Bash Injection

Introduction

Category: Miscellaneous
Solves: 54
Points: 50

This is a pretty straightforward challenge, requiring us to execute command injection to obtain the flag stored in the system. Although it is simple, I still got stuck a few times, which I will discuss in detail below.

Solving the challenge

At first glance

When we first start the challenge, we are greeted with a Netcat command. By running the command, we are then prompted to enter a username and password.

Since we have no inkling as to what the credentials are, we can first try to determine what this programme does. Thus, I used `user:pass` to try to log in, and the following output is returned:

> [exe] -> bash -c './login.sh "user" "pass"'
> [out] -> Invalid username.

From this, we can see what was executed along with its output.

We can observe that a script, 'login.sh' is run to validate the credentials. Our username and password are provided to the script as its first and second arguments, respectively.

At this point, we notice that there seems to be no form of input validation. Thus, there is a possibility for us to hijack the `bash -c` command and execute whatever we want. The password field is suitable, since it will make it easier for us to run the script normally and append our intended command.

Finding the vulnerability

When deciding on the password input, we must ensure that it causes the resulting command to be syntactically correct. In this case, our password must end with `'"` to match the `"'` that is appended to our password before it is passed to the script.

Thus, I tried logging in again with username `user` and `pass"' && ls '"` to test it out.

The following output is given:

> [exe] -> bash -c './login.sh "user" "pass"' && ls '""'
> [out] -> Invalid username.

When I reached this point, I got stuck for a while as I was unable to receive an output from the `echo '""'` portion. After a few minutes of fiddling, I realised the problem was that I was running the extra command **outside** of the `bash -c` command. Consequently, the output was not captured and returned to us.

Therefore, I modified the password input and tried again. This time, I tried `pass" && echo "hello!!!"''"`. (note that the password still ends with `'"`) The following output is returned:

> [exe] -> bash -c './login.sh "user" "pass" && echo "hello!!!"""""'
> [out] -> Invalid username.hello!!!

As we can see, the `echo` command worked! Thus, we found a command injection vulnerability and can now exploit it to obtain the flag.

Exploiting the vulnerability

Unfortunately, I got stuck here for a while again as I was trying to get a netcat reverse shell working. I couldn't do so as there was no `/tmp` folder on the system, thus I was unable to create a named pipe on the system for the reverse shell to work.

It took me several minutes to give up on the idea of a reverse shell, before I tried a simple and primitive `/bin/bash -i`, which worked perfectly!

> [exe] -> bash -c './login.sh "user" "pass" && /bin/bash -i"""'
> [out] -> Invalid username.bash: cannot set terminal process group (1): Inappropriate ioctl for device
> bash: no job control in this shell
> bash-4.4$ ls
> ls
> login.sh
> run
> bash-4.4$

We can now see that there are 3 files in the shell's current directory. We already know `login.sh`, which is what was used to verify our credentials. There is also a `run` file, which is the file run when we connect to the server using Netcat, upon inspection.

Finding the flag

I got a little lost again here, as I was looking for a `flag.txt` or some other sort of file that contained the flag. I looked around the system to no avail, before I decided to inspect the `login.sh` file.

Upon inspection, we found all we needed to know! The script contained the correct credentials, as well as the flag that we are looking for. It would have saved me quite some time if I just decided to inspect the `login.sh` file at first, but it is what it is... :)

```
cat login.sh
#!/bin/bash

username="$1"
password="$2"

[ "$username" = "friedrice4"   ] && {
 [ "$password" = "maggimee2" ] && {
    printf 'Login complete: greyhats{86sh_1n73ct10n_y6333}'
 } || {
    printf "Incorrect password."
 }
} || {
 printf "Invalid username."
}
```

Flag: `greyhats{86sh_1n73ct10n_y6333}`