Oopsie

Machine: OOPSIE
IP address: 10.10.10.28

The machine is a web based application as evident from its nmap scan.
It has only port 22 and port 80 open.

LEARNINGS: web, SQL enumeration, backdoor and reverse shell spawning in php with file upload, privilege escalation using built in programs in the host machine.

user_flag

On opening the webpage at http://10.10.10.28, we see that it is a normal webpage with no input fields anywhere.

But when we use burp to crawl the site, we observe that there is a sub directory by name /**cdn-cgi** which contains another sub directory called /**login**.

So now if we go to the full url at **http://10.10.10.28/cdi-cgn/login** , we see a login page.

At the start, we observe that the name of the site is MEGACORP, so some previous creds from the machine ARCHETYPE might be used here.

Now, for the username we use **admin** and password is **MEGACORP_4dm1n!!** taken from the ARCHETYPE machine.

After successful login, we observe that we see 4 sections in the main page. out of that, 3 sections are table entries from some database. one section is blocked saying that only **super admin** has access to it.

If we observe the URL on the Accounts, it resembles a simple SQL injection vulnerability. It takes only one parameter called **id**.

Now, we have to automate this sql injection using the BURP SUITE's intruder to test a sequence of numbers from 1 to 100.

Out of all the received responses, id number 30 has the name called super admin. We note down its ACCESS ID as 86575 and name as super admin.

But previously we saw that the upload section is accessible only via this super admin role.

Hence, by observation we note that this access is managed using cookies.

In cookies, we see that there are 2 cookies named role and user.
User is a number and role is name of the role.

After changing the cookie values to super admin details, we unlock the upload option..

Now, an upload option generally is vulnerable to file injections and backdoors with out proper scrutiny or sanitization from the server side.

As the website is php based, we upload a php backdoor with reverse shell using the upload option.

On our system, we set up a netcat listener and wait for a connection

Now, we traverse to /uploads/filename.php to activate the backdoor.

We see a reverse shell on our terminal.

In the shell, we locate user.txt and get the flag.

root_flag

To get the root flag of the machine, we need to enumerate the system more.

In the www/html/cdn-cgi/login folder we find that there is a db.php file which might be have been used by the web server to connect to their sql client.

So we view the file and find a username and password
by name **robert** and password:**M3g4C0rpUs3r!**

We use the previous shell we obtained and escalate it to full fledged shell using
python3 -c 'import pty; pty.spawn("/bin/bash")' .

This will create a full fledged shell and enable us to switch to user **robert**.

Now, after switching to robert, we find that the user robert is part of a group called bugtracker

Now, we use the find command to list all the files which are accessible by the users of this group.

We come across an interesting compiled file called bugtracker in /usr/bin

If we do a ls -la on it, we find that the file is created by root but can be executed by anyone in the bugtracker group.

on executing it, we are prompted to enter a number as bug id.
when we enter a number, we find that the number we entered is treated as a file name in the path /root/reports...

The user robert doesn't access to visit any file in the /root folder.

But the program Bugtracker which contains root privileges, can read any folder or file in the /root.

Also the function called by the bugtracker program is “cat” to read the bug id file.

Hence we can leverage this to read the root flag.

In the prompt for bug id, if we enter **../root.txt** , we are presented with the root flag!!