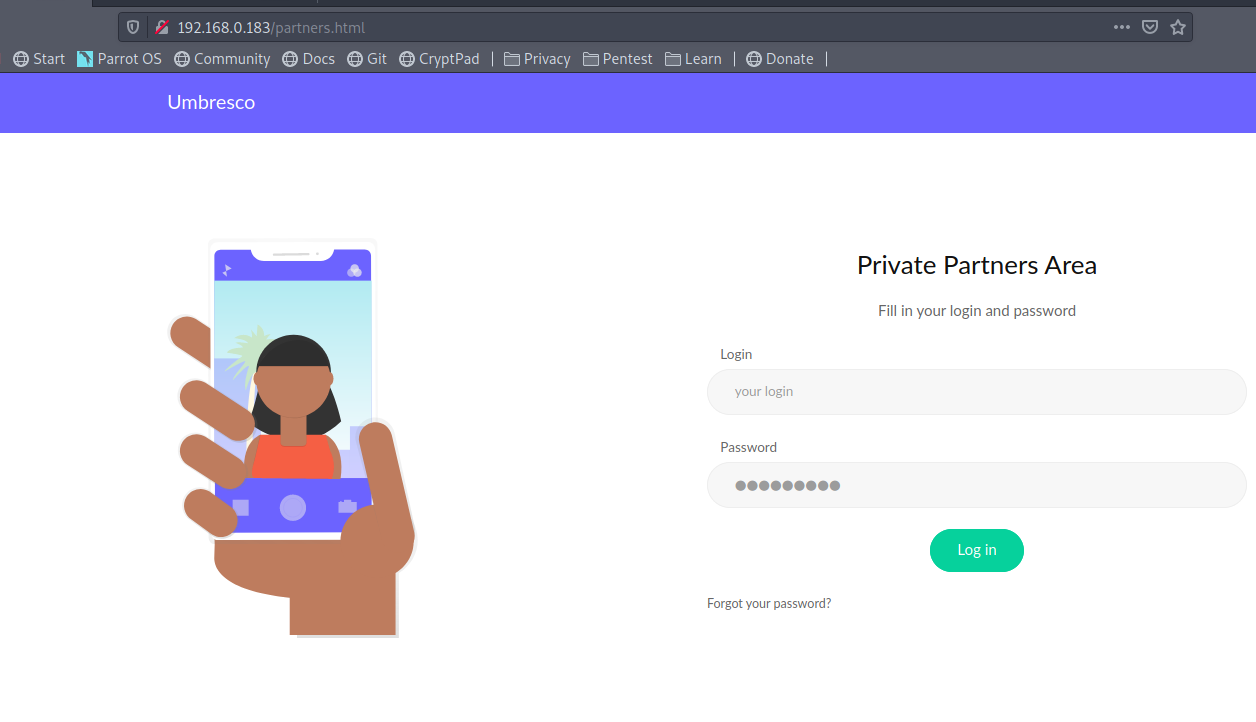
**Vulnerability/Penetration Testing**

1. **Command Injection**

Command injection is an attack that aim for executing arbitrary commands on the host operating system from the vulnerable application. Command injection attack are possible especially due to the usage of executing system command in the web application directly without sanitizing the user input. Attacker can utilize this vulnerability to send a reverse shell, uploading backdoor into the victim machine, stealing credentials from the database, etc.

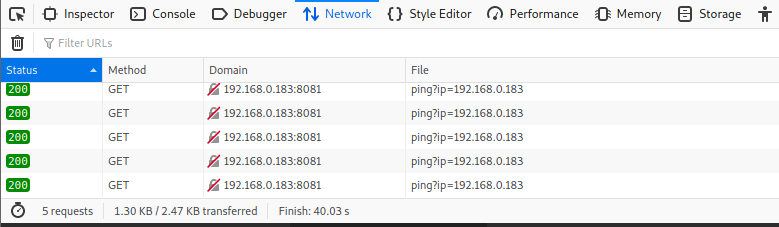
We will be utilizing firefox to browse the website, using firefox inspect element network tab to monitor for any request that made, burpsuite to intercept the packet and using repeater to modify and send the packet, and using python3 to create a script to make command injection to the vulnerable part easily, bash scripting to create backdoor, python SimpleHTTPServer to host a webserver to upload backdoor, netcat to get the reverse shell. We will be hosting our vulnerable web service in our Parrot Security Machine to show the Proof of Concepts.

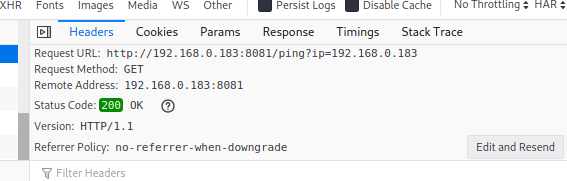
**Step 1: Browse the website login page**

****

So the image above shows that we have found the private partners login page of Umbresco IT Sdn Bhd website.

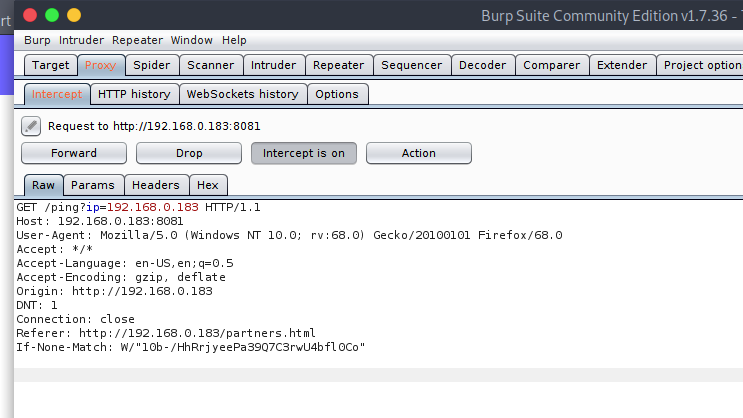
**Step 2: open firefox inspect element network tab to monitor requests made**

****

****

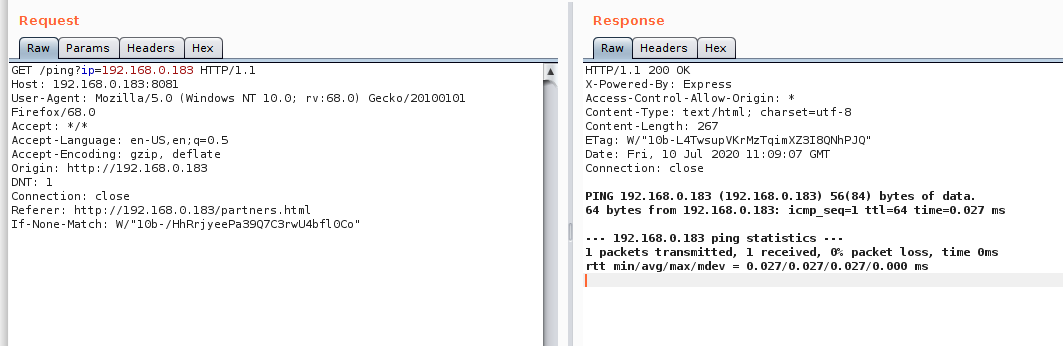
From the monitoring result, we notice that the web service keep on sending GET request to the domain “192.168.0.183:8081/ping” with the parameter “ip=192.168.0.183” every 10 seconds. The status code shows 200 means the request has been succeeded. So, from here we make an assumption that the “192.168.0.183:8081/ping” ip parameter might be executing system commands to perform the ping.

**Step 3: Launch burpsuite to intercept the packet**

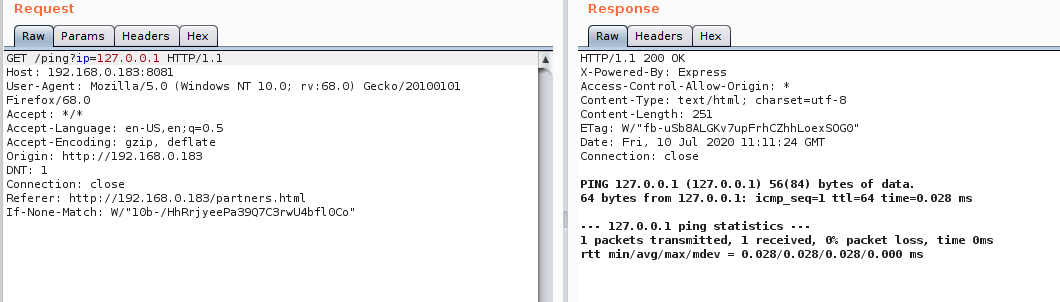
****

The image above shows that we have intercept the GET request made to 192.168.0.183:8081/ping. So now we will be sending the packet to repeater.

**Step 4: Editing the request packet with repeater to get the result**

****

So after we send the packet requests to repeater we click on the go button to forward the request to get the output. So same as our assumption the GET request made are executing system commands as we can see at the right screen which is the response screen it shows the output of ping command.

****

We try to change the parameter value of “ip” to 127.0.0.1 and forward the GET request, the response screen shows the result of the ping pinging to 127.0.0.1. From the result we can see that it only ping once, so we assume that the ping flag that it set is “-c 1” which only ping once.

So from the result we try to craft the possible os command that it execute in the remote server.

Possible OS command execute: **ping -c 1 < ip value from GET request >**

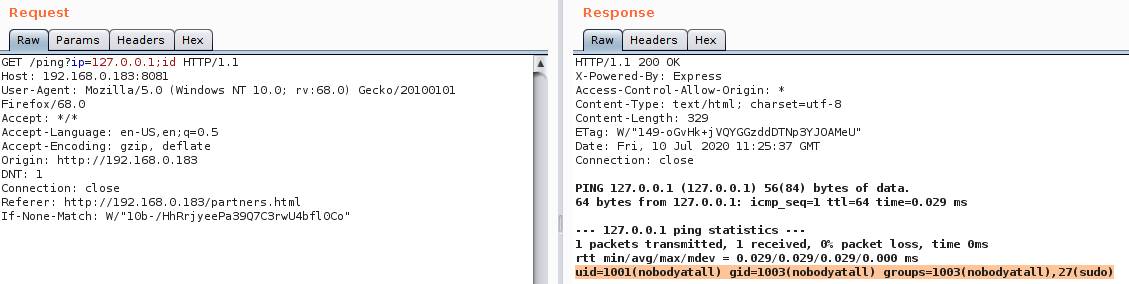
**Step 5: Injecting OS Commands to test the vulnerability**

From the perspective that we craft the possible OS command that inject in the web service, we can try to execute another OS command in the “ip” parameter value part. From our reconnaissance result, we know that the remote server operating system is Linux OS, so we try to inject linux OS command to test it out. We know that in linux we can execute multiple commands in a single line command execution with ‘;’. So we try to craft our payload that we will be testing out.

ip value from GET request: **127.0.0.1;id**

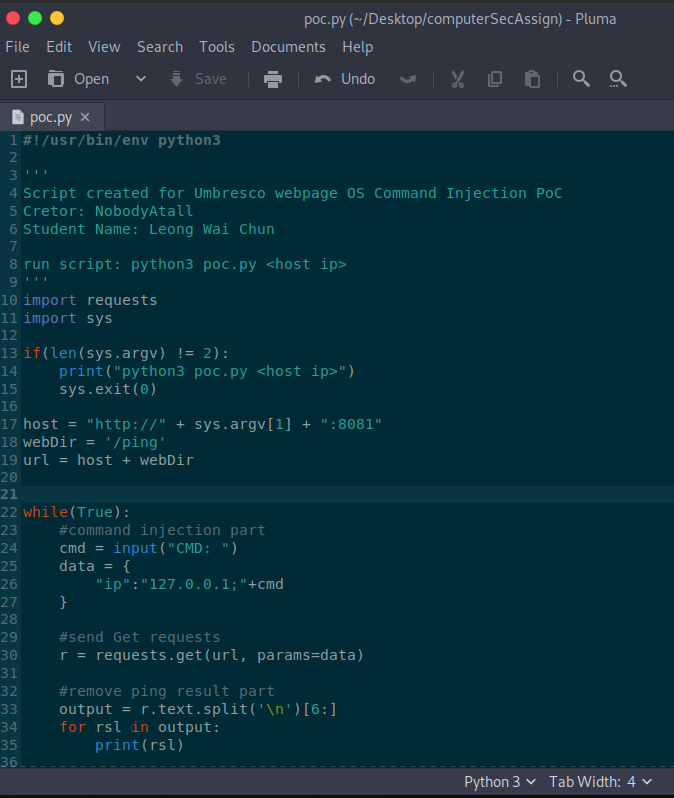
Payload: **ping -c 1 127.0.0.1;id**

So as we can see here if we send the ip parameter value with “127.0.0.1;id”, the system command that will be executing on the remote server will be “ping -c 1 127.0.01;id” which means that the server will first execute the “ping” command then it will execute the “id” command. Let try it out in the burpsuite repeater on our assumption.

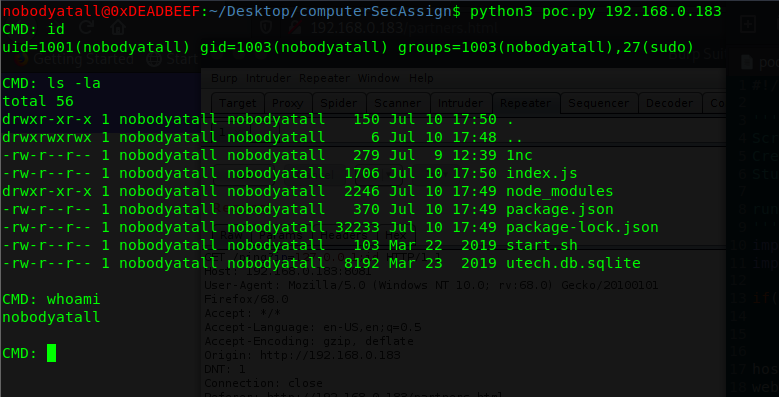


Just as we expected the “ip” parameter is vulnerable to OS command injection. We had gotten our “id” command result (the one that highlighted on the response screen).

**Step 7: Creating and testing python3 script to inject OS commands easily**

****

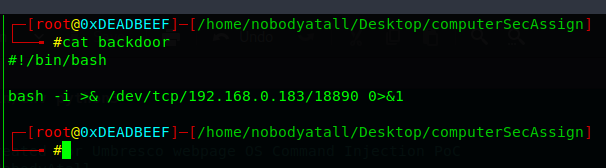
The image above shows the python3 script that we have created to perform the OS command injection on the “ip” parameter in “/ping”.



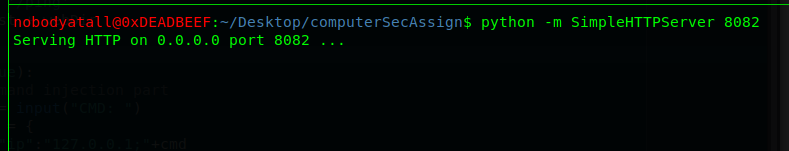
So the image above shows that our script is working and we have successfully perform OS command injection in the web service’s server.

**Step 8: Upload backdoor and get reverse shell on the remote server**

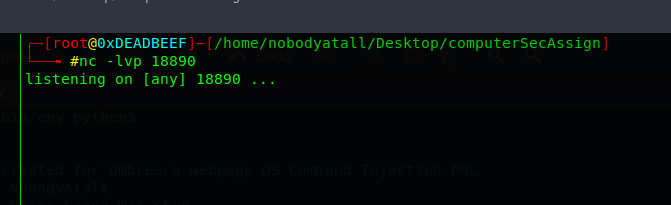
So let try to create our backdoor bash script to get a reverse shell when we executing it.



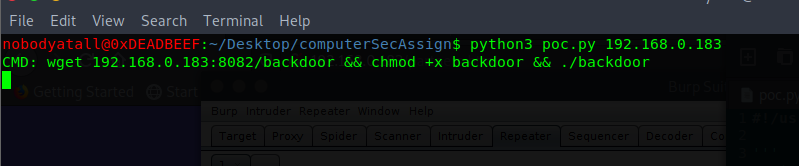
The image above shows the payload that we will be executing on the remote server.



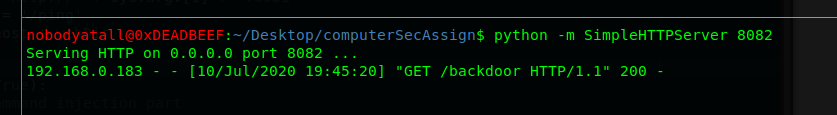
We will be opening our SimpleHTTPServer web server on port 8082 to upload our backdoor later.



We try to open our netcat listener first to capture the reverse shell return when we execute the backdoor script.



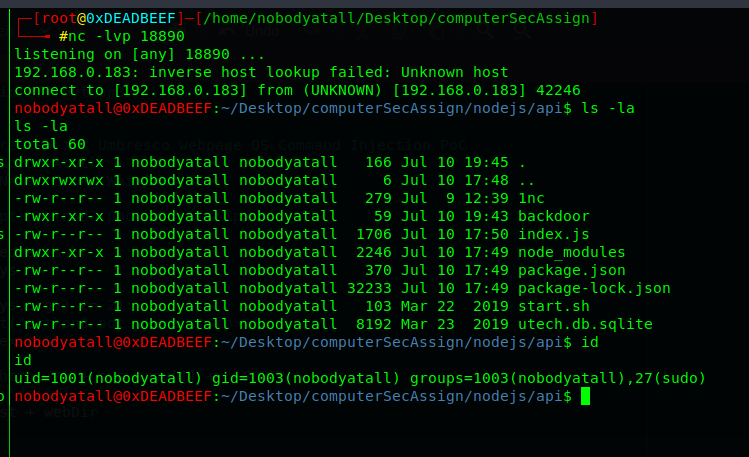
So now let execute the script. The 1st part command wget will be use to download the backdoor script to the remote server, the 2nd part command chmod will be setting execute permission for the backdoor, then the last one will be executing the backdoor



The above image shows that the backdoor has been successfully download to the remote server.



Now our backdoor had just connected to our netcat.



Now we can execute system command in the remote server with the reverse shell.

**Summary**

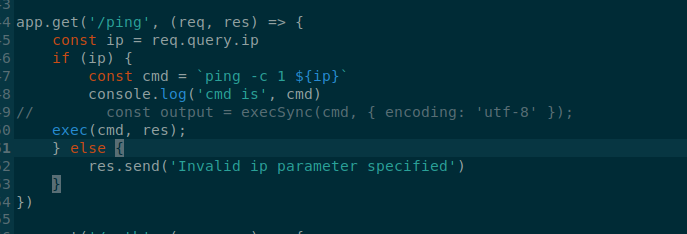
Tool: Parrot Security OS

Method: burpsuite, firefox inspect element (network tab), netcat, bash scripting, python scripting, SimpleHTTPServer

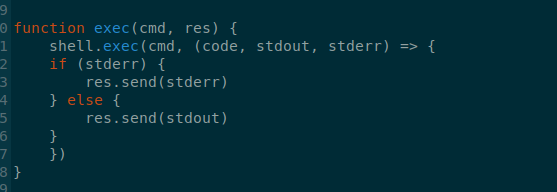
Aim: Inject OS commands in vulnerable web application and getting reverse shell

**Countermeasures**

1. **For Command Injection**
2. **Validate the IP value with RegEx (Regular Expression)**

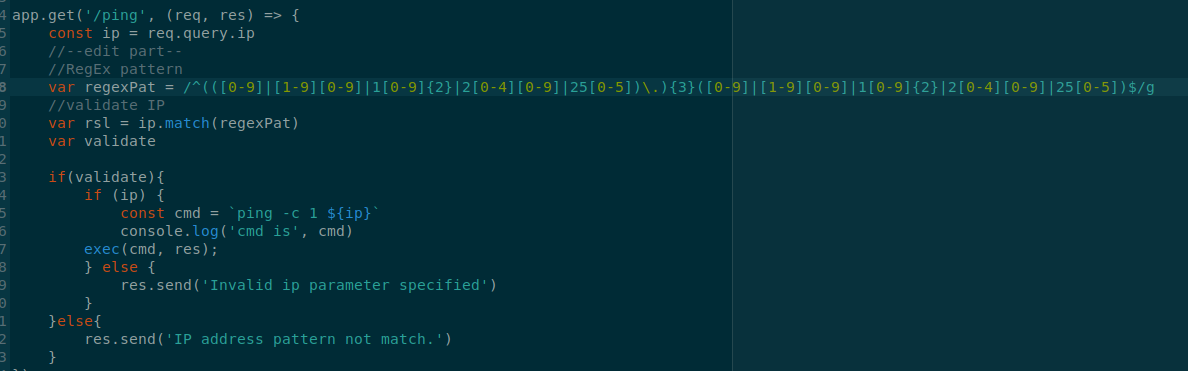
****

When the IP value pass to the backend service, it shows that the IP input value is not validate and it directly pass into the exec() function

****

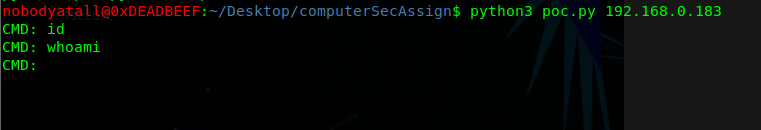
Over here we found that the exec() will be using shell.exec() to perform system command execution.

**Step 1: Edit the code to validate IP address with RegEx**

****

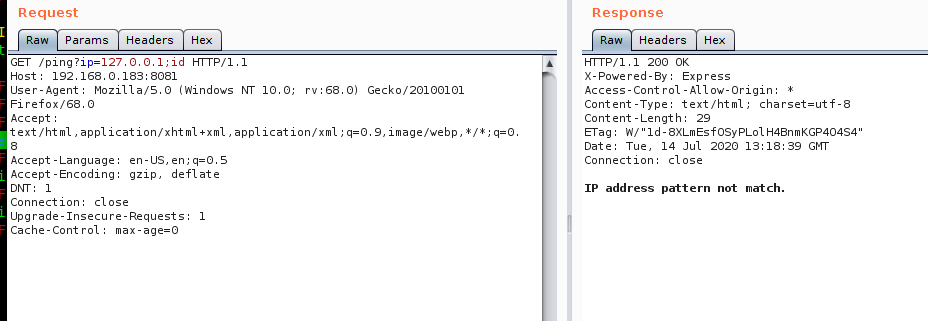
The image above shows that we have added the RegEx pattern matching, if the rsl Boolean is false then it will show “IP address pattern not match.”. So, the attacker will not be able to inject any system commands from the parameter.

**Step 2: Try to run poc.py python script to test the vulnerability**

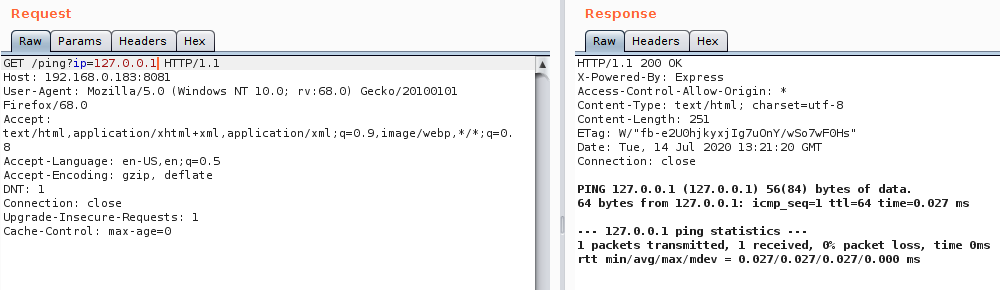
****

We notice that when we execute the system commands with our poc.py python script, it does not return any value to us. So we will be using burpsuite to check what happened when we send the particular command.

**Step 3: Check the response when the payload sent with burpsuite**

****

So, the result just as we expected the attacker cannot abuse that ip parameter to perform system command injection as our RegEx pattern does not match the IPv4 address pattern. The response return a message “IP address pattern not match” which means that at the backend service, it detected that the IPv4 value does not match.

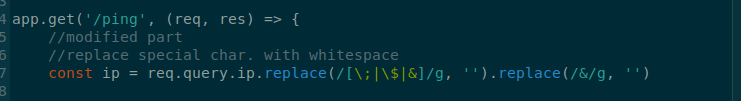


The image above shows that if we place only the IPv4 address only, the backend service will allow the ip value and execute the “ping” command to ping the particular IP address. So, the command injection vulnerability has been fixed.

1. **Remove Special Characters from the input IP value**

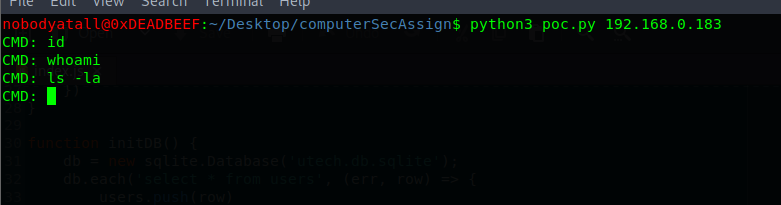
An attacker in order to perform command injection to abuse the vulnerability, the attacker need to use special characters like “&”, “|” and “;” to execute another system command in the server side. Another method to prevent this attack is by removing special characters that pass to the backend service.

**Step 1: Edit the code to remove special characters**



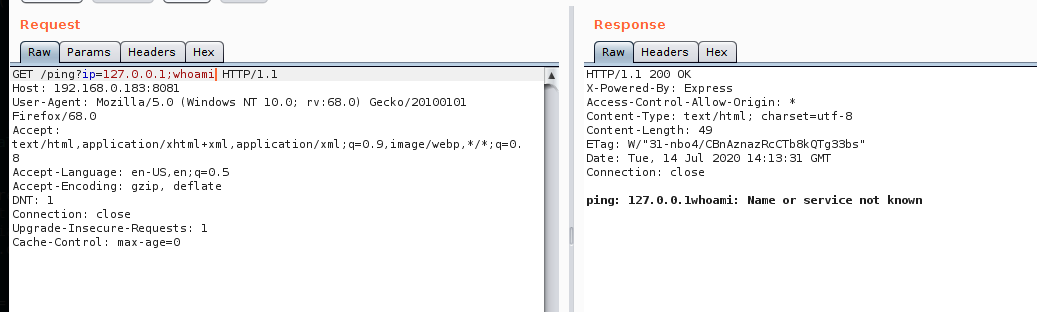
So, the image above shows that the input IP value, if it find any special characters entered it will remove it and replace it with whitespace.

**Step 2: Try to run poc.py python script to test the vulnerability**

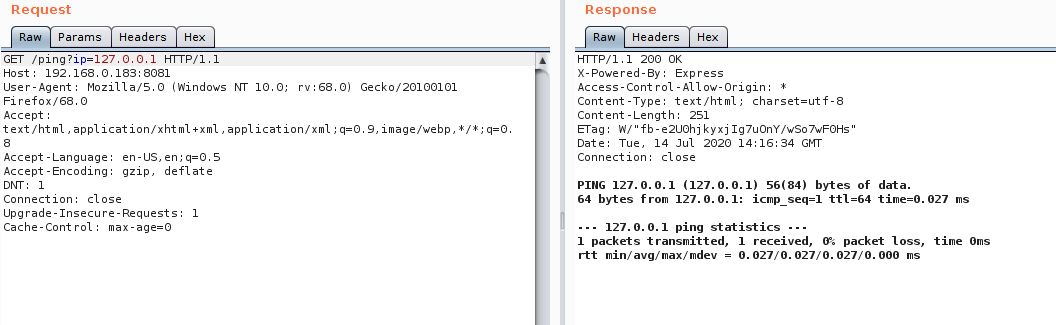
****

We notice that when we execute the system commands with our poc.py python script, it does not return any value to us. So we will be using burpsuite to check what happened when we send the particular command.

**Step 3: Check the response when the payload sent with burpsuite**

****

So, as we expected when we inject the payload <IP address>+<special character>+<system command>, it does not work. The response screen shows that the “;” is removed and it shows a ping command error which telling us the name or service not known, so the attacker cannot abuse the particular method to perform system command execution to abuse the vulnerability in the ip parameter.



The image above shows that if we place only the IPv4 address only, the backend service will allow the ip value and execute the “ping” command to ping the particular IP address. So, the command injection vulnerability has been fixed.