Pandora (Linux)

New things I learned/used from this challenge

Enumeration

This was my first challenge where I had to enumerate SNMP.
I used snmpwalk to do this after I installed the MiBs to be able to get useful information.

From the snmpwalk I was able to recover a password for one of the users.
The snmpwalk and grep commands I used to find this can be found in my OSCP Cheat Sheet.

```
# Much faster to  use . The "-Cr1000" parameter does 1000 request a second.
snmpbulkwalk -Cr1000 -c public -v2c <IP_ADDRESS> . > yoursnmpwalk.file

# Optional default tool but much slower
snmpwalk -c public <IP_ADDRESS> -v2c
```

Lateral Movement/usr/bin/pandora_backup

From SNMP we were able to get an initial foothold with the “daniel" user.
Looking through the web application files, I was able to see that the Pandora CMS was running on localhost port 80 and was only accessible locally.
I was able to make an ssh tunnel to route that localhost port 80 traffic from the victim's remote machine to my local attacking machine's port 8000.

```
# Local Port Forwarding/Tunneling
ssh <user>@<target> -L <local_port>:127.0.0.1:<remote_port>
```

Once I had access to the Pandora control-panel login, it gave me information on the version that Pandora was running.
Doing some research into some vulnerabilites for this version, I was able to find an unauthetnicated RCE through a SQL injection from one of the extensions running on Pandora.
From here I was able to get command execution, I setup a reverse shell and now I succesffuly moved over the the “matt” user.

Priv Escalation
This is where thing's got tricky.
As the “matt” user on the victim's machine, I ran LinPEAS and found an unknown SUID binary named /usr/bin/pandora_backup
I encoded the binary as base64 and brought it over to my Kali machine to do some reverse-engineering on the binary.

Here's what Ghidra was able to decompile

```
bool main(void)

{
  __uid_t __euid;
  __uid_t __ruid;
  int iVar1;

  __euid = getuid();
  __ruid = geteuid();
  setreuid(__ruid,__euid);
  puts("PandoraFMS Backup Utility");
  puts("Now attempting to backup PandoraFMS client");
  iVar1 = system("tar -cvf /root/.backup/pandora-backup.tar.gz /var/www/pandora/
pandora_console/*");
  if (iVar1 == 0) {
    puts("Backup successful!");
    puts("Terminating program!");
  }
  else {
    puts("Backup failed!\nCheck your permissions!");
  }
  return iVar1 != 0;
}
```

We see from the system() command that it runs a tar file of the pandora_console using the tar command.

What is vulnerable in this binary is that when tar is being used, the command uses the relative path instead of the absolute path.

We can do PATH hijacking

◇Basically the PATH environment variable is a list of directories to search through from beginning to end to find the matching executable.

◇So if we create a malicous file that has the same name as the command without the absolute path, we can run that file with escalated priviliges.

◇If i make a file named “tar” and inside of it contains “/bin/bash”, and then make it executable.

◇Then if i change the path via

```
export PATH=<DIRECTORY_WITH_MALICOUS_FILE:$PATH
```

◇Then all we need to do is run the binary, it will use the PATH variable that we changed to look in the directory of the malicous file, since the malicous file is also named “tar” it will execute it, and then we have a root shell!