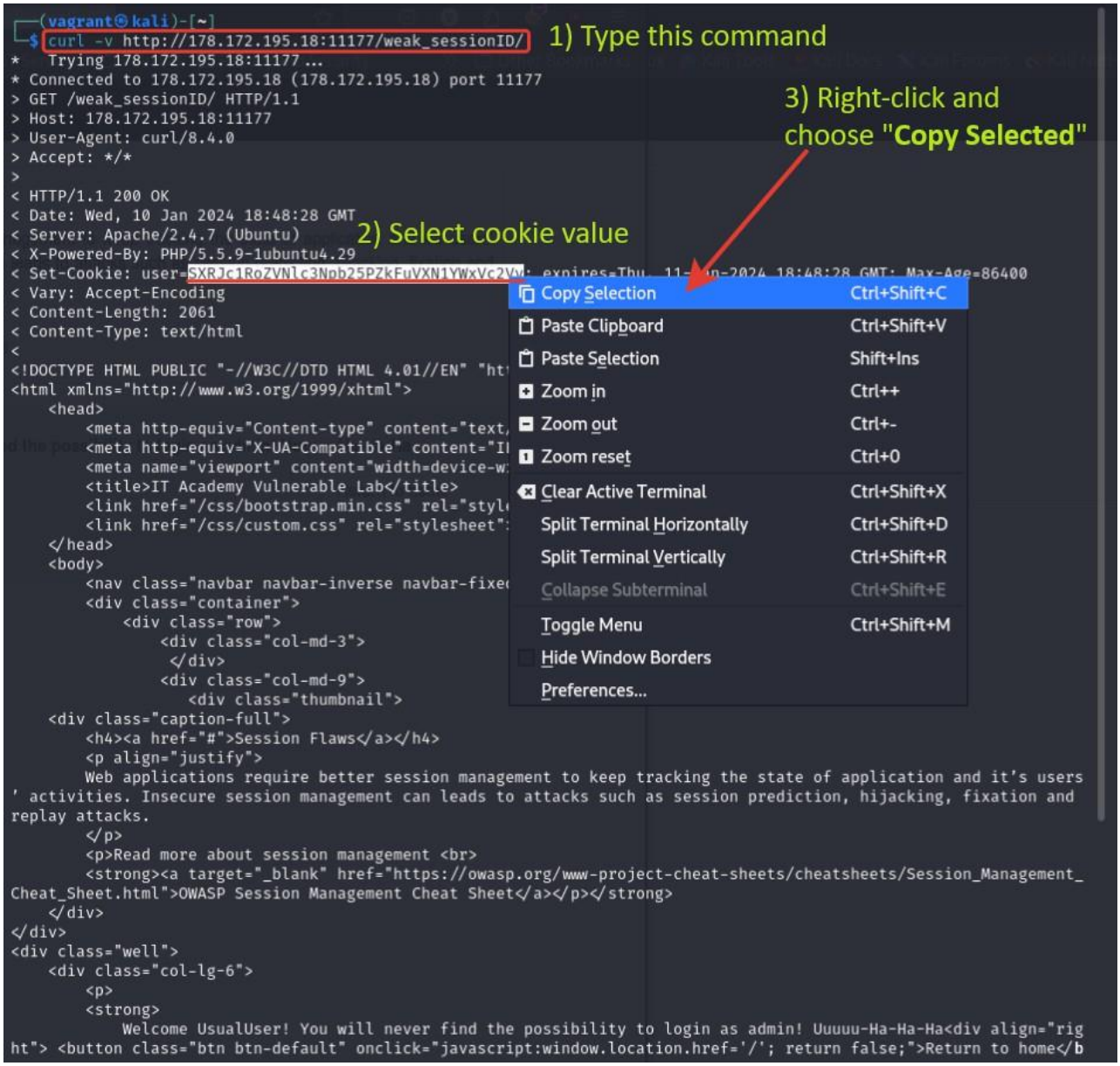
3. Let's see more information to find cookies.

   > **NOTE:**
   >
   > Here we found the `user=SXRJc1RoZVNlc3Npb25PZkFuVXN1YWxVc2Vy` cookie encoded in Base64 format.

> Let's see what data is encrypted.

4. Select `SXRJc1RoZVNlc3Npb25PZkFuVXN1YWxVc2Vy`

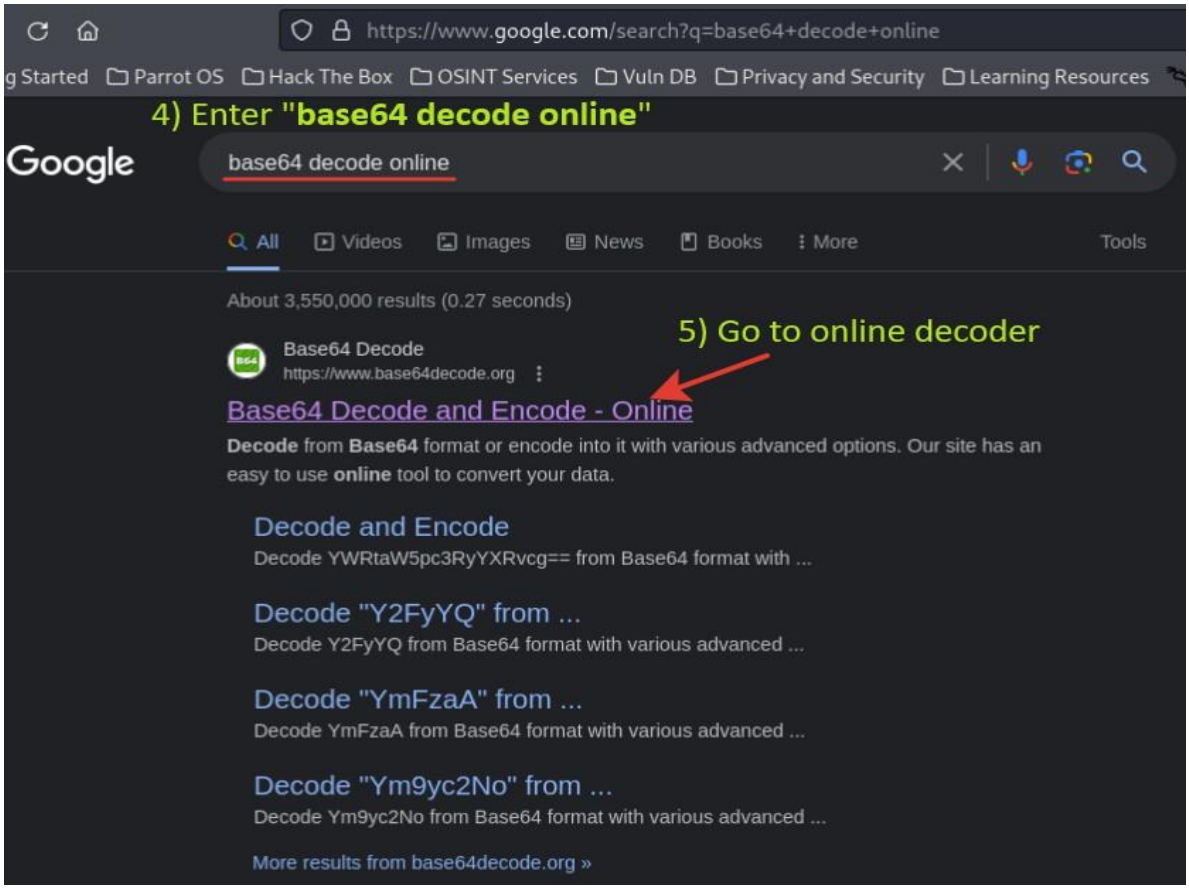
5. Right-click the selected ad and select "*Copy Selected*."



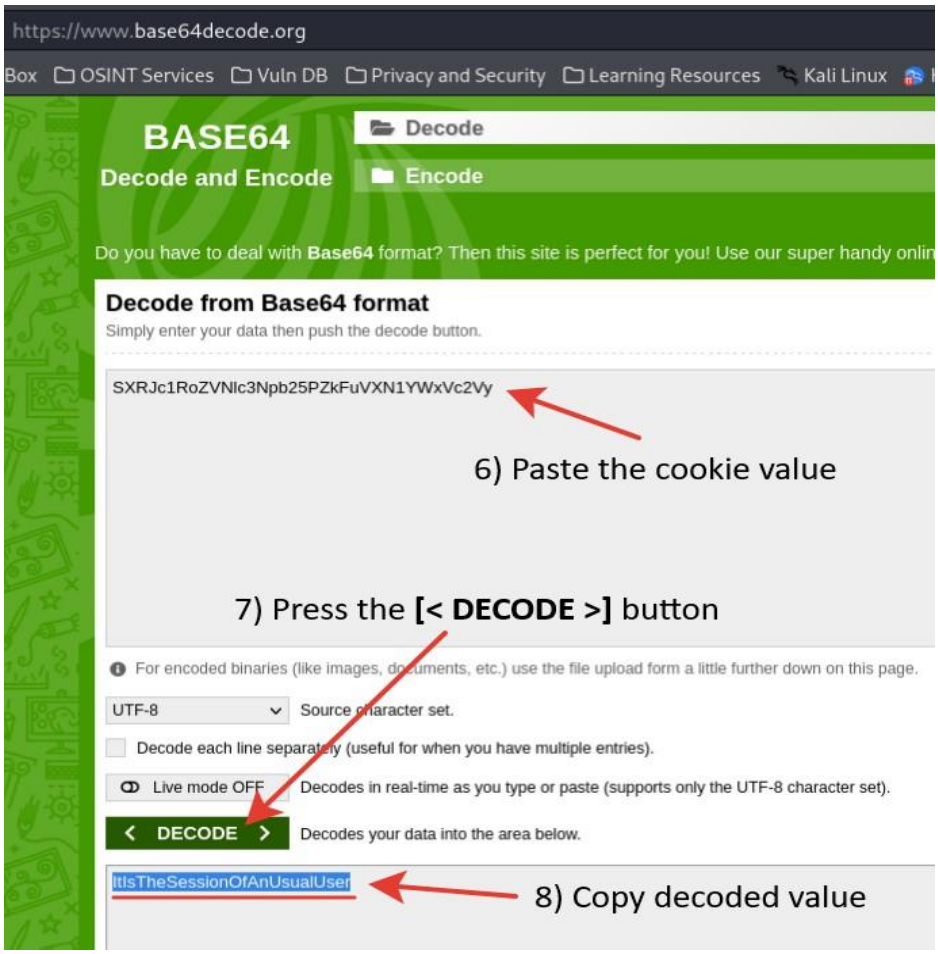
6. Open your preferred web browser.

7. Enter `base64 decode online`.

8. Open the first link (*e.g.*: *https://www.base64encode.org/*)



9. Right-click the "**Decode from Base64**" input and paste the copied value.
10. Press the `[< DECODE >]` button.
11. The `ItIsTheSessionOfAnUsualUser` value has been decoded.



> **Try to guess:**
>
> If we change `UsualUser` part to `Admin` and code it, it will affect the application...

12. Click the `[Encode]` button on the top of page.
13. Insert `ItIsTheSessionOfAnUsualUser` into the "**Encode to Base64**" input.
14. Change `ItIsTheSessionOfAnUsualUser` to `ItIsTheSessionOfAnAdmin`.
15. Press the `[> CODING <]` button.
16. Select `SXRJc1RoZVNlc3Npb25PZkFuQWRtaW4=` and copy the new encoded value.

17. Go back to the terminal and type `curl --cookie "user=SXRJc1RoZVNlc3Npb25PZkFuQWRtaW4=" -v http://178.172.195.18:11177/weak_sessionID/`

18. Obviously, this affected the behavior of the application and we received a "*Flag*".

```
┌──(vagrant㉿kali)-[~]
└─$ curl --cookie "user=SXRJc1RoZVNlc3Npb25PZkFuQWRtaW4=" -v http://178.172.195.18:11177/weak_sessionID/
*   Trying 178.172.195.18:11177 ...
* Connected to 178.172.195.18 (178.172.195.18) port 11177
> GET /weak_sessionID/ HTTP/1.1
> Host: 178.172.195.18:11177
> User-Agent: curl/8.4.0
> Accept: */*
> Cookie: user=SXRJc1RoZVNlc3Npb25PZkFuQWRtaW4=
>
< HTTP/1.1 200 OK
< Date: Wed, 10 Jan 2024 20:31:18 GMT
< Server: Apache/2.4.7 (Ubuntu)
< X-Powered-By: PHP/5.5.9-1ubuntu4.29
< Set-Cookie: user=SXRJc1RoZVNlc3Npb25PZkFuVXN1YWxVc2Vy; expires=Tue, 09-Jan-2024 20:31:18 GMT; Max-Age=-86400
< Set-Cookie: user=SXRJc1RoZVNlc3Npb25PZkFuYWRtaW4%3D; expires=Tue, 09-Jan-2024 20:31:18 GMT; Max-Age=-86400
< Vary: Accept-Encoding
< Content-Length: 2047
< Content-Type: text/html
<
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-type" content="text/html; charset=utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>IT Academy Vulnerable Lab</title>
        <link href="/css/bootstrap.min.css" rel="stylesheet">
        <link href="/css/custom.css" rel="stylesheet">
    </head>
    <body>
        <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation"></nav>
        <div class="container">
            <div class="row">
                <div class="col-md-3">
                 </div>
                <div class="col-md-9">
                   <div class="thumbnail">
    <div class="caption-full">
        <h4><a href="#">Session Flaws</a></h4>
        <p align="justify">
        Web applications require better session management to keep tracking the state of application and it's users
' activities. Insecure session management can leads to attacks such as session prediction, hijacking, fixation and
replay attacks.
        </p>
        <p>Read more about session management <br>
        <strong><a target="_blank" href="https://owasp.org/www-project-cheat-sheets/cheatsheets/Session_Management_
Cheat_Sheet.html">OWASP Session Management Cheat Sheet</a></p></strong>
    </div>
</div>
<div class="well">
    <div class="col-lg-6">
        <p>
        <strong>
            Ok, you won. Use this flag to pass the task: {51e8224582c15f5f82a8add9e88}<div align="right"> <button c
lass="btn btn-default" onclick="javascript:window.location.href= '/' ; return false;">Return to home</button></div>
        </strong>
```

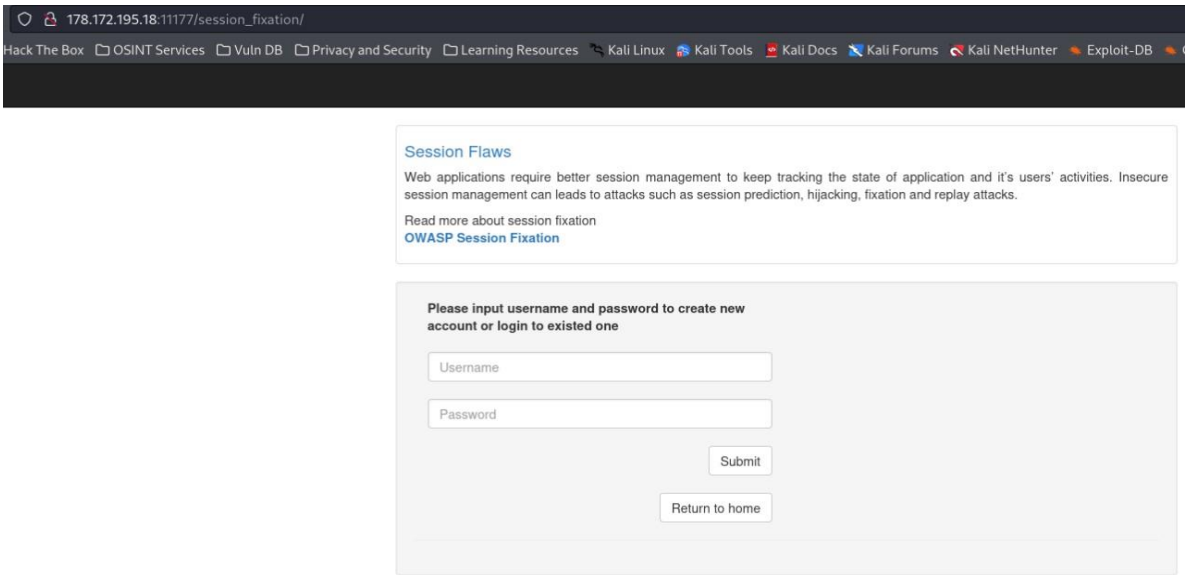
13) Add to curl command --cookie "user=SXRJc1RoZVNlc3Npb25PZkFuQWRtaW4="

14) Collect the "Flag"

Session Fixation

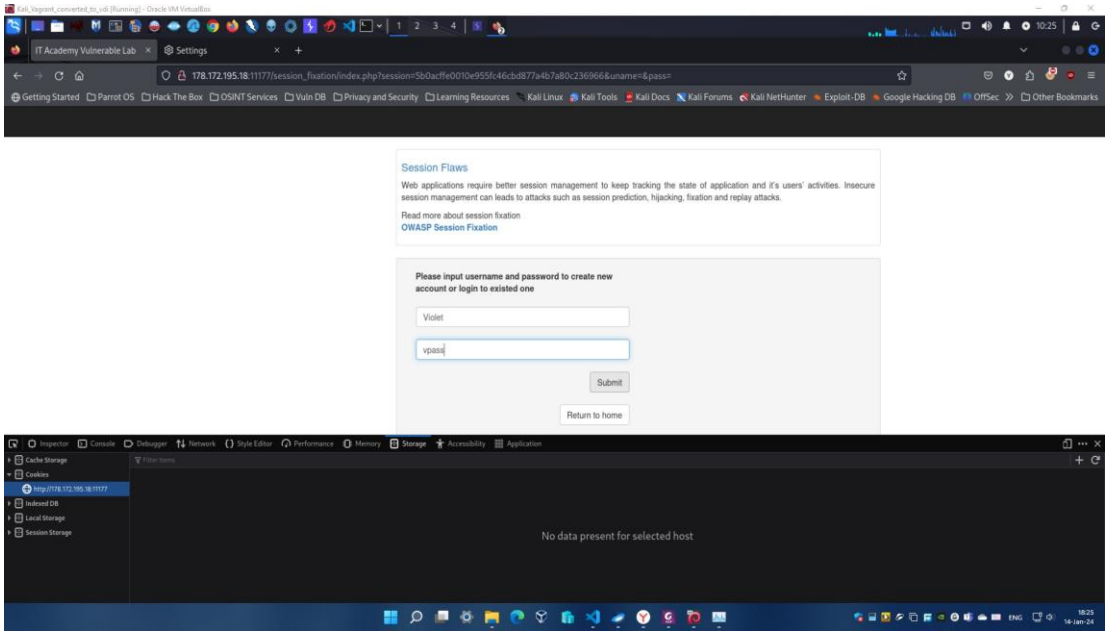
Task

1. Your target is admin!
2. Try to attack him via any messenger and get the access to admin account.
3. You will find the flag there.



Solution

1. To use a session fixation attack, we need to verify that the basic requirements are met.
    1. Session IDs must be persistent and can't be changed by privilege escalation.
    2. The web application must support operations with the provided sessions and not modify them.
2. Before we begin, let's clear all cookies (*if any*) from the target web application.
    1. Press the `[F12]` button (*to open the developer toolbar*).
    2. Go to the "*Storage*" tab.
    3. Expand the "*Cookies*" menu.

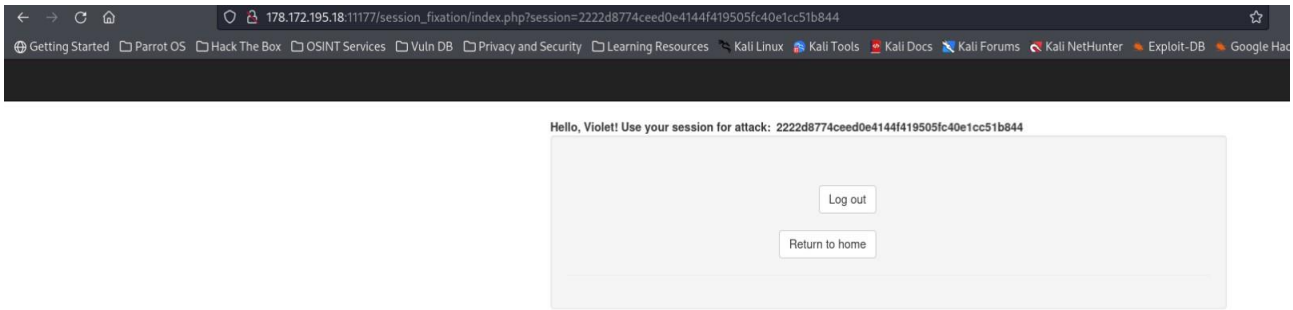


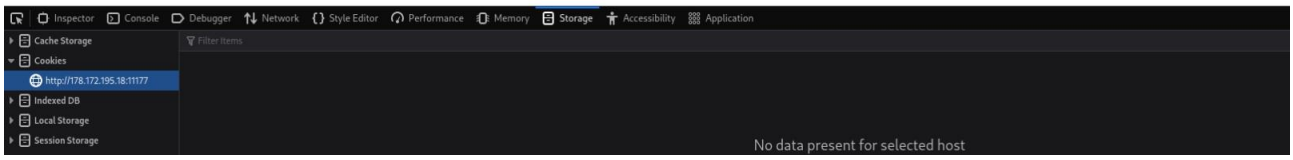
   4. Right-click on them and select "**Delete All**".

3. Now we need to enter any username and password (*for example:* `Violet`/`vpass`).

> **NOTE:**
>
> If the user does not exist, a new account will be created.
>
> If the message "*You have entered an incorrect username or password*" appears, the user exists.

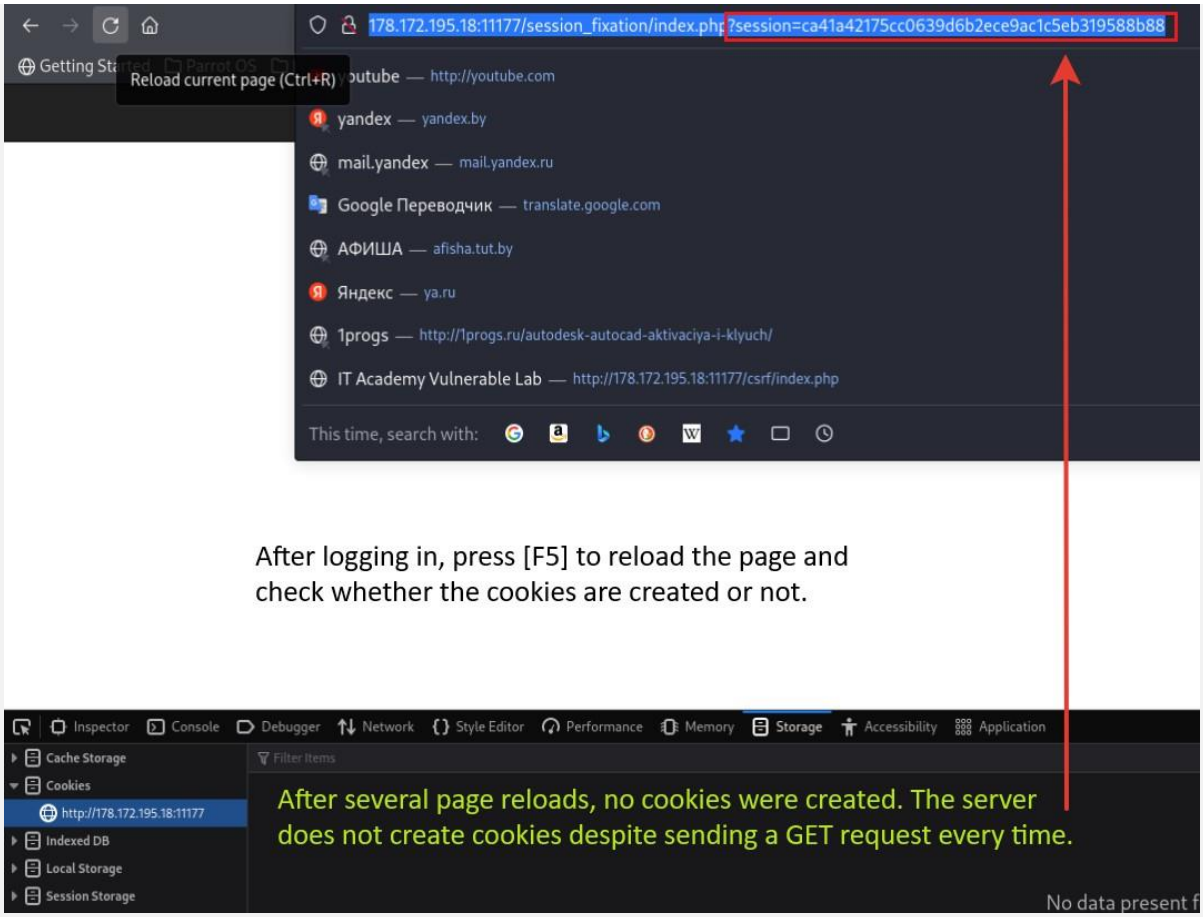
4. After logging in, we need to check what are the cookies were installed.
   1. Press the `[F12]` button.
   2. Go to the "*Storage*" tab.
   3. Expand the "*Cookies*" menu.

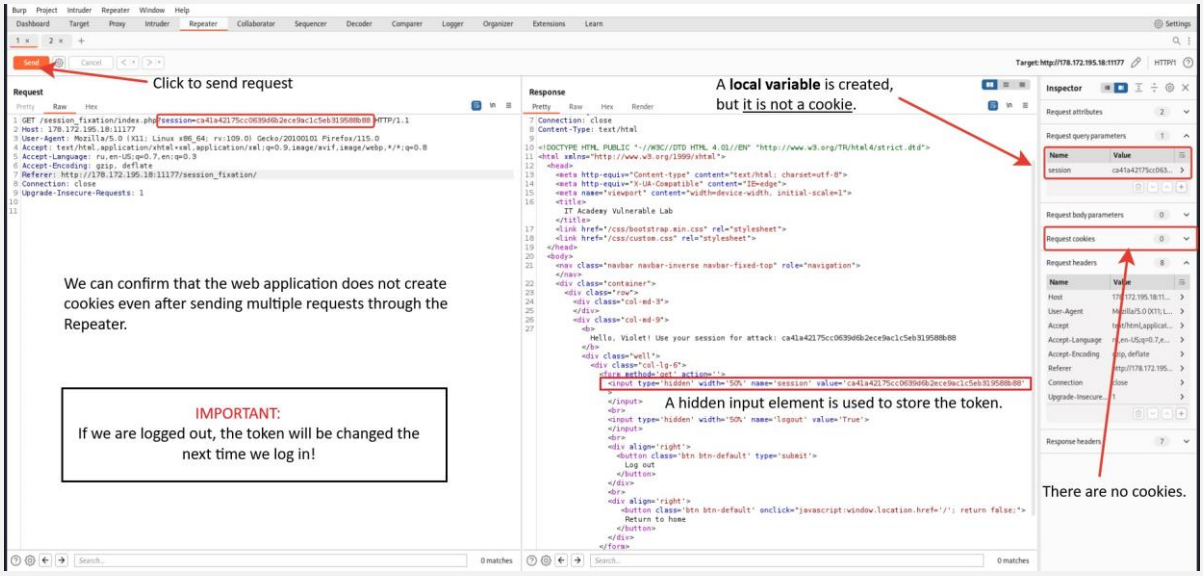


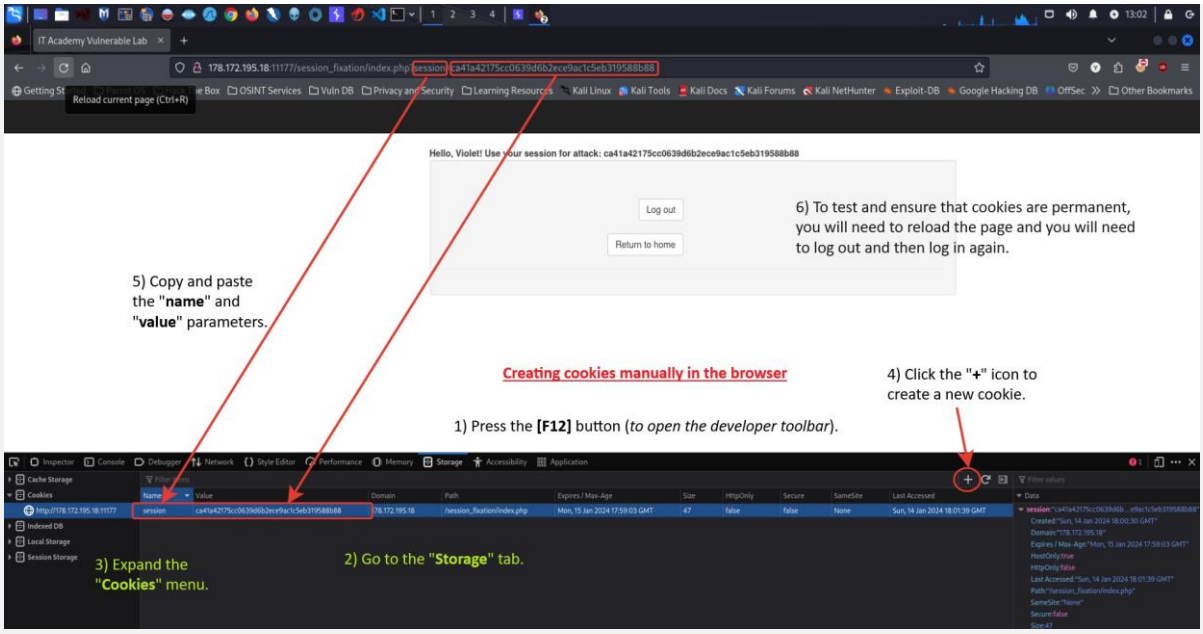


5. As we can see, cookies were not set. As a result, all necessary requirements are met.

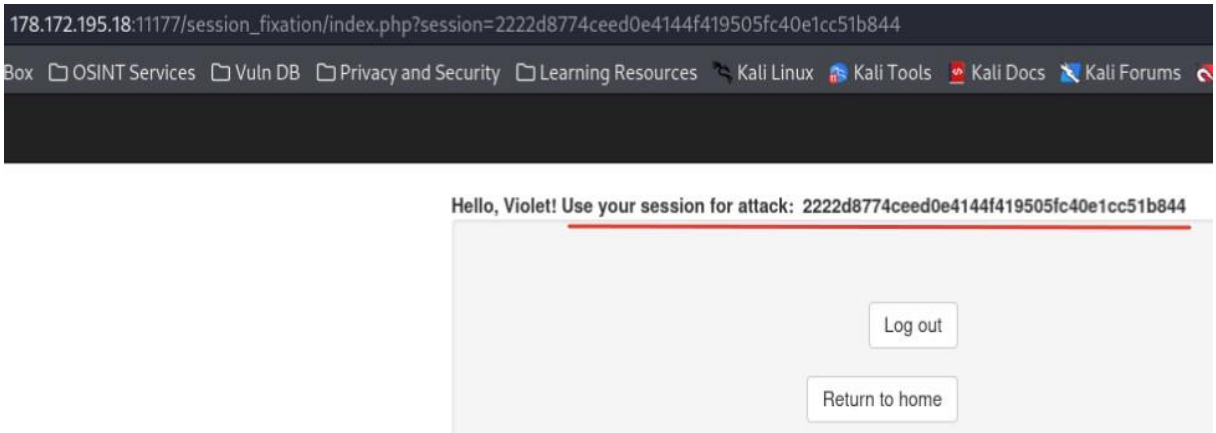
> **IMPORTANT NOTE:**
>
> If we look at the URL, the construct `?session=2222d8774ceed0e4144f419505fc40e1cc51b844` should set that session ID, but nothing happened. I don't know why, but this may make it impossible to carry out this attack. I'll try to implement this, but I'm not sure if it's possible.
>
> According to RFC 6265, cookies can be set via the Set-Cookie header, using JavaScript, or received from a server application, they are also can be installed manually in a real browser. A GET request can only create a temporary variable whose lifecycle is limited by page reloads.


178.172.195.18:11177/session_fixation/index.php?session=ca41a42175cc0639d6b2ece9ac1c5eb319588b88
Reload current page (Ctrl+R)
yandex — yandex.by
mail.yandex — mail.yandex.ru
Google Переводчик — translate.google.com
АФИША — afisha.tut.by
Яндекс — ya.ru
1progs — http://1progs.ru/autodesk-autocad-aktivaciya-i-klyuch/
IT Academy Vulnerable Lab — http://178.172.195.18:11177/csrf/index.php
This time, search with:
After logging in, press [F5] to reload the page and check whether the cookies are created or not.
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Cache Storage
Cookies
http://178.172.195.18:11177
Indexed DB
Local Storage
Session Storage
After several page reloads, no cookies were created. The server does not create cookies despite sending a GET request every time.
No data present f


Click to send request
A local variable is created, but it is not a cookie.
We can confirm that the web application does not create cookies even after sending multiple requests through the Repeater.
IMPORTANT:
If we are logged out, the token will be changed the next time we log in!
A hidden input element is used to store the token.
There are no cookies.



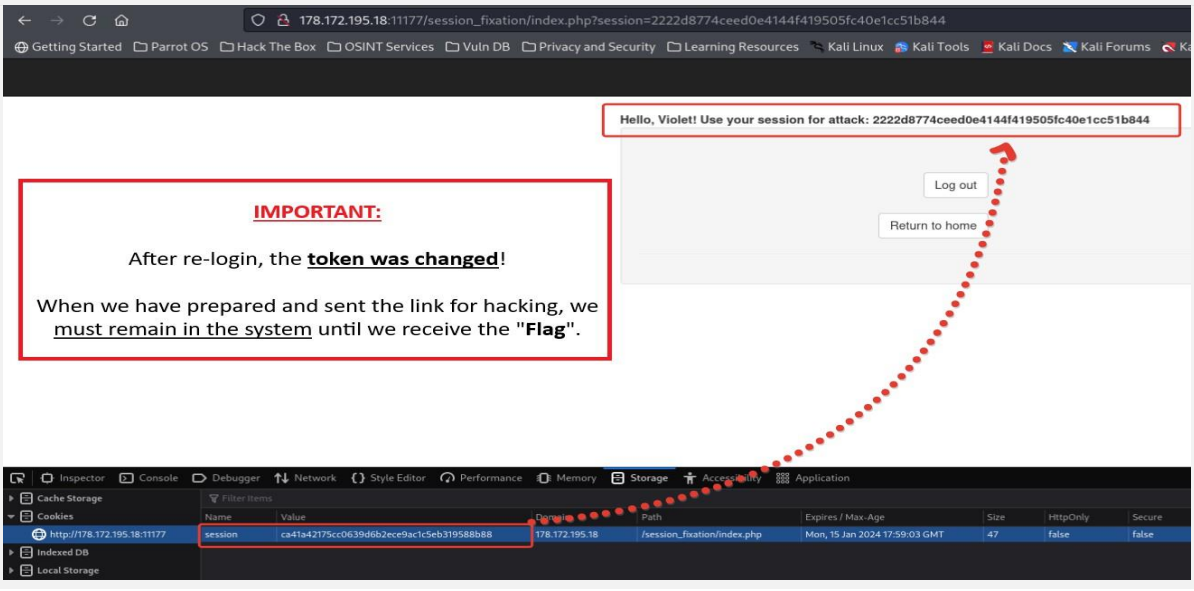
6. We see the message "*Use your session to attack: 2222d8774ceed0e4144f419505fc40e1cc51b844*".



IMPORTANT NOTE:

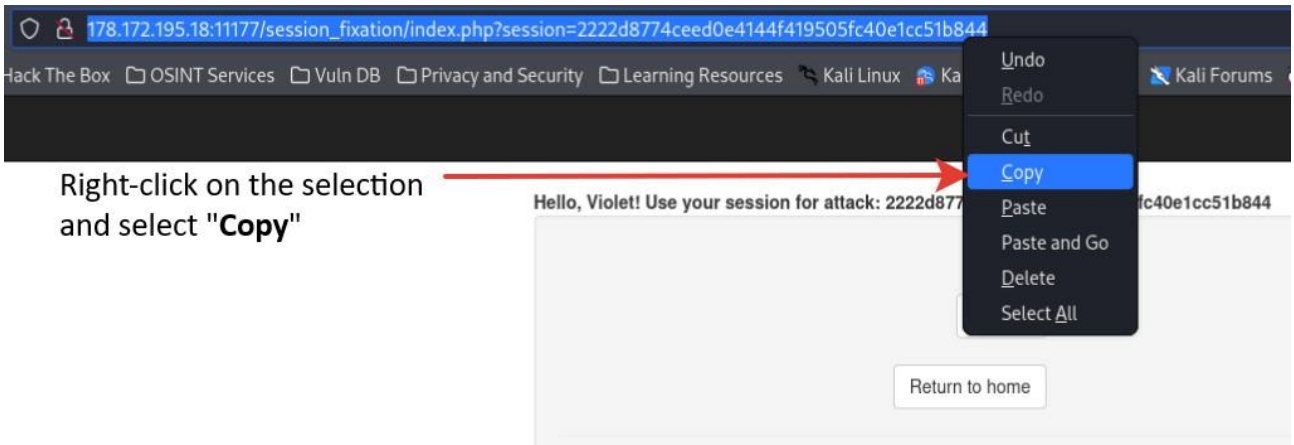
After re-login, the **token was changed**!

When we have prefered and esnt the link for hacking, we **must remain in the system** until we receive the "*Flag*".



7. To carry out the attack, we need to prepare a link and send it to the admin.

8. Select everything in the address bar. Right-click on the selection and select "**Copy**".



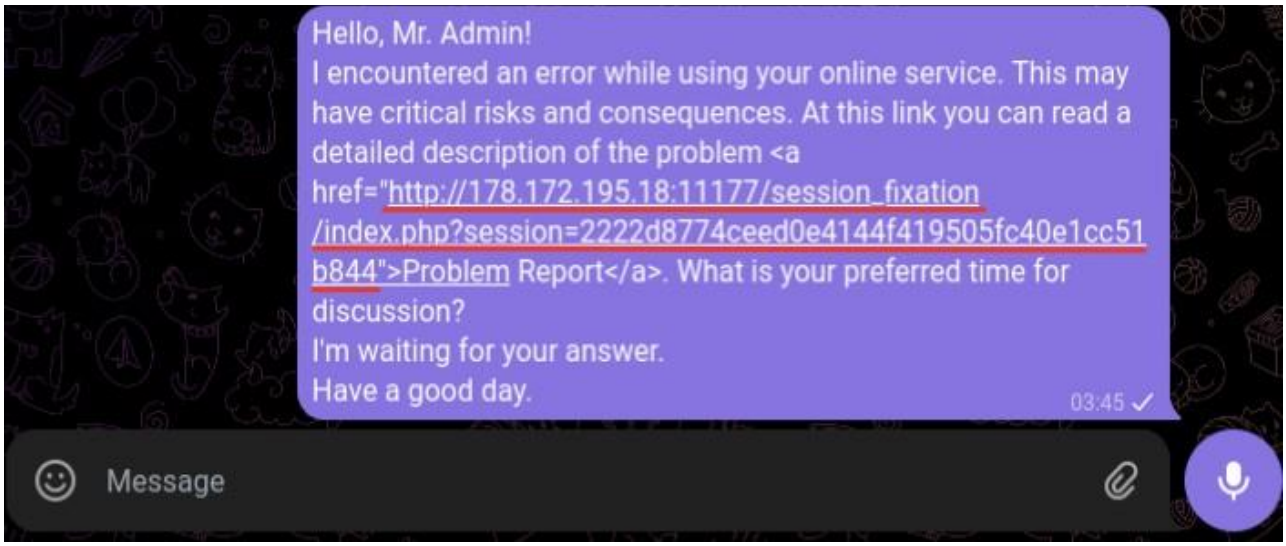
> **NOTE:** It already contains all the necessary parameters.

9. In fact, this line is already a link, but to hide the context, it is advisable to place it in the `href=""` attribute of the `<a>` tag and find a suitable messenger that will allow you to display only the link title, hiding the HTML code.

   You should end up with something like this.

```
<a href="http://178.172.195.18:11177/session_fixation/index.php?
session=2222d8774ceed0e4144f419505fc40e1cc51b844">Problem Report</a>
```

10. Open the messenger and, using social engineering, try to distract the administrator's attention so that he clicks on our link. Additionally, you can find out the time when he will do this.



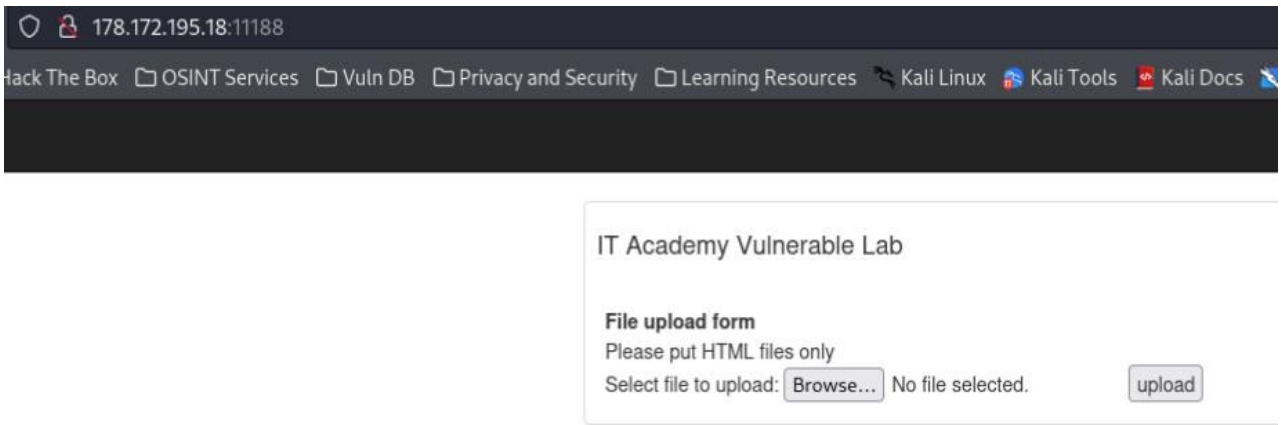
11. Refresh the page periodically to check for expected privilege increases. If an administrator answers you and asks a question about the missing report, you can be sure that he will use our link.

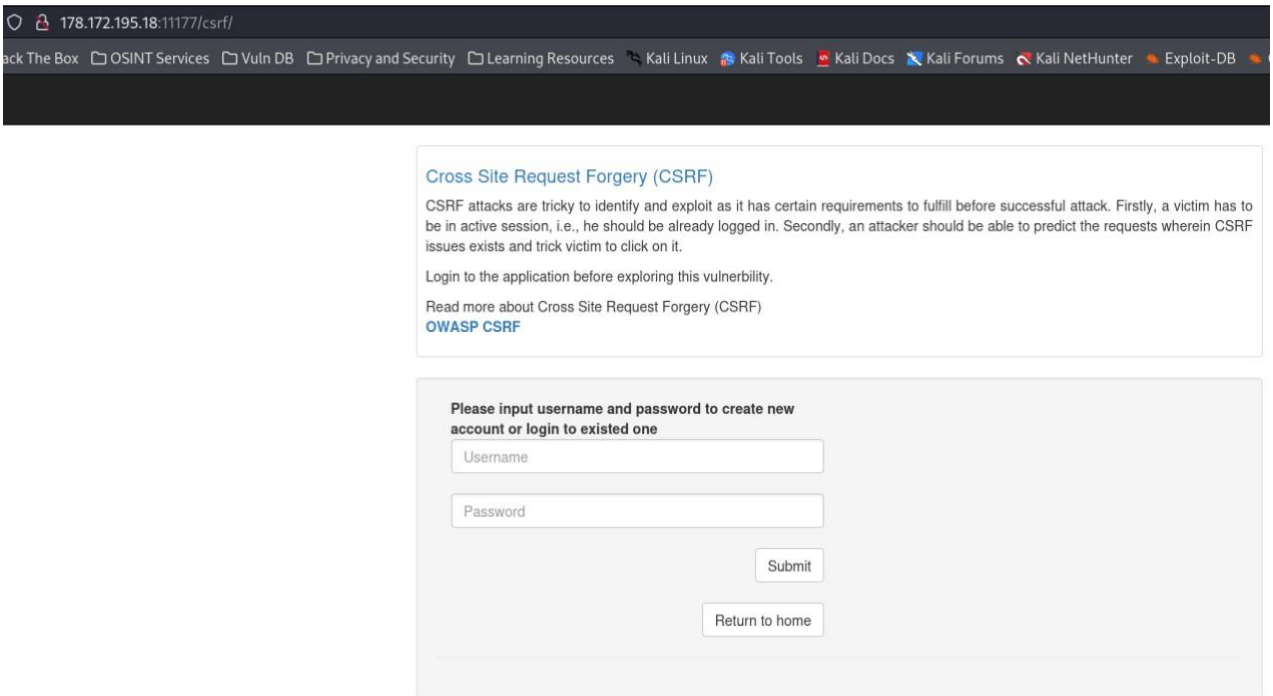
12. Refresh the page and get the "*Flag*".

CSRF

Task

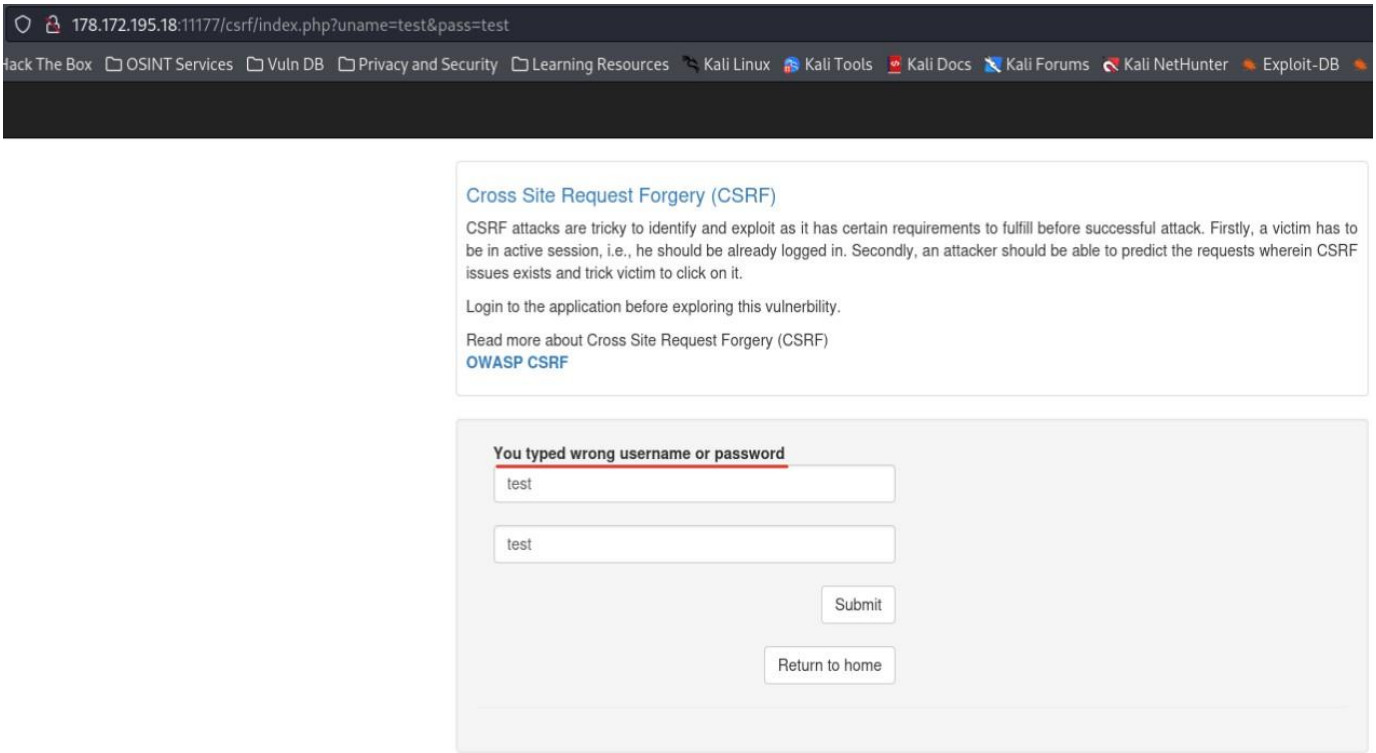
1. Your target is admin!
2. Try to attack him via any messenger and get the access to admin account.
3. You will find the flag there.
4. This page can help you during preparation and attack phases.



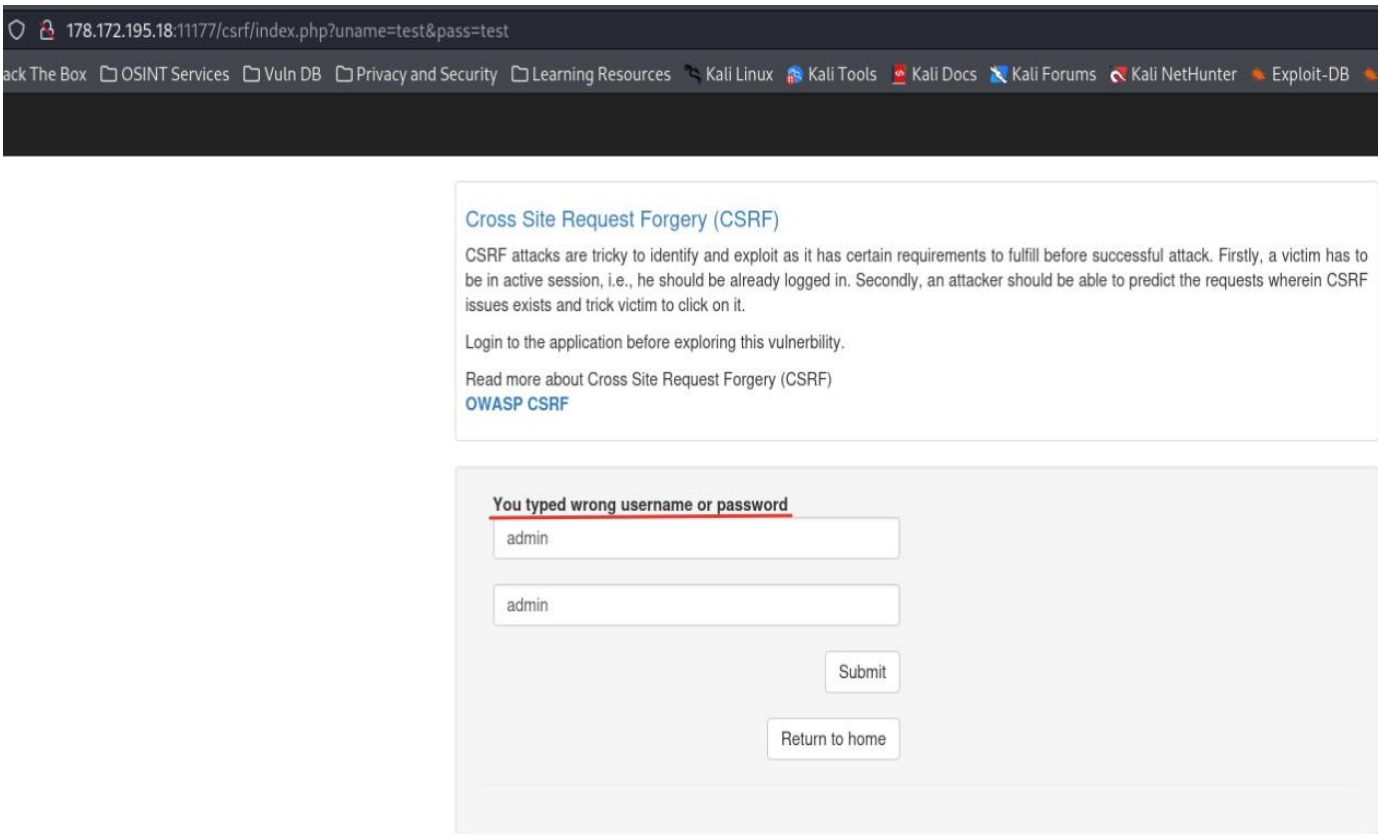


⚠️**WARNING:** *There is a bug in this task!!!* ⚠️

When I try to login as a `test` user with the password `test`, I get the error "*You have entered an incorrect username or password*".



The same situation happens again when I try to login as user `admin` with password `admin`.



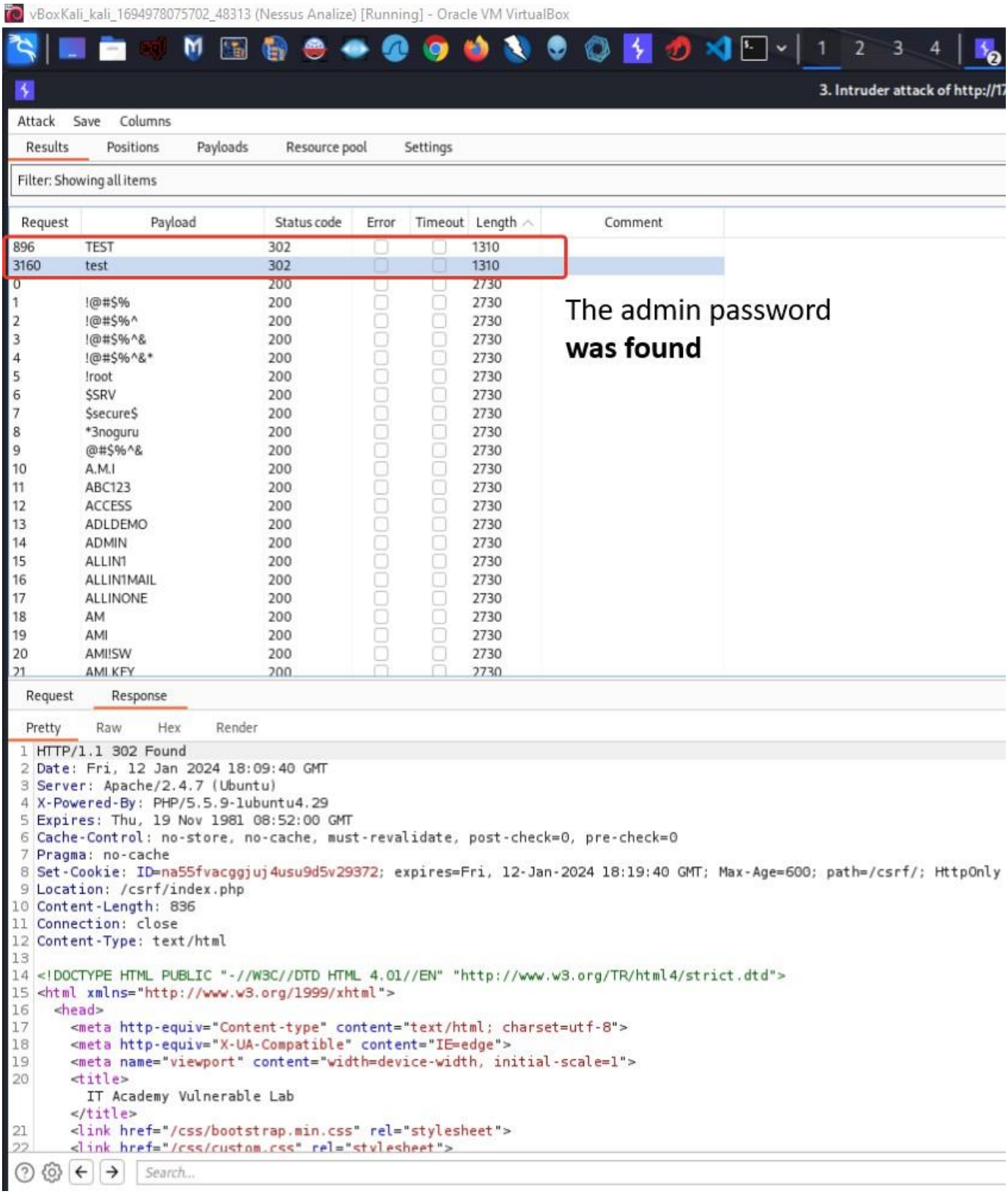
Of course, the system checks whether the user exists in some storage. The message that appears indicates

that user's `admin` or `test` exist. On the other hand, if no users exist, the system must **create a new one**.

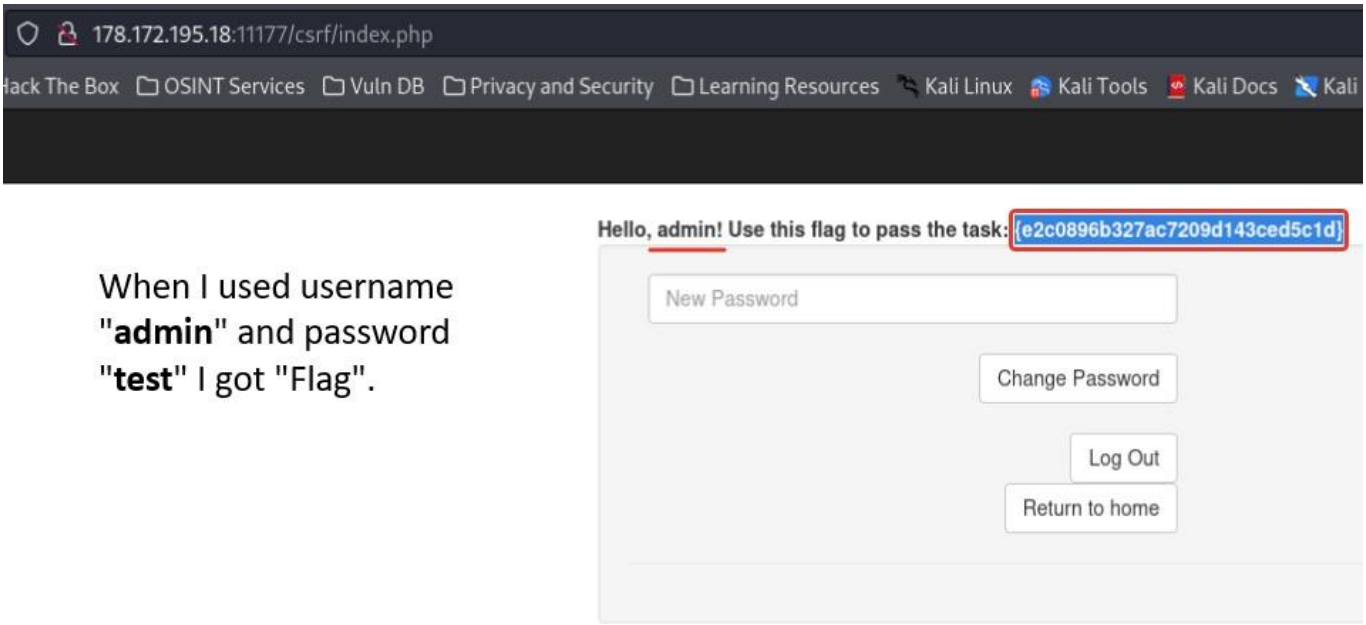
Let's **analyze what happens** when we use the csrf attack to solve this problem.

We should prepare an HTML page on which we will place the "*newPass*" form, where we will specify the password we know. Then send it to the administrator. When the administrator opens the page, his password should be replaced with ours.

Attention! If the administrator user has recently been attacked, perhaps his password has been changed to some simple one and now we can find it out by brute force. Let's check it out!



As we can see, the administrator password was found. Try logging in with these credentials.



Possible ways to fix the error

There are many ways to fix the error. It is necessary to reset the administrator password periodically.

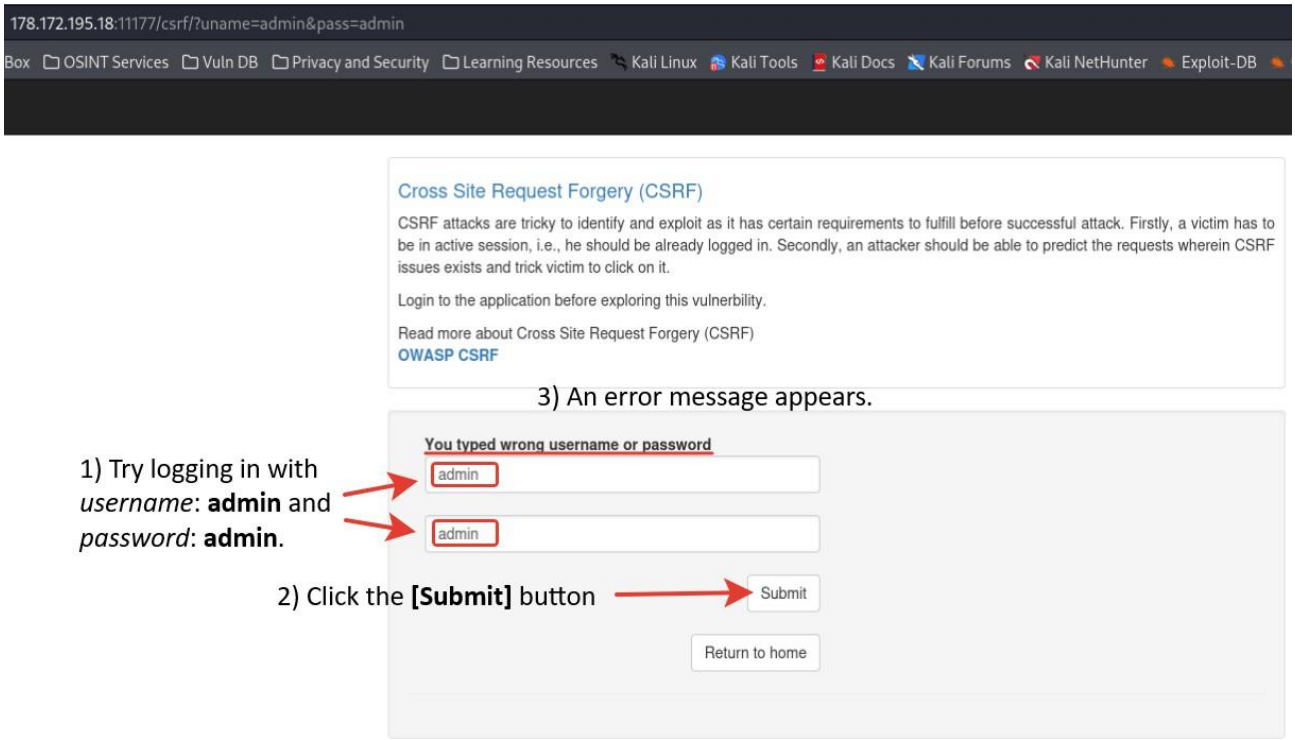
For example (*implementation dependent*):

1. Use a system script (*for file databases*)
2. Use stored procedures (*for a database management system*)
3. Change the business logic of the application after receiving the flag.

and so on.

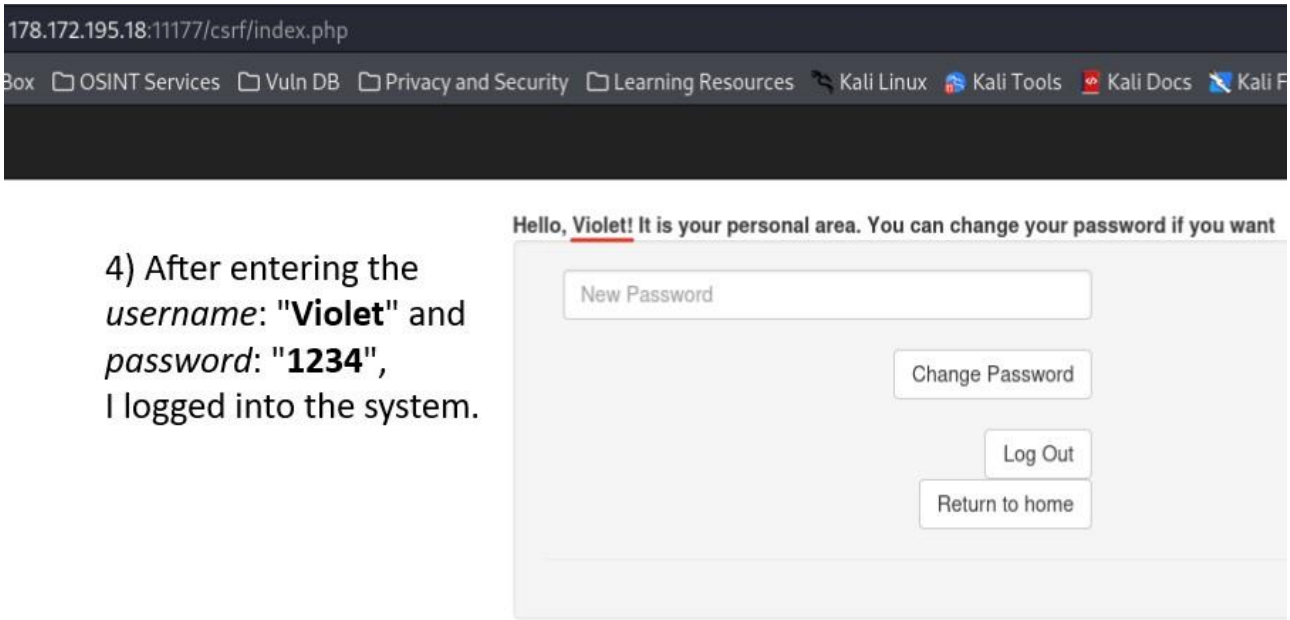
Solution

1. We need to login as administrator, but we don't know the administrator password.

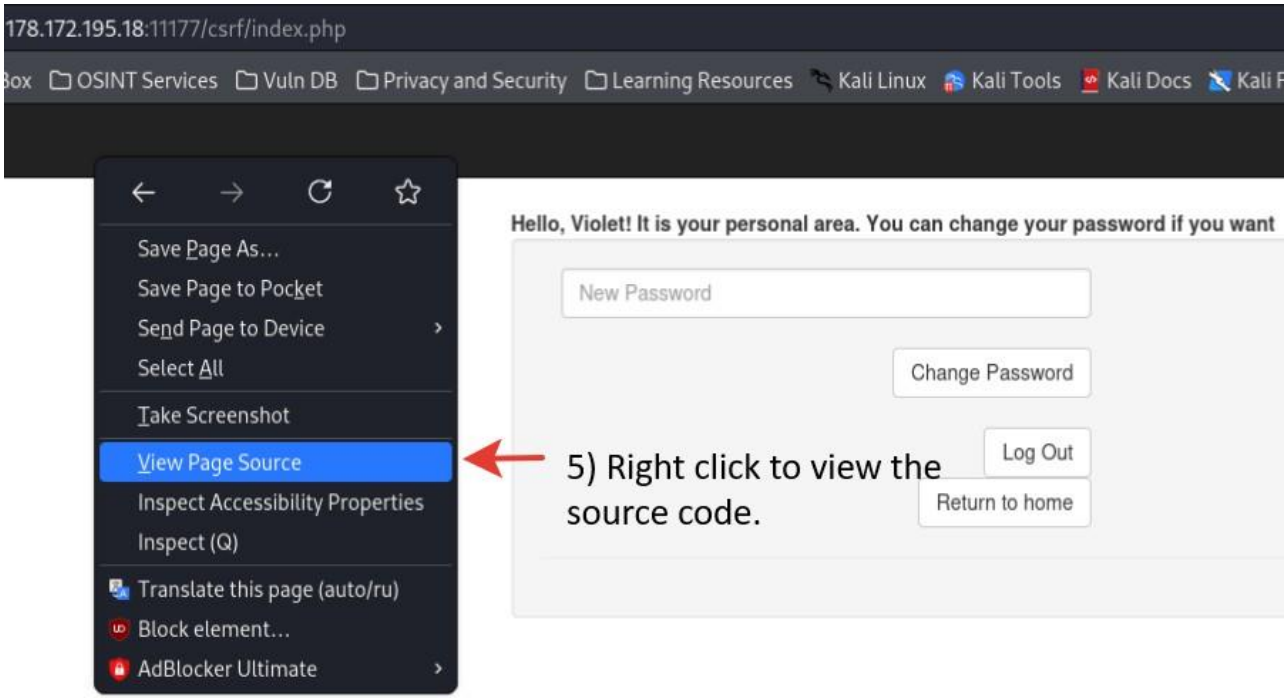


2. First of all we have to enter any username and password (*e.g.*: `test`/`test`). If the user does not exist a new account will be created.

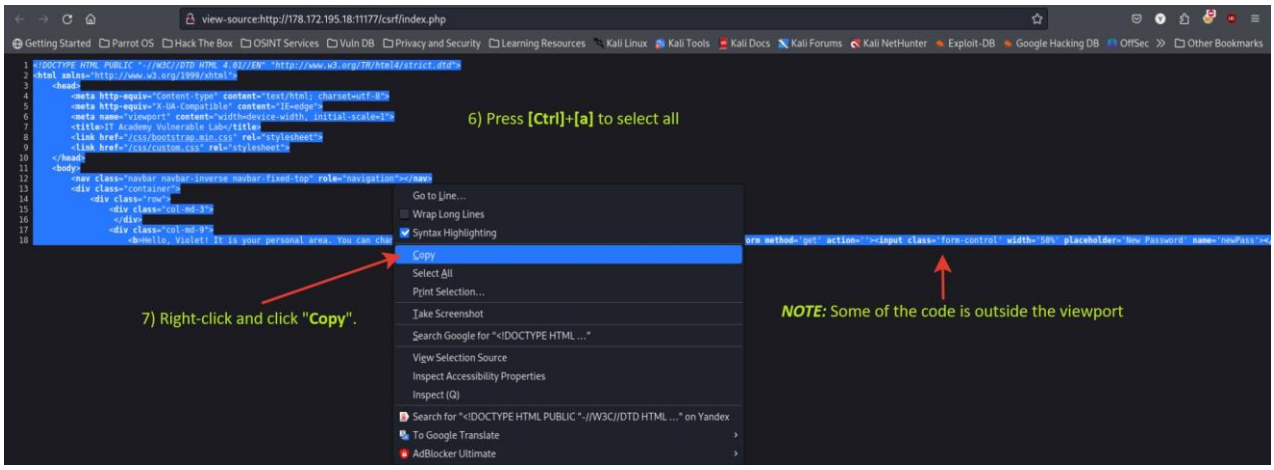
> **NOTE:**
>
> If the message "*You have entered an incorrect username or password*" appears, the user exists.
>
> I came across this and it helped me find a bug. I had to use a different user to login



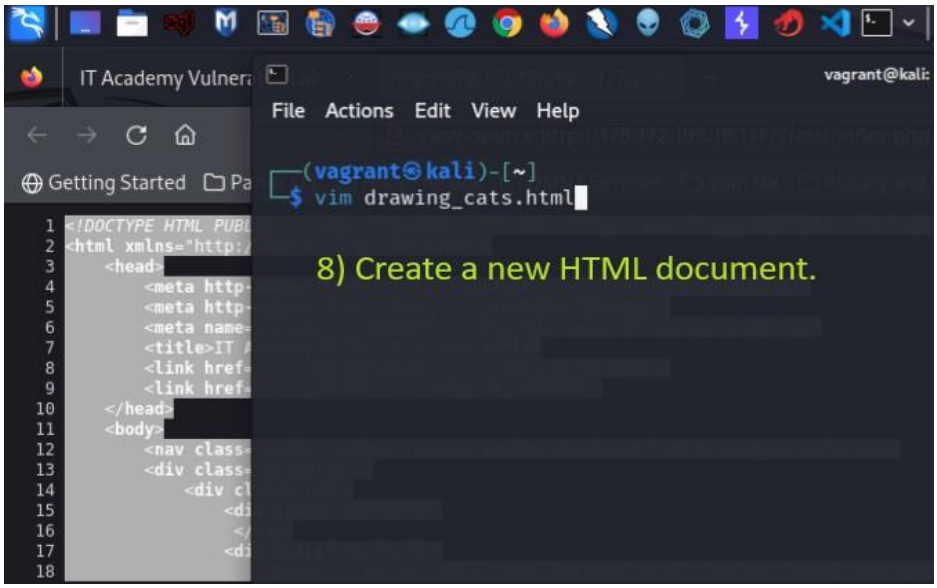
3. Right-click anywhere on the page and select "*View Page Source*"



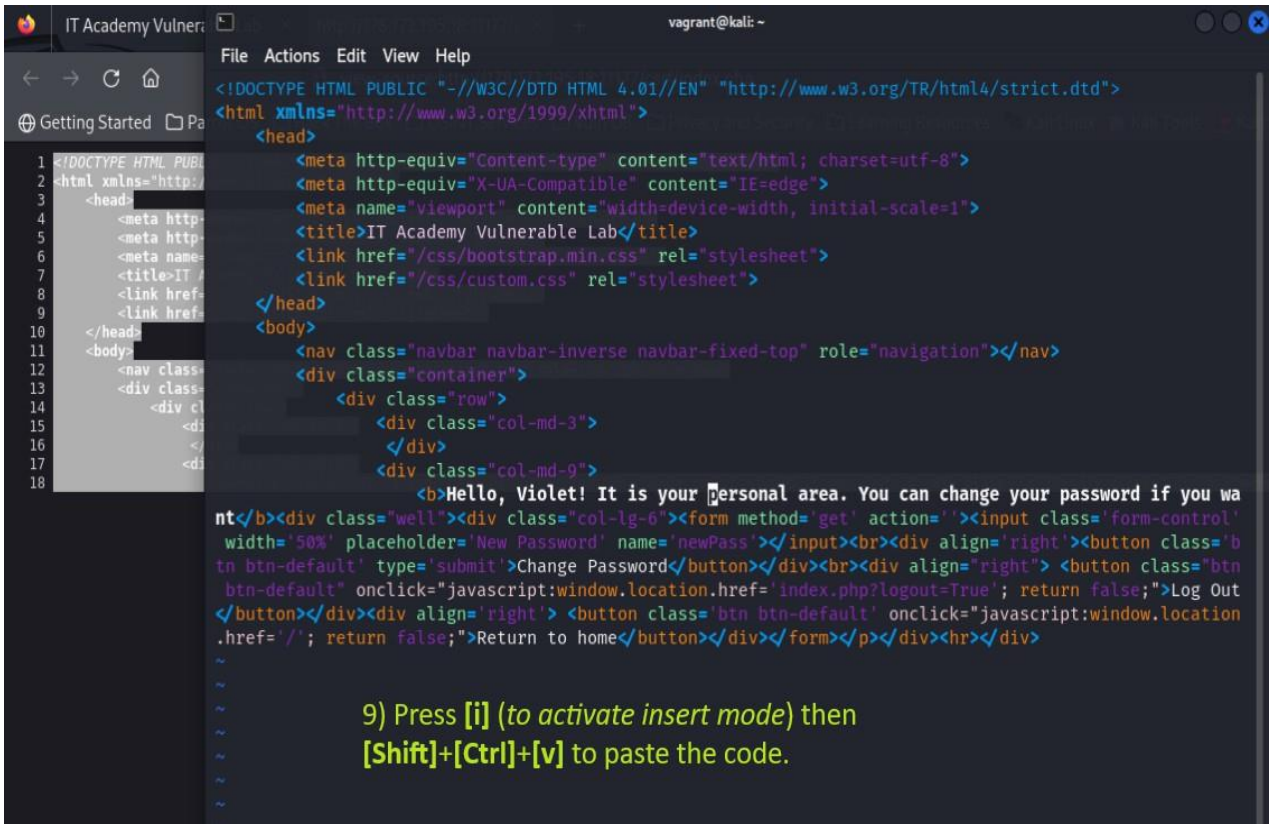
4. Some of the code is outside the viewport. To make analysis easier, press `[Ctrl]`+`[a]` to select it and right-click to "***Copy***" the selection.



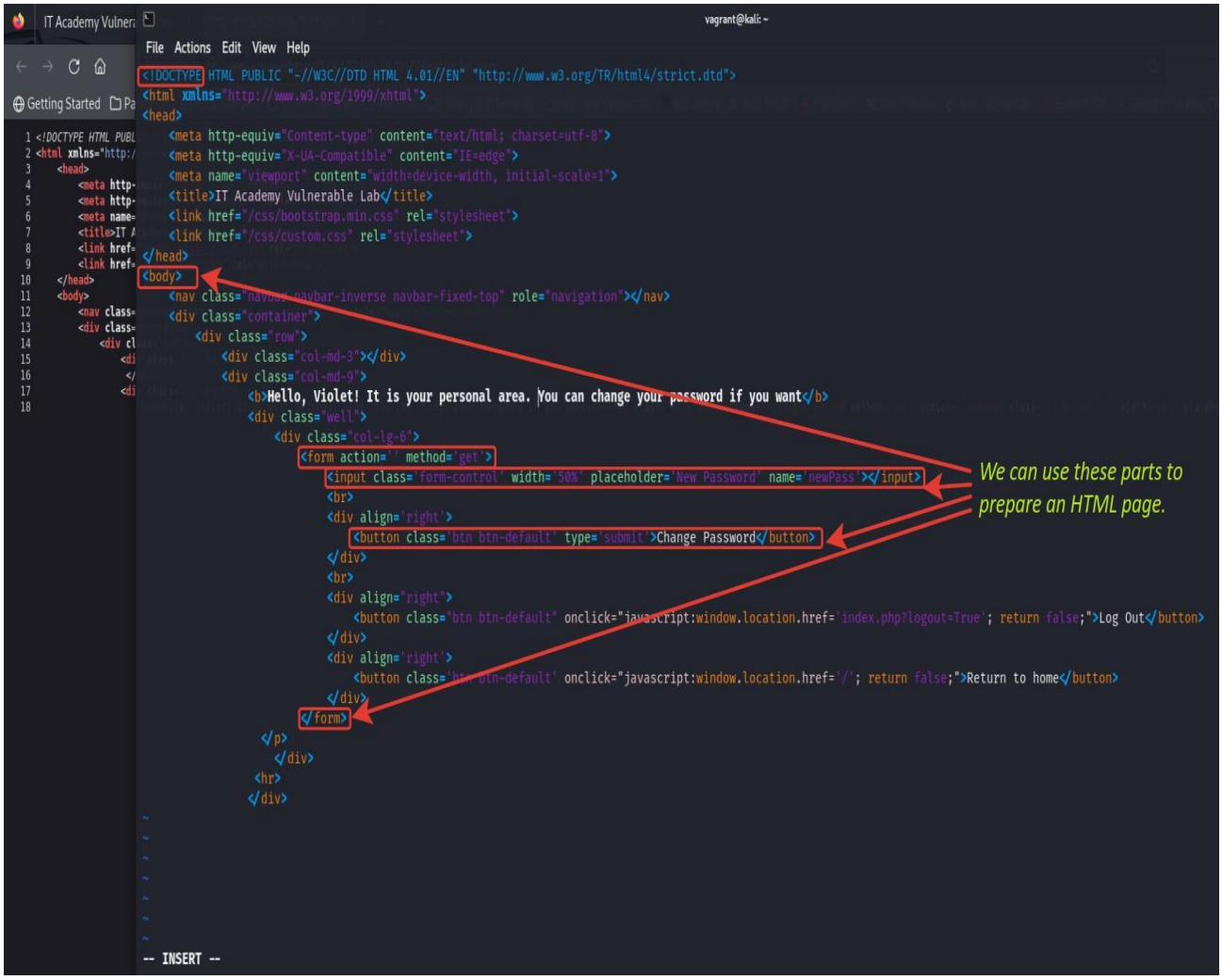
5. Open your preferred text editor to create the exploit html page.



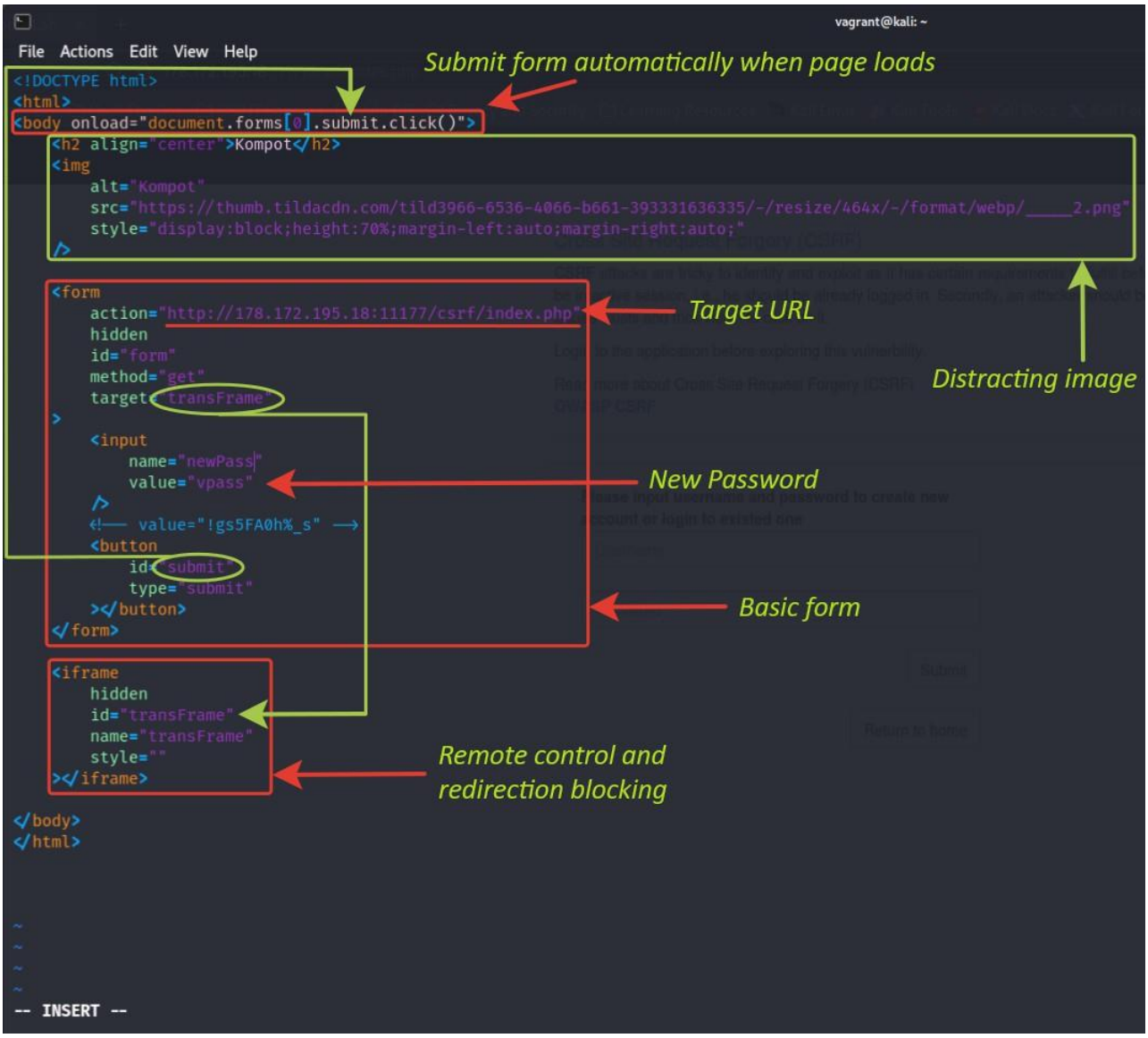
6. Press `[i]` then [Shift]+[Ctrl]+[v] to paste the code.



7. Refactor and analyze the source code.



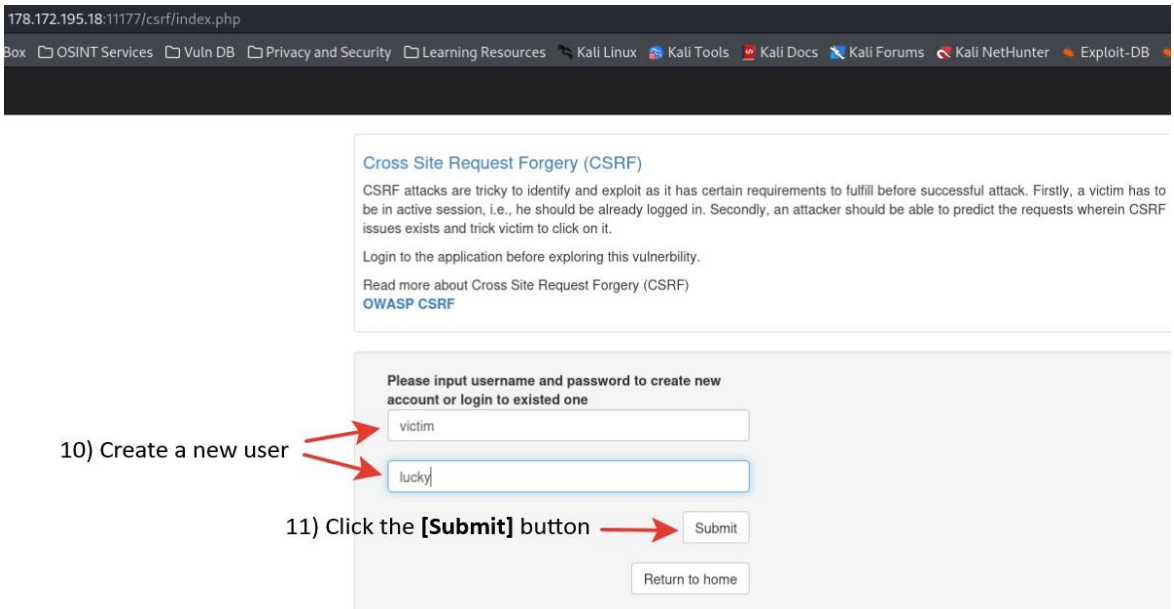
8. Cleaning up unnecessary code, adding automation scripts and distracting image.



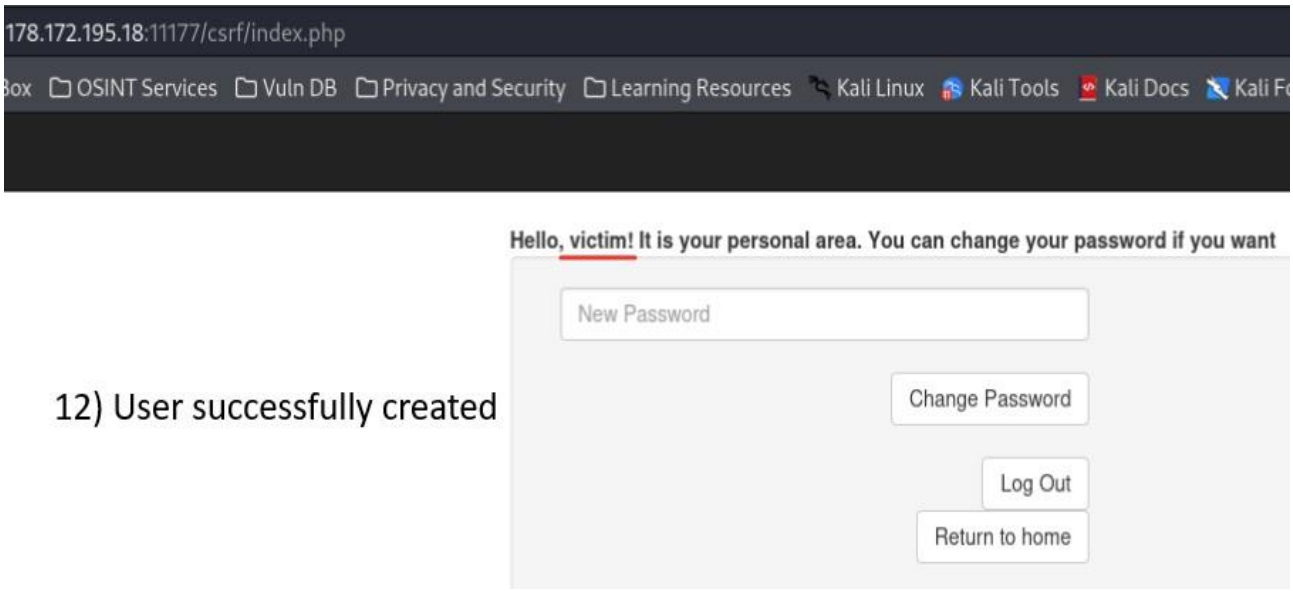
9. Press `[Esc]` (*to activate command mode*), then enter `[Shift]`+`[;]`, `[w]` (*write*), `[q]` (*exit*) and `[Enter]`.

10. All necessary well done. Let's check now it is work.

11. First, we need to create a new victim user. So, username "`victim`" password "`lucky`".

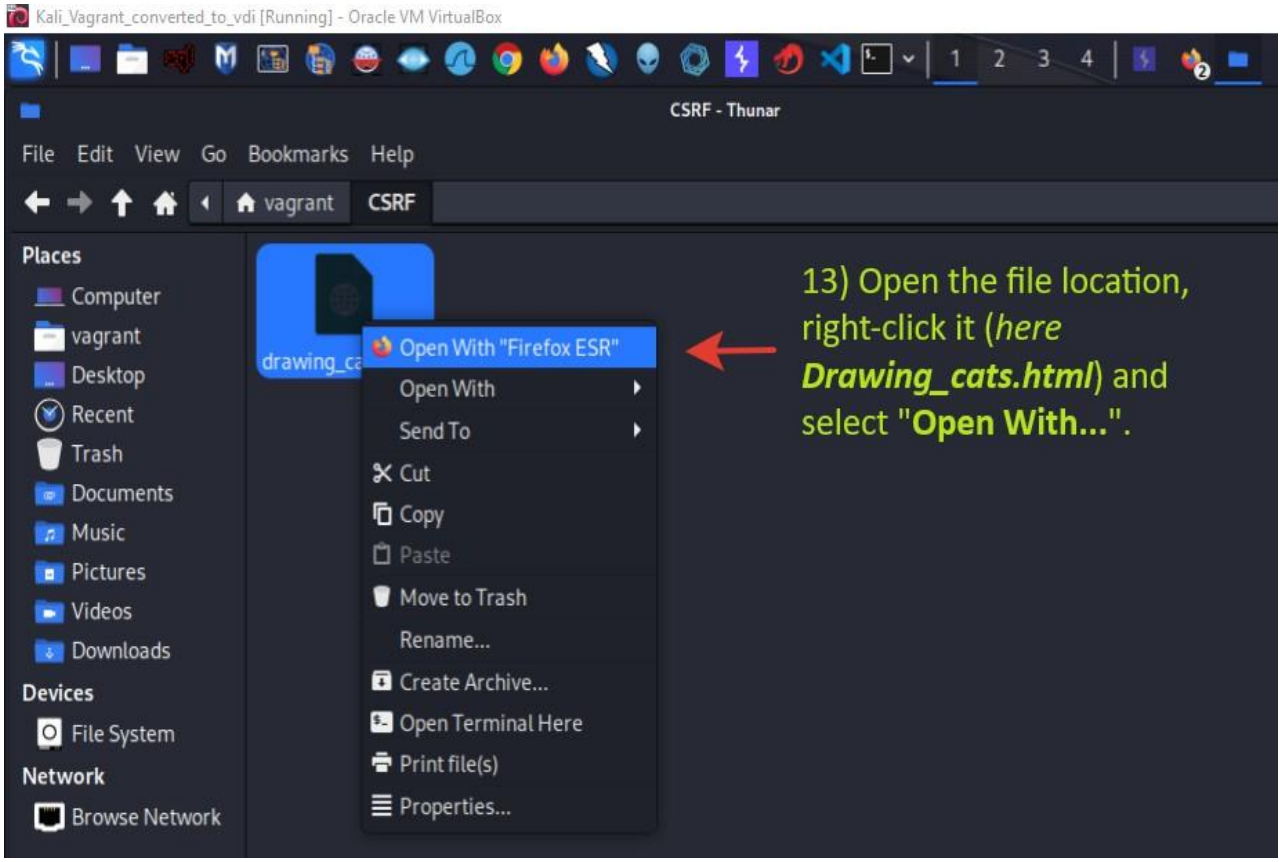


12. Enter *username:* "`victim`", *password:* "`lucky`" and click the `[Submit]` button.

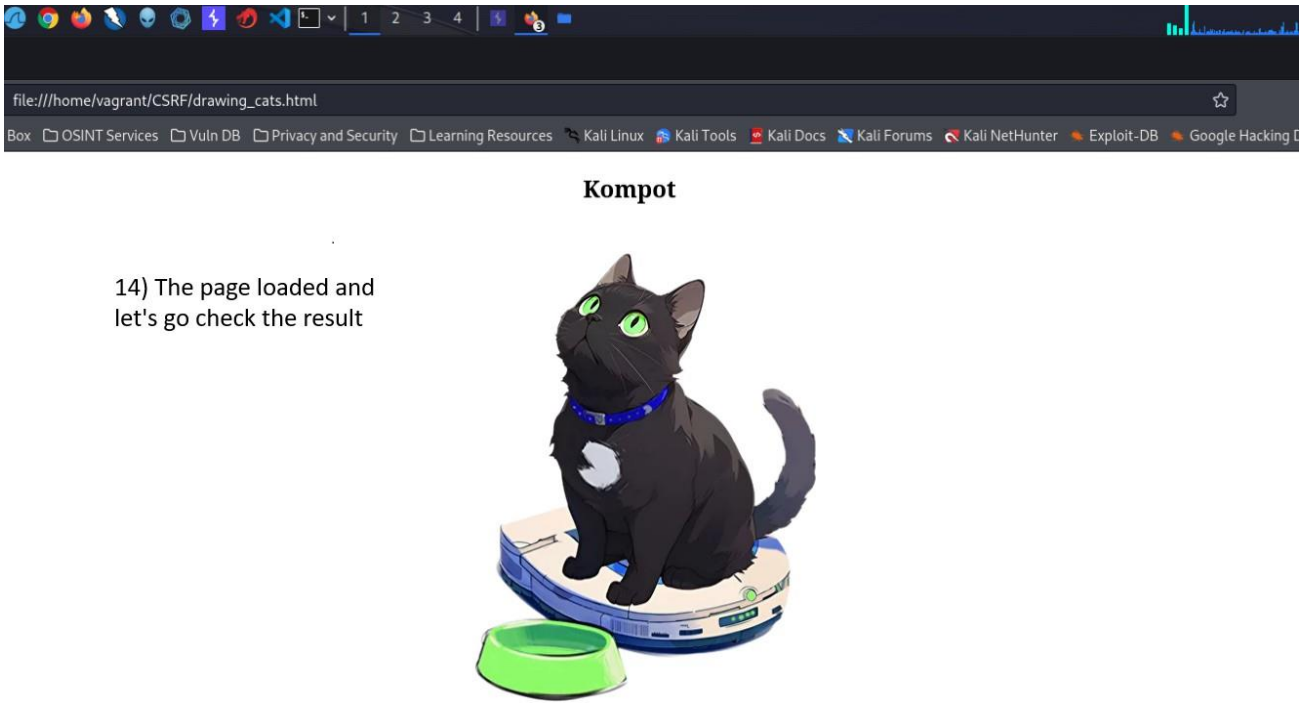


13. To carry out an attack, just open the prepared page in the browser.

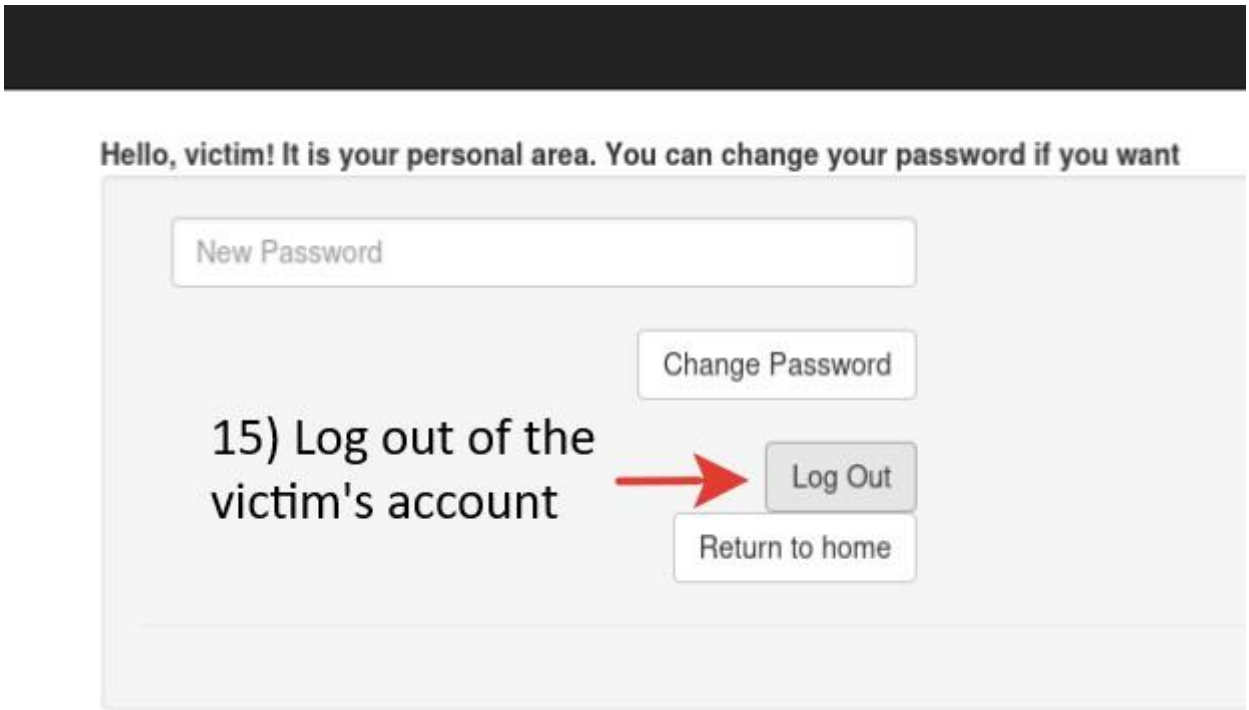
14. Open the file location, right-click it (*here* `drawing_cats.html`) and select "***Open With...***".



15. The page loaded and let's go check the result.

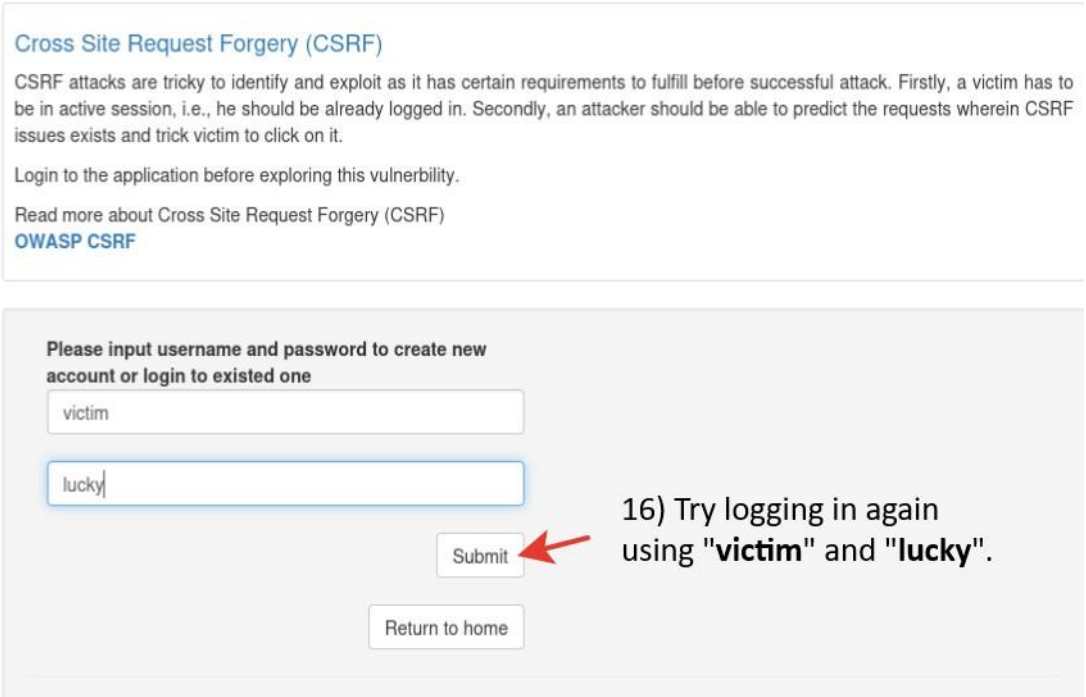


16. Log out of the victim's account.



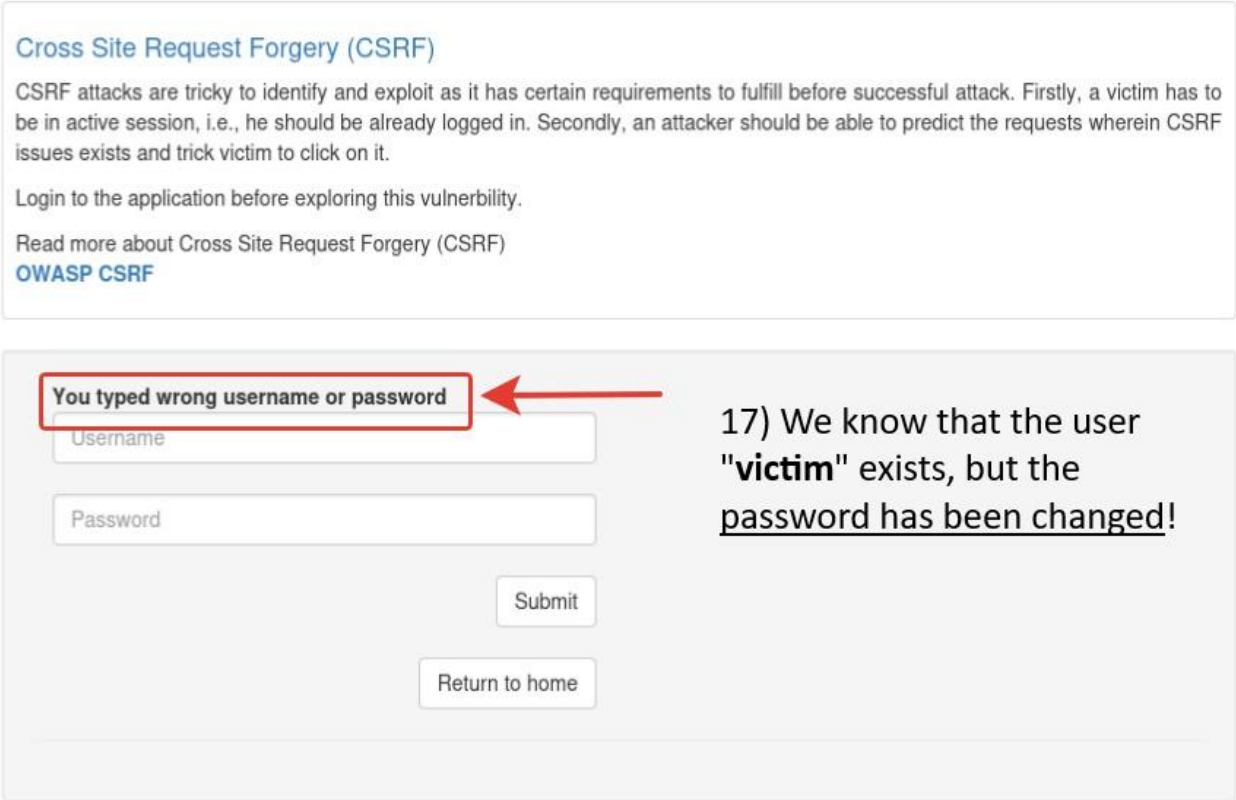
17. Try logging in again using "`victim`" and "`lucky`".

[redacted]

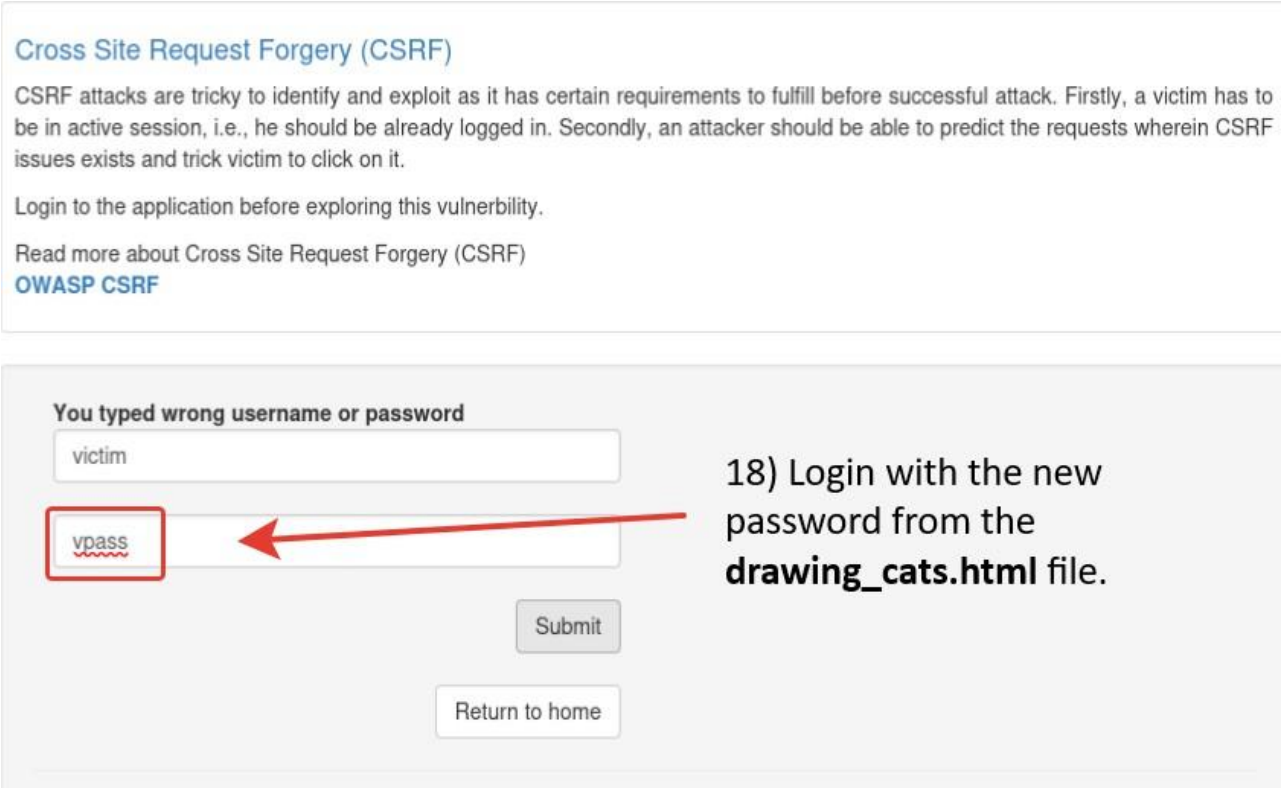


18. We received a message stating that the username or password is incorrect.

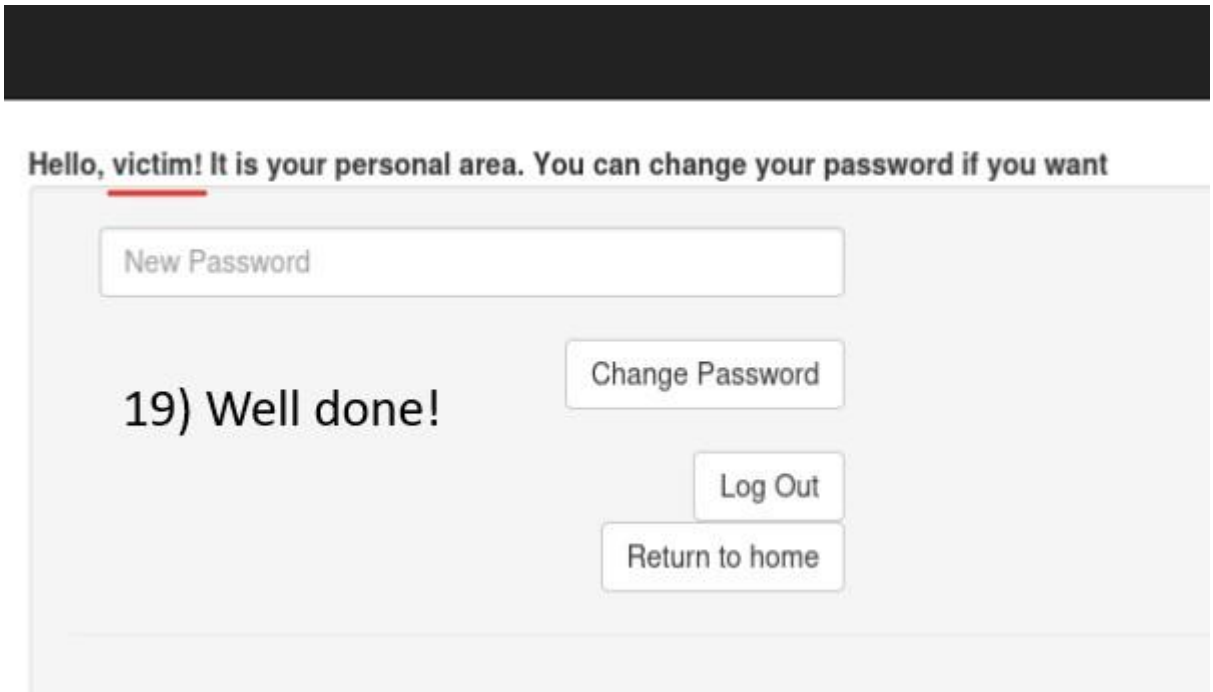
[redacted]

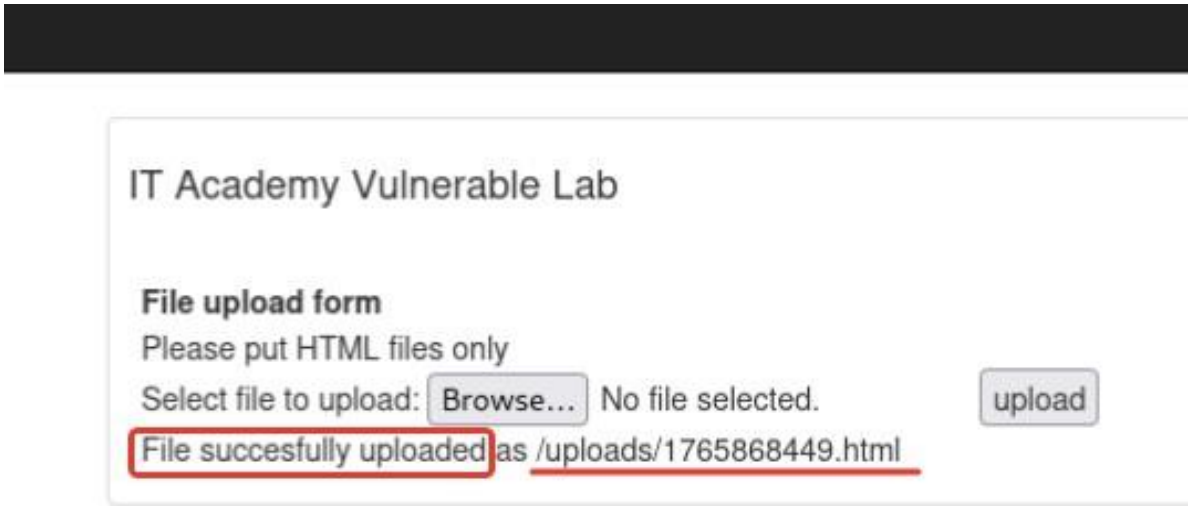


19. Try logging in with the password from the html file (*vpass*)


Cross Site Request Forgery (CSRF)
CSRF attacks are tricky to identify and exploit as it has certain requirements to fulfill before successful attack. Firstly, a victim has to be in active session, i.e., he should be already logged in. Secondly, an attacker should be able to predict the requests wherein CSRF issues exists and trick victim to click on it.
Login to the application before exploring this vulnerbility.
Read more about Cross Site Request Forgery (CSRF)
OWASP CSRF
You typed wrong username or password
victim
vpass
Submit
Return to home
18) Login with the new password from the drawing_cats.html file.

20. Well done!


Hello, victim! It is your personal area. You can change your password if you want
New Password
Change Password
19) Well done!
Log Out
Return to home



21. Submit the `drawing_cats.html` file to the admin.

> **NOTE:** The file was renamed and saved as `1765868449.html`.

22. After a while, try to log in as "`admin`" with the password from the *`Drawing_cats.html`* file.