Mark Seliternikov
picoCTF - JaWT Scratchpad [400 points]

Challenge details:

JaWT Scratchpad | 400 points

Tags: Category: Web Exploitation

AUTHOR: JOHN HAMMOND

Description

Check the admin scratchpad!

https://jupiter.challenges.picoctf.org/problem/63090/ or

http://jupiter.challenges.picoctf.org:63090

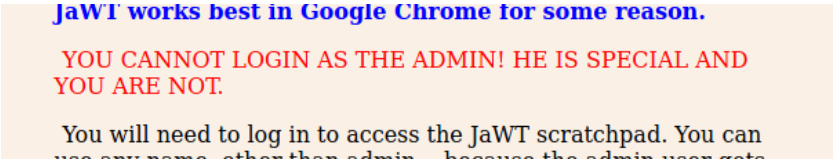
Hints

1 2

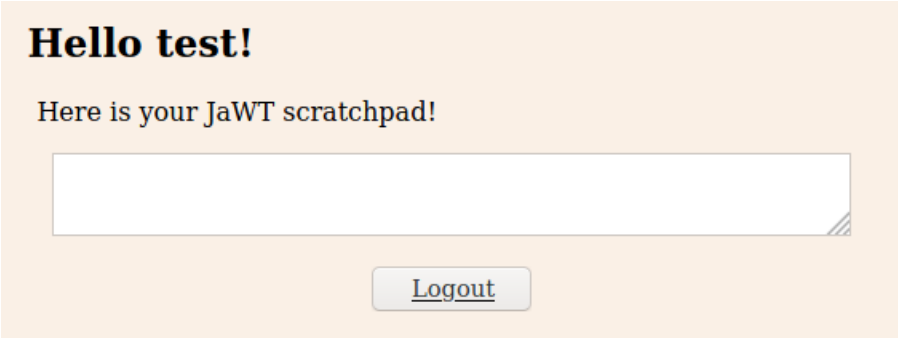
Hints: What is that cookie?, Have you heard of JWT?

When I opened the webpage:

"The admin gets a special scartchpad", well... Obviously I need to log in as admin then in order to solve this. You can't log in as admin, so there needs to be a workaround:



So I tried to log in as “test”, and see what the hints were about… (cookie and JWT).

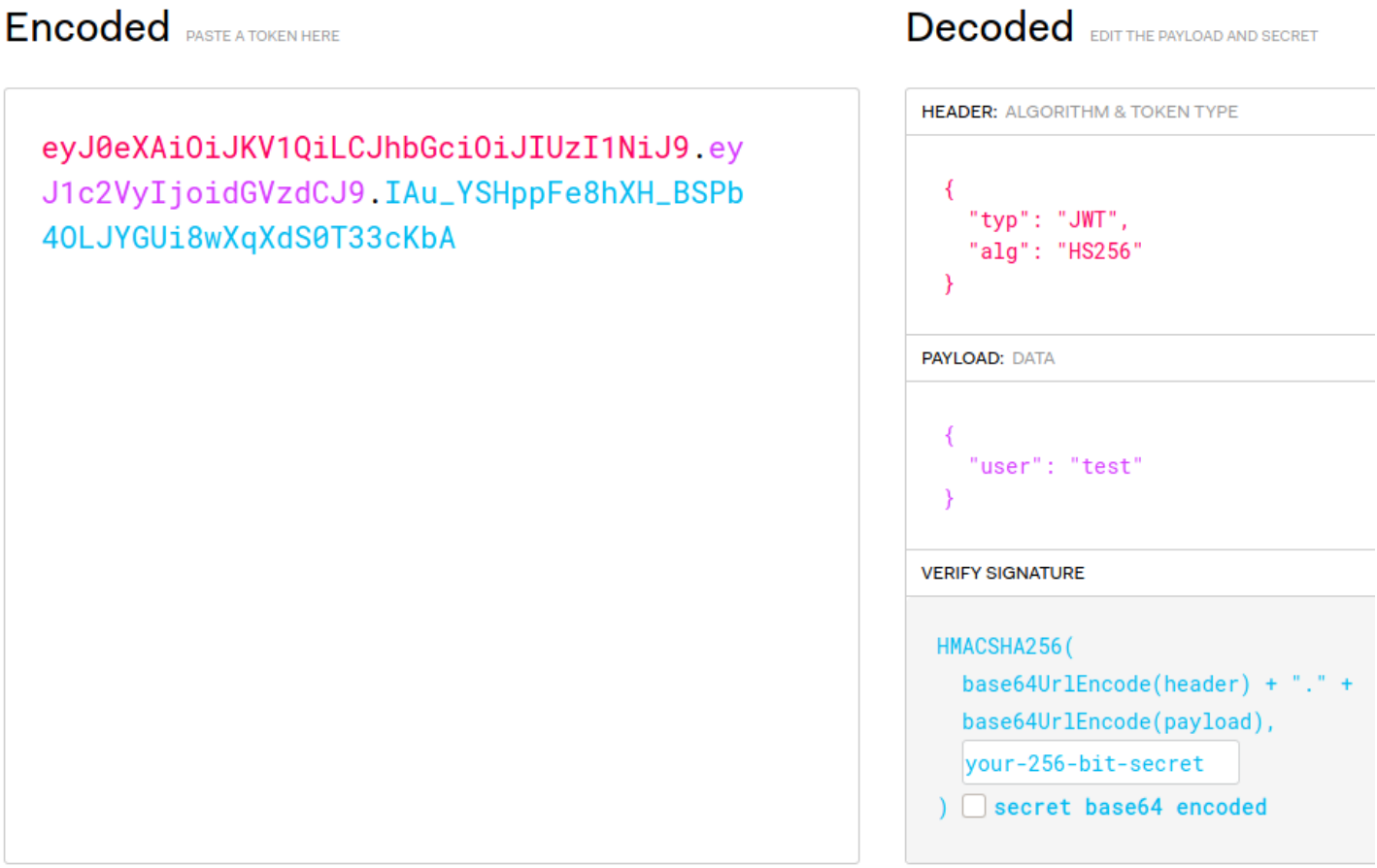


Now that I’m in I’m checking if I have anything in my cookies, and apparently I do!

Name	Value	Do
jwt	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoidGVzdCJ9.IAu_YSH...	jup

After that I needed to research what JWT is, and apparently there’s a link in this very website to a jwt debugger! :)

So I’ve entered my JWT (Json Web Token) to the debugger and this is what I get:



After doing my research about JWT and JWT vulnerabilities I understood how these work.
Basically, There's a header which also has the "alg" part which determines how the signature was formed (what algorithm was used) this header is base64 encoded.
Then there's the payload (the body), which contains the information for the website (in our case it's the user).
The last part is the important part which is the signature which is the the output of the algorithm used to create the signature (in our case its HMAC-SHA256).

The signature is a combination of the header (base64 encoded) + payload (base64 encoded) + secret key (which we don't know……..yet ;) ….)

So from what I've said above it can be understood that we need to find this "secret key".
Before doing that I've tried a few other methods which failed:

1) Just changing the "user" value to "admin", Didnt work :(
2) None attack: The none attack is just changing the "alg" to none, which means that the token doesn't have a signature. It's mainly used for debugging and I hoped that it would be the vulnerability. (it wasn't).

So what I had left is to bruteforce my way to find this secret key! I noticed that there's a link to github in this challenge to "john the ripper" which is used in password cracking:
https://github.com/openwall/john

This is a very sophisticated tool for cracking passwords (I'm glad they were kind enough to introduce me to this tool otherwise this challenge would've been a lot longer!).
Apparently this tool is included in kali-linux! (which is the distro I've recently started using on a VM for CTFs)

After researching about the tool and how to use it I've discovered that it can recognize if a text is a JWT. but you must provide it a list of passwords (if you don't it uses the default which I tried to use first and didn't work).

I wanted to generate a list of passwords and later on realized that it'll take my pc lots and lots of time (I wanted in the 10 char range). But then I thought to myself, "why? I can just google an already made list and call it a day!"

When I tried to look for a list of passwords I stumbled upon a goldmine!
https://github.com/danielmiessler/SecLists

This is tool is amazing because it has lots of password lists! (It's also included in kali's repositories!)
These are the lists I've tried and failed (not including the default one):

```
xato-net-10-million-passwords-1000000.txt
xato-net-10-million-passwords-100000.txt
xato-net-10-million-passwords-10000.txt
xato-net-10-million-passwords-1000.txt
xato-net-10-million-passwords-100.txt
xato-net-10-million-passwords-10.txt
xato-net-10-million-passwords-dup.txt
xato-net-10-million-passwords.txt
```

```
darkc0de.txt
darkweb2017-top10000.txt
darkweb2017-top1000.txt
darkweb2017-top100.txt
darkweb2017-top10.txt
```

Sadly they didn't work, However then I noticed there's a directory called "leaked-databases" which contained actual passwords leaked from databases. I opened it and I've decided to use the largest list I can find there. I noticed there's one in particular that had to be compressed:

```
rockyou.txt.tar.gz
rockyou-withcount.txt.tar.gz
```

This means that this one is quite large… Okay let's see what we get this time!:

```
└─$ john test.txt --wordlist=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA256 [password is key, SHA256 256/256 AVX2 8x])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ilovepico        (?)
1g 0:00:00:01 DONE (2021-04-16 10:02) 0.7092g/s 5246Kp/s 5246Kc/s 5246KC/s iloverob4evax.
.ilovemymother89
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Yep… that's the secret key! (not so secret anymore)

Then what I did was generate a JWT in https://jwt.io/ :

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoiYWRtaW4ifQ.gtqDl4jVDvNbEe_JYEZTN19Vx6X9NNZtRVbKPBkhO-s

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

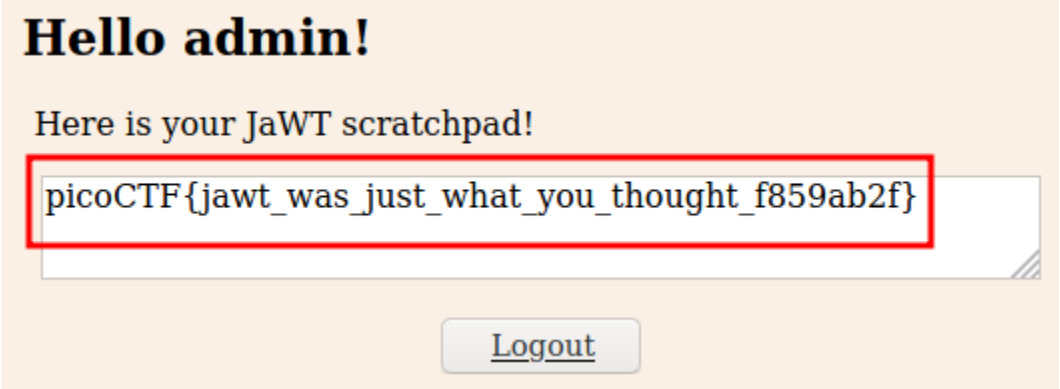
PAYLOAD: DATA

```
  "user": "admin"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  ilovepico
) ☐ secret base64 encoded
```

After that I copied the new generated JWT and pasted it into the jwt cookie… and ta-da!



That’s all folks! :)