BruteForce Password Cracker For Websites

Abhinav Srivastava (109187992)

<u>Table Of Contents:</u>

1. Problem Statement:

The project deals with designing and implementing a basic password bruteforcer for websites that is "smart" about what kind of passwords it will try on a website's login form. More specifically, the password bruteforcer will collect all keywords from a website and try these keywords, as well as specific variations of those, as passwords in a login form. Specifically, given a website as a target, the tool should:

- Autonomously crawl the website following the links of each page
- Collect the text of each page and break it up into keywords
- Create permutations of the collected keywords (uppercase, lowercase, leetspeak, bigram of words)
- Autonomously find the login form of a page
- Be able to understand when a username/password combination succeeded or failed

2. Design:

The project involves two steps : Finding the login form and secondly cracking the password. The website can have many links and each link needs to be searched for the login form. So its becomes important to find the all the links and do the BFS and stop where it finds the login form. To make sure it has found the login form, I have planned to work with Selenium + Python and use the utility of "Xpath" to find whether form elements - text field, password field and button field are present or not. So all the links will be traversed and tracked by Xpath to see for the form elements. If present it will save that URL. Once I have the form elements, next step is to crack the password using the keywords from the website. For that, BFS will trace all the links of the website and store all the html text of all the links into a string. Later that string is tokenized and stored in a list which will be used later to generate lowercase, uppercase, leetspeak and bigrams. For leetspeak, I have used basic leet [2]. Also, from the list I am removing all the stop words that are present in ntlk english corpus. For bigrams, I am grouping words in pairs and finding all such pairs.

For the setup, I have prepared the exact same site as provided in the github link by the professor but in flask framework siince it does not need apache or anything to host/serve itself.

3. Software Requirements:

NLTK : NLP python package to use for stop words and to tokenize

Platform : Any OS (mentioned packages must be installed)

Programming Language : Python (v2.7)

Flask : Python framework for website creation and hosting on localhost

Selenium : Selenium webdriver to open the web, find the form and extract the content

4. Installation and Execution

Flask Framework : pip install flask

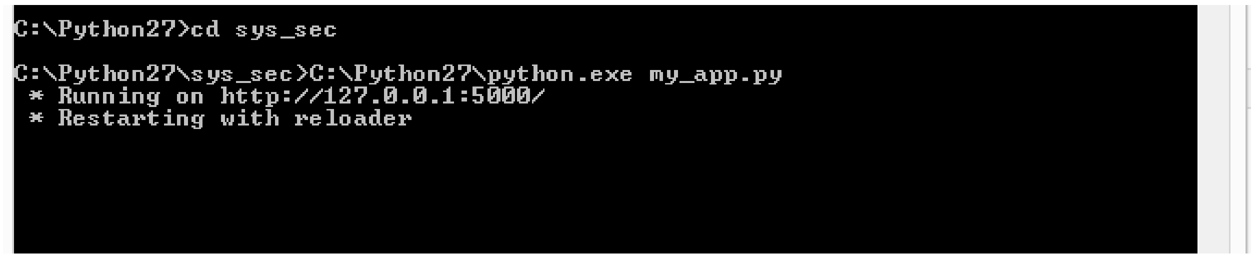
Selenium : pip install selenium

Python : Installed from python website [3]

NLTK : Installed from nltk website [4]

Execution :

To start the server script and enable the webservice:



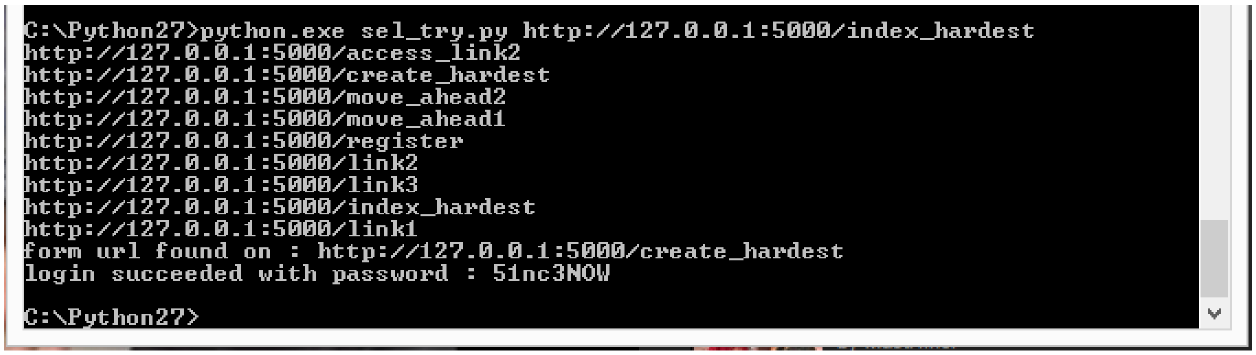
```
C:\Python27>cd sys_sec

C:\Python27\sys_sec>C:\Python27\python.exe my_app.py
 * Running on http://127.0.0.1:5000/
 * Restarting with reloader
```

C:\Python27>python.exe sel_try.py http://127.0.0.1:5000/index_hardest

5.Ouput Snapshots:

1. Server Side for hardest website

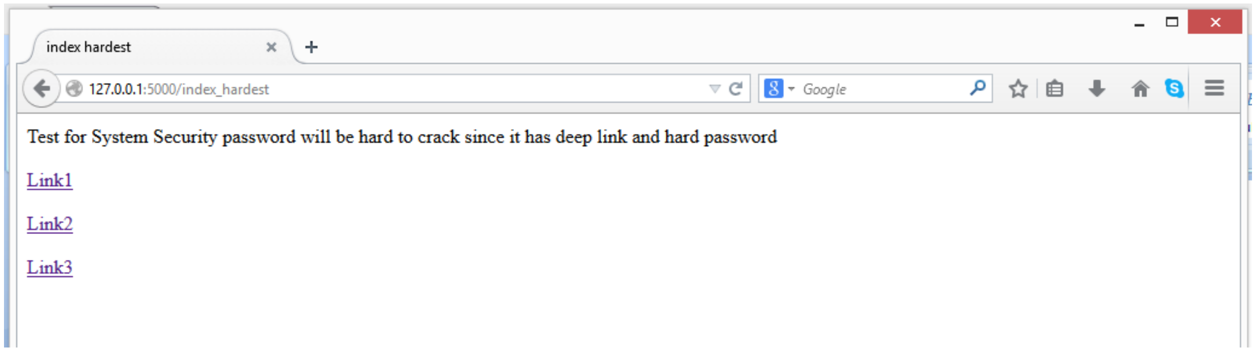


```
C:\Python27>python.exe sel_try.py http://127.0.0.1:5000/index_hardest
http://127.0.0.1:5000/access_link2
http://127.0.0.1:5000/create_hardest
http://127.0.0.1:5000/move_ahead2
http://127.0.0.1:5000/move_ahead1
http://127.0.0.1:5000/register
http://127.0.0.1:5000/link2
http://127.0.0.1:5000/link3
http://127.0.0.1:5000/index_hardest
http://127.0.0.1:5000/link1
form url found on : http://127.0.0.1:5000/create_hardest
login succeeded with password : 51nc3NOW

C:\Python27>
```

2. Client Side for hardest website

a) Index Page


index hardest

127.0.0.1:5000/index_hardest

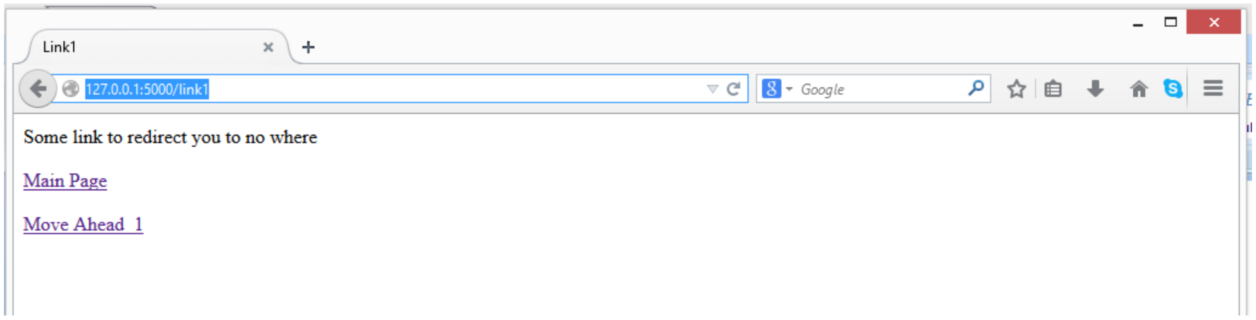
Test for System Security password will be hard to crack since it has deep link and hard password

Link1

Link2

Link3

b) Clicking - Link1


Link1

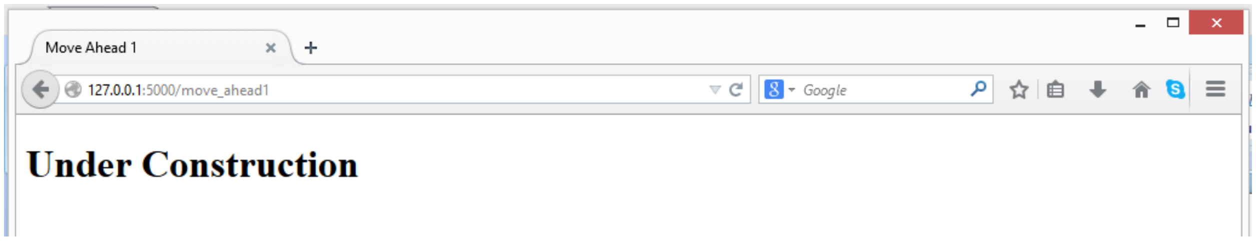
127.0.0.1:5000/link1

Some link to redirect you to no where

Main Page

Move Ahead 1

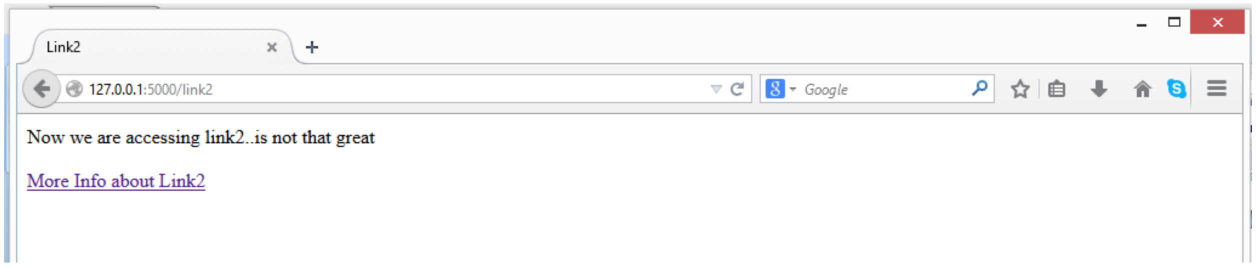
c) Main Page will direct back to a) and clicking Move Ahead 1 will result in:


Move Ahead 1

127.0.0.1:5000/move_ahead1

Under Construction

d) Clicking Link2

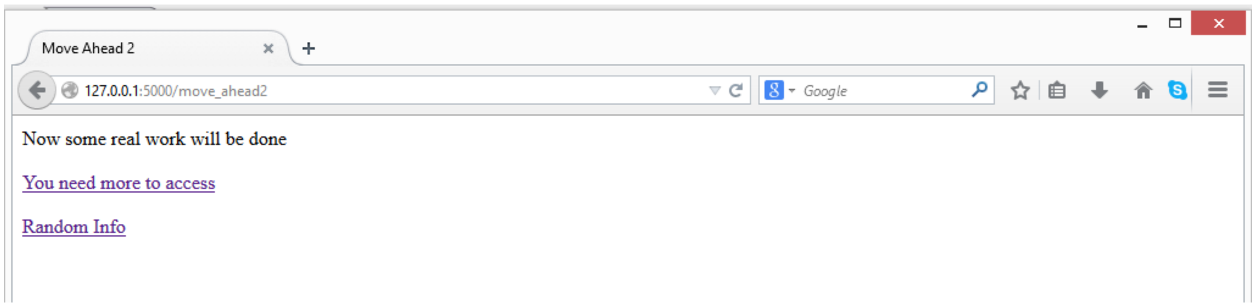

Link2

127.0.0.1:5000/link2

Now we are accessing link2..is not that great

More Info about Link2

e) Clicking Move Ahead 2

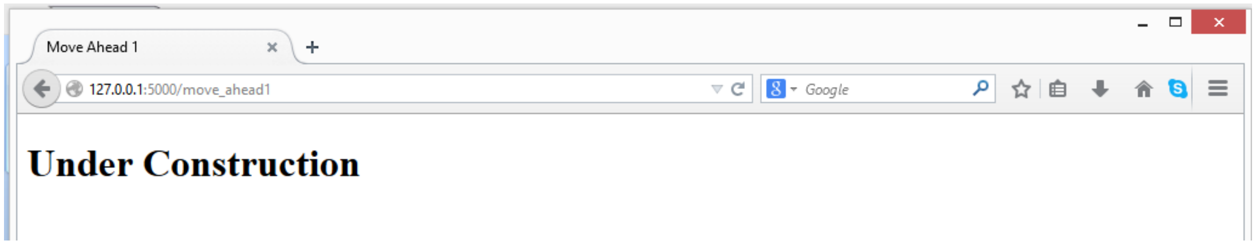


Now some real work will be done

You need more to access

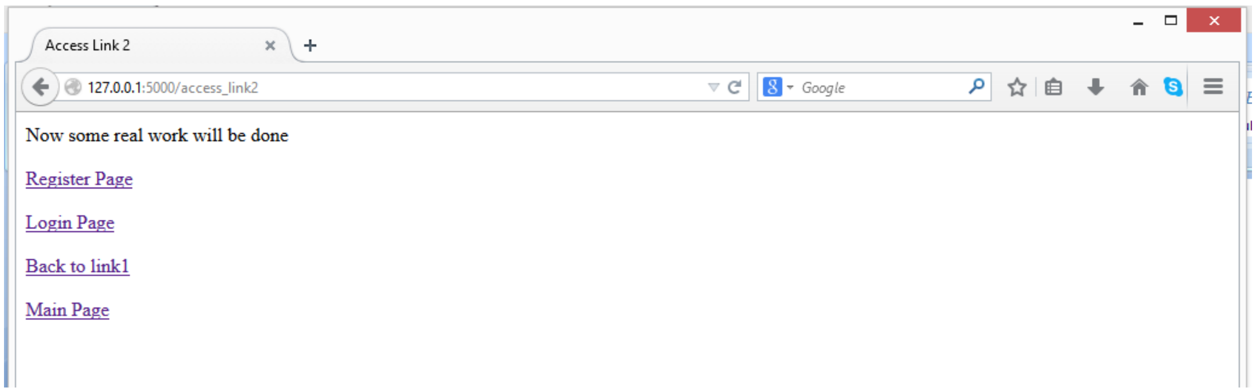
Random Info

f) Random Info results in:



Under Construction

g) After clicking You need more to access :



Now some real work will be done

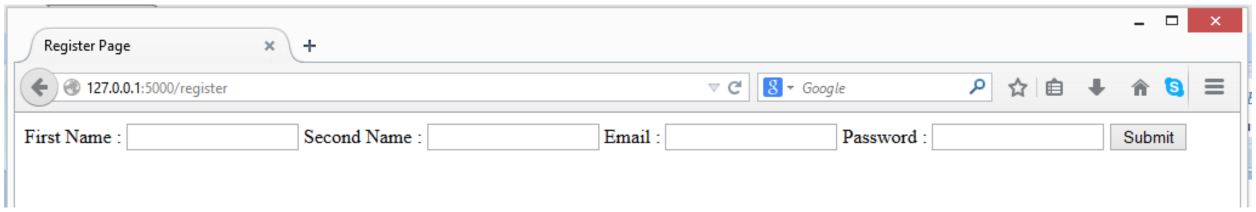
Register Page

Login Page

Back to link1

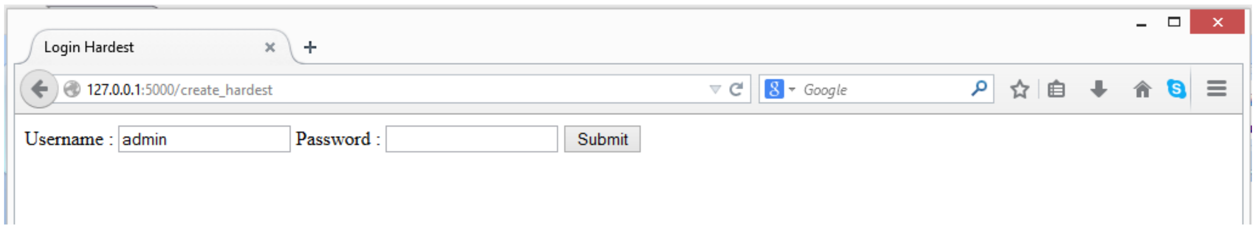
Main Page

h) Register Page



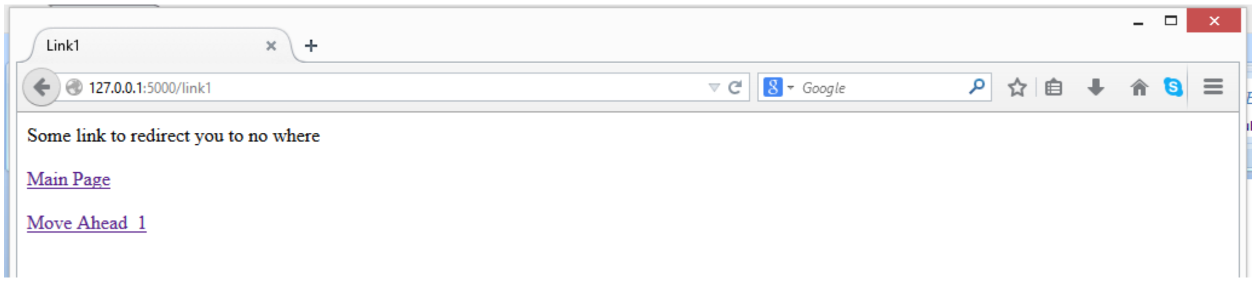
First Name : Second Name : Email : Password :

i) Login Page



Username : admin Password : Submit

j) Back to link 1

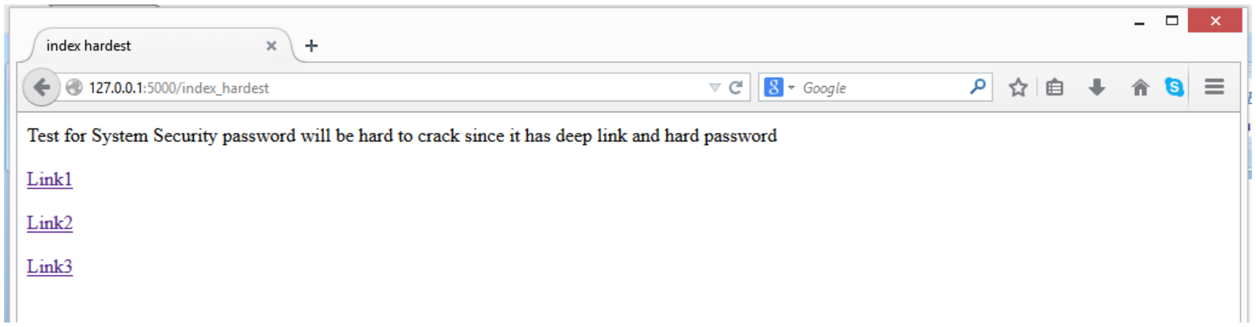


Some link to redirect you to no where

Main Page

Move Ahead 1

k) Main Page



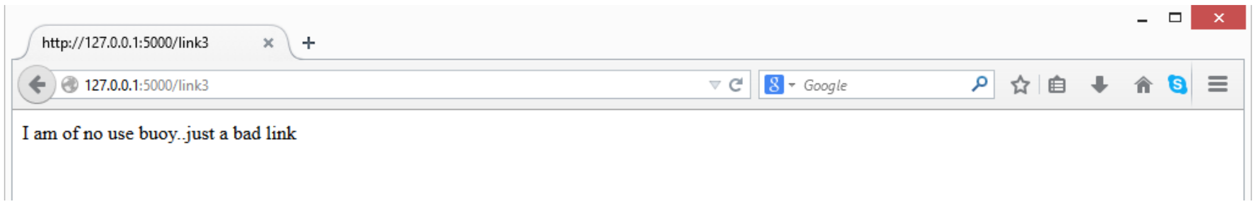
Test for System Security password will be hard to crack since it has deep link and hard password

Link1

Link2

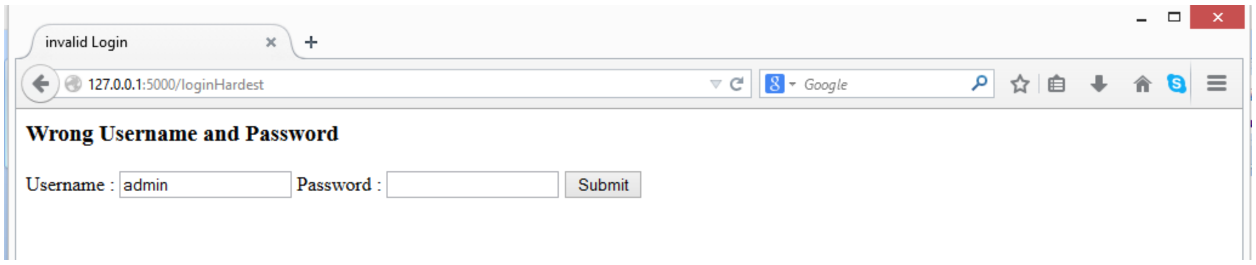
Link3

l) Link3



I am of no use buoy..just a bad link

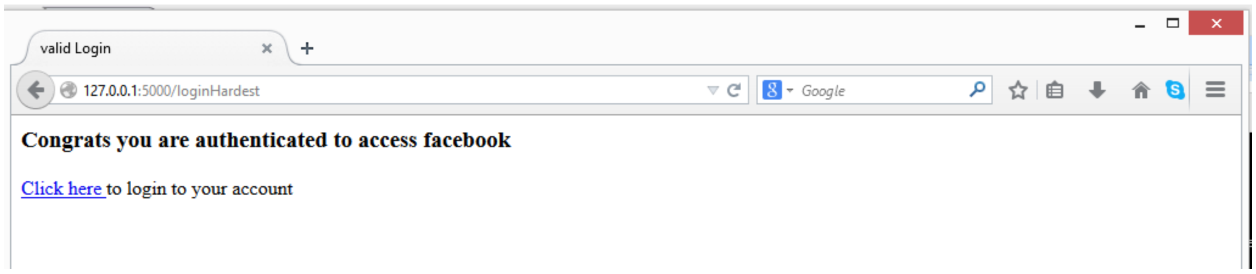
m) Wrong Username and Password



Wrong Username and Password

Username : admin Password : Submit

n) Correct Credentials



Congrats you are authenticated to access facebook

Click here to login to your account

and clicking on "click here" will direct to the facebook login page.

6. Conclusion:

The bruteforce password cracker work well on websites demonstrated in the project which has some keyword from the html text as password. However, it won't work for all the websites due to the structure of the website suppose if multiple pages have login page, tool will stop at the first link that has got the login page. Also, since it consist of cardinal words, uppercase, lowercase, leetspeak and bigram pair of words, the tool takes lot of time since it may have to match all the mentioned word cases to crack the password. Since it has been built on Flask framework so adding new routes is not a hard task fi somebody wants to add new links to the website to make it even more hard to crack but that will also multiply the amount of time tool will take.

7.References:

[1]NLTK Tutorial : http://www.nltk.org/

[2]Basic Leet Speak : http://www.robertecker.com/hp/research/leet-converter.php?lang=en

[3]Python Installation : https://www.python.org/downloads/windows/

[4]NLTK Installation : https://pypi.python.org/pypi/nltk

[5]Selenium Webdriver API : http://selenium-python.readthedocs.org/en/latest/api.html

[6]Selenium Tutorial : http://selenium-python.readthedocs.org/en/latest/getting-started.html