**RootMe — A THM Room Write-Up**

My first THM Write-up!

Graphical user interface, application

Description automatically generated

The RootMe room was created to test skills pertaining to NMAP, PrivEsc, and Local File Inclusion.

**Feedback**

Please let me know what you think! This is my first write-up and will be working on more in the future.

When I first started out in THM, I had to use a few lookups and while I was happy to get an answer I was stuck on, I did not understand WHY or WHAT I was actually doing. While doing your own research to get a better understanding is important, why not learn when you are right in the thick of a challenge to keep pushing forward. I made this write-up to do just that and hopefully this helps someone better understand.

**Tools**

I used Cherrytree to conduct this write-up, Kali’s screenshot tool, and Gimp for image editing.

Thank you again and good luck!

**Task 1 — Deploy the Machine**

Task 1 speaks for itself. If you are using your own resources, ensure you are connected through OpenVPN first. Then start machine.

If you have a THM subscription, launch the AttackBox.

**Task 2 — Reconnaissance**

Time for NMAP!

**2.1 Scan the machine, how many ports are open.**

Run the following command:

sudo nmap -sC -sV ‘machine-ip’ -vv

-sC flag performs NMAP Scripting Engine (NSE) which runs default scripts to automate network tasks.

-sV flag enables version detection. Version detection is important when trying to understand a device/network.

-vv flag is very verbose. I like to see what all is happening with the scan when it finds it. This is a personal touch and not needed for the tasks.

Text

Description automatically generated

NMAP Scan

**Scan the machine, how many ports are open?**

\*

**2.2 What version of Apache is running?**

The -sV flag gives us the versions of the host we are scanning. In the following image, we are looking at port 80. Specifically after the word Apache.

Text

Description automatically generated

NMAP Scan

**What version of Apache is running?**

\*.\*.\*\*

**2.3 What service is running on port 22?**

Using your nmap scan you should see the following:

Text

Description automatically generated

NMAP Scan

Looking at port 22, look under the SERVICE header.

**What service is running on port 22?**

\*\*\*

**2.4 Find directories on the web server using the GoBuster tool.**

I use Dirbuster. You are more than welcome to use GoBuster. DirBuster just does a “deeper” dive into the directories.

If you are using DirBuster, you can run the following command:

dirb http://'machine-ip':80 /usr/share/wordlists/dirb/common.txt -w

I added the -w at the end to have dirbuster not stop on WARNING messages.

If you are using GoBuster, you can run the following command:

gobuster dir -u http://'machine-ip' -w /usr/share/wordlists/dirbuster/directory-list-medium.txt

**Side Notes**

Wordlists range on where they are stored so double-check the path of YOUR wordlists. Regardless of which tool you use, take a break and let the tool do the work. Make a drink. It’s 5 o’clock somewhere!

**Find directories on the web server using the GoBuster tool.**

No answer needed.

**2.5 What is the hidden directory?**

Look through your results. There should be a list of [http://'machine-ip:80'/'directories'.](https://medium.com/system-weakness/'machine-ip:80'/'directories'.) What makes directories “interesting” are the status codes that come back after scrapping.

Status code 100s — Informational codes indicating that the request initiated by the browser is continuing.

Status code 200s — Success code when browser request was received, understood, and processed by the server.

Status code 300s — Redirect codes returned when a new resource has been substituted for the requested resource.

AKA…INTERESTING!

Status code 400s — Error codes indicating a problem with the request.

Status code 500s — Server error codes indicating that the request was accepted, but an error on the server prevented fulfillment of the request.

Text

Description automatically generated

DirBuster Scan Result

**What is the hidden directory?**

/\*\*\*\*\*/

**Task 3 — Getting a Shell**

In this task, we are to find a form and upload it to get a reverse shell which will get us the flag.

**user.txt**

Throughout this step, crashes may happen and you may have to restart. Not a big deal. Once the file is uploaded, you can just head to the /upload/ site directory, restart your listening terminal with a new command, and just click the file link on the webpage. This will get you back in. It can be a headache at times, but push through.

Have fun!

**Web Layout**

Navigate to the directory required and you will see the following page:

Graphical user interface, text, application, chat or text message

Description automatically generated

If you head to the /uploads/ page, there is nothing there to use. So we will need to get a form to upload into this page to get to our required user.txt.

Table

Description automatically generated

**Setting Up the Reverse Shell**

If you are using Kali, Laudanum is a collection of files that can be used when SQL injections are found that should already be installed.

If you do not have it, install is:

sudo apt install laudanum

or head to <https://www.kali.org/tools/laudanum/#laudanum.>

Graphical user interface, text, application

Description automatically generated

Laudanum

If you are not using Kali, head to: <https://pentestmonkey.net/tools/web-shells/php-reverse-shell.>

If you head to the file location you will see the reverse shell .php file.

Text

Description automatically generated

To avoid going to the file location you can copy the file with the following command to bring it to where you are located:

cp usr/share/laudanum/php/php-reverse-shell.php . (Yes you need this last dot!)

Now we need to edit the .php file with our ip address and listening port in order to get access. You can use the editor you are most comfortable with. I use nano so my command will be:

nano php-reverse-shell.php

Once inside the file, scroll down until you see the following image:

Graphical user interface, text

Description automatically generated

Obviously where it says “CHANGE THIS”, make the changes as required/preferred. Once changes are made, save the file.

**Setting Up the Listening Port**

When it comes to setting up a listening port, I normally just open it’s own terminal. Do what you prefer here.

Use netcat to setup your listening port with the following command:

nc -lvnp ‘listening port’

-l means it is in listen mode for inbound connections

-v means verbose to get information as it comes in

-n is for numeric IP-addresses.

-p means the port number we want to use to listen on.



Listening port

**Attack Execution**

Now that we have our form filled out and a listening port setup, we are ready for upload! Click browse, select the .php file that we edited and hit the upload button. We should have gotten an error.

Graphical user interface, text, application, chat or text message

Description automatically generated

Translates to PHP is not allowed! Meaning we have to change our filetype because the server does not allow .php uploads. Easy fix! We will just change the filetype. You can rename the file here in this step to something shorter if you’d like. I kept it the same just for this walkthrough.

Use the following command to make the change:

mv php-reverse-shell.php php-reverse-shell.php5



Once completed, upload the new .php5 file and we should see the following image:

Graphical user interface, text, application, chat or text message

Description automatically generated

This translates to: The file has been successfully uploaded! Look!

Click the “Veja!” link and your listening terminal should show you having access! Type in the “whoami” command to verify.

Text

Description automatically generated

Whoami

**Searching for User.txt**

Now that we have access, we need to find our user.txt file. Use the following command to find the file:

find / -type f -name user.txt 2>/dev/null

/ flag means that when the find command is ran, it will find the absolute path of what you are searching for.

-type flag specifies the type of objective you are searching for. Using ‘f’ with it means we want to search for a regular file.

-name flag is used when you have the exact file name.

2> /dev/null is used to discard errors.

The ‘2’ refers to the standard error (stderr) stream.

the > is a file redirection operator. It’s purpose is to direct what is to the left of it (so 2) to the right side of it (so /dev/null). Meaning that any ‘2’ findings will be sent to /dev/null.

/dev/null is a null device which is present in every Linux system and it’s purpose is to discard anything sent to it.

Basically, any errors found during the search, we will not get and don’t want to see anyways.

This will give us the file location of user.txt.



user.txt location

Since we know the location, all we have to do is read the file and voila! Use the following command to reveal the flag:

cat /var/www/user.txt

**user.txt**



user.txt flag

**Task 4 — Privilege Escalation**

Now that we have access, it’s time to PrivEsc!

**4.1 Search for files with SUID permission, which file is weird?**

We need to search files that have SUID permissions. SUID stands for Set Owner User ID. This is a temporary permission given to a user to operate as the owner. This means that when the SUID is set on an executable file, the file will be executed with the same permissions as the owner of the executable file.

Use the following command to find binaries with SUID permissions.

find / -perm -u=s -type f 2>/dev/null

/ flag means that when the find command is ran, it will find the absolute path of what you are searching for.

-perm flag we indicate that we are going to look for permissions.

-u=s flag we indicate that we are going to look for setuid user permissions.

-type f flag to find those files with SUID permissions.

2>/dev/null again to send the errors to /dev/null.

Text

Description automatically generated

Scan Result

The scan has a lot to siff through. Try running the scan on your own Linux machine. There will be just a few that will be different.

**Search for files with SUID permission, which file is weird?**

/\*\*\*/\*\*\*/\*\*\*\*\*\*

**4.2 Find a form to escalate your privileges.**

Because we are dealing with a Linux machine, we need to find a location that has a list of binaries that can be used to bypass local security restrictions. GTFOBins is a collection of legitimate functions of Unix binaries that can be used to break out of restricted shells, escalate or maintain elevated privilege's, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.

Head to [https://www.gtfobins.github.io](https://www.gtfobins.github.io/) and click on the Python Binary. Select SUID at the top of the page.

“If the binary has the SUID bit set (In our case, /usr/bin/python) it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access as a SUID backdoor.” BOOM! We have a way in!

**4.3 root.txt**

Now that we found a way to get access to root, we need to execute a command to access root. AKA PrivEsc!

Use the following command:

python -c ‘import os; os.execl(“/bin/sh”, “sh”, “-p”)’

-c flag specifies the command to execute.

import os — pefroms an import of the operating system.

os.execl — First, the “exec()” executes the current process image with a new process image. Think of it as overlaying the existing process and not creating a WHOLE new process. The new-process image file can be one of three types:

1.) An executable binary file in XCOFF file format

2.) An executable text file that contains a shell procedure (only the execlp and exdecvp subroutines allow this type of new-process image file)

3.) A file that names an executable binary file or shell procedure to be run. <- (This is what will happen in our challenge)

There are variants of “exec” that we can use. “L”, “V”, “P”, and “E”:

L — a list of command line arguments supplied to the command to produce a single-column output.

V — uses an array format when the parameters that are to be sent to the exec() process are variables, or not known in advance.

P — will use the PATH variable to locate the executable file.

E — used to pass a list of environment settings for the new process image.

“/bin/sh” — is used in conjunction with the execl command to provide the PATH.

“sh” — to run as a shell command.

“-p” flag used to give permission to run with SUID permissions.

Text

Description automatically generated

root.txt flag

Thank you again and feedback is welcomed! Both good and bad!

Graphical user interface, application

Description automatically generated

PWN’d!

**Link to the room!**

[THM](https://tryhackme.com/room/rrootme)