ECSC Estonia Prequalifier - The gauntlet pt 1

Basic nmap scan:

```
nmap the-gauntlet.hkn -sC -sV
Starting Nmap 7.95 ( https://nmap.org ) at 2025-02-24 11:55 EET
Nmap scan report for the-gauntlet.hkn (10.42.5.146)
Host is up (0.0066s latency).
Not shown: 998 closed tcp ports (conn-refused)
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
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 7.07 seconds
```

At port 5000 there is a login page.
I tried first thing to put `' OR '1'='1` as the password and got logged in as Bob, and with a random password it did not work, meaning this is SQLI vulnerable. Let's change the payload so we get logged in as the admin user.

Username: `' OR username LIKE '%admin%' -- -`
Seems like there is no user with such a name, but Bob works. So let's use sqlmap to dump the database.
SQLDUMP result:

```
+----+-------------------------+----------+
| id | password                | username |
+----+-------------------------+----------+
| 1  | NotThePath-KeepDigging | Bob      |
+----+------------------------+----------+
```

They got us.

We find a cookie like this when logged in:

```
eyJ1c2VyIjp7ImlzX2FkbWluIjpmYWxzZSwidXNlcm5hbWUiOiJCb2IifX0.Z7xJNA.ZrFjYjP
d2hO5RLPlEIIR6X4tNIo
```

It looks like a JWT but only the first part is base64 encoded and is `{"user":{"is_admin":false,"username":"Bob"}}`
Whenever we log in the first part is the same but the second and third parts change. Changing the is_admin value ourselves does not seem to work either, since it logs us out entirely. Seems like the second and third part somehow verify the first part. However the second parts last 2 characters are the only ones that change and the third part is generated each time seemingly randomly.

The second part seems to rotate each second, in the order `A`, `q`, `g`, `w`. When it does this, it also increments the letter before to it (alphabetically) once it completes a loop. It seems the last part is also dependent on the second part, as it stays the same if the second part stays the same.

The second part is the EPOCH timestamp in seconds base64 encoded and the padding removed. The third part is somehow generated based on the first and second part.

When putting a very long name to log in as using an UNION SELECT as the sql payload to log in, it will get compressed with zlib compression. Another interesting thing to note is that the `/` character in base64 encodings seems to get replaced by `_` and the `+` character by `-`. This means all the text should actually be base64 encoded with these replacements. The last part of the token seems to be a SHA1 hash based on its length in bytes being 20, but what exactly it is calculated from is unknown for now, all that is know is that it depends on both the first part and the second part (the timestamp).

Notes about SHA1 hash calculation:

- The password of the user is not used in the calculation, neither is the user's ID

Yeah I reversed all this, turns out its how flask itself handles cookies. I reversed flask. Great. I am so happy.
Anyways use this tool https://github.cosm/Paradoxis/Flask-Unsign along with it's wordlist to crack the secret of `itsasecret` and then create a cookie with `is_admin` set to True.

This gives access to the `/admin` panel, where you can do the ping command. Command injection time.

There is filtering based on illegal characters. Here they are:

- Space - just use `${IFS}`
- `&`, `|`, `` ` ``
- `$()` is allowed, easy

Just upload a netcat binary to the machine by just getting rid of the ip and using `$()`

Yeah part 1 frustrated me so much I speedran part 2 so no more writeup.