House Snake

Introduction

This is House Snake, a HTTP Basic Authentication Brute Forcing tool that I created using Python, frustration, and a lot of swearing. In Theory it will use a list of username and a list of passwords to brute force past a HTTP Basic Authentication login. I named it house snake for two reasons; It was programmed in Python, and much like a house snake, it isn't very threatening.

HTTP Basic Authentication

HTTP Basic Authentication is the simplest technique for access control to web resources. It does not make use of cookies, session identifiers or login pages, instead it makes use of standard fields in the HTTP header.

The Beginning

I started the process of building House Snake by first finding out what HTTP Basic Authentication is. For that I turned to the all knowing one, Google. I read a few blogs posts about it but didn't really get to understand it until I found and read RFC 7617.

After reading up on HTTP Basic Authentication, I realised I would have to be able to test my creation. I figured I had two options, either I find a vulnerable website willing to let me legally brute force them, or I could host my own web server.

I decided it would probably be better to host my own server. The reasons for this are threefold; firstly I don't know which of the websites I could potentially use are trustworthy, secondly I didn't want to run the risk that my cobbled together code would crash their servers, and lastly I thought it would be a good learning experience if I did it myself.

It took me a few days to research the setting up of a web server. Admittedly my own server didn't need to be any good, but I studied best practices all the same. I started off with only a single website. After I had it up and running I did some more research into how to implement HTTP Basic Authentication. For anyone interested, Digital Ocean has very good guides that cover both topics.

The next step in my journey was to find out how to actually interact with my site using python. I took some time to look up various python modules that I could possibly use. I found a few that could do the job, but settled on *requests*.

The Middle

With the preemptive research out the way, I proceeded to actually do some programming. I started out slow, just trying to get a hang of *requests.* The basic script sent a GET request to the website and provided a username and password. In the event that the username and password were valid, the website would return a status code of 200 OK, otherwise a 401 UNAUTHORIZED was returned.

The basic script used a single URL and a single username and password pair. When that worked, I stepped it up to use a password list. After getting that to work, I copied and changed the function a bit to allow for a username list.

Using more than one URL meant I had to host more than one website on my server. I put that aside for the time being while I learned how to allow the script to make use of terminal arguments. I used the *argparse* module to allow the user to make use of some arguments when running the script.

I followed an online guide and created a few simple scripts in order to get the hang of *argparse.* I got the hang of the basics of the module and was able to incorporate it into my project.

I now had a program that makes use of terminal arguments to select from a few options, either a single username password pair, or lists of either. These could be used in few different combinations if the user desired. I also added a verbose option to display all usernames and passwords attempted.

I needed the script to also use a list of targets. To allow for this I needed to host more than one website on my server. After a bit of research and trial and error I was able to host five websites with unique domain names.

I edited the functions that allowed for username or password lists in order to be able to make use of a list of target urls.

The End

The powers that be required that my project accept a .csv file containing targets, and export the results to a json file. This required some extra research and practice with the appropriate modules. It took some doing, but I was finally able to get my project to conform to the requirements set out for us.

House Snake is now able to brute force a single target, or multiple targets, using a combination of usernames and passwords, either singular or from provided lists. It can alternatively be given a single list containing username and password pairs.

Usage

House Snake is written in python 3, but two of its cosmetic modules are not compatible with python 3. To avoid any unnecessary errors, it should be run using python 2, or made into an executable via chmod.

When you run it, you need to provide it with some parameters. These are as follows:

- -u/-U: username list or a single username
- -t/-T: a list of targets(in csv format) or a single url
- -p/-P: password list or a single password
- -i: a single text file containing usernames and passwords

There are also some optional arguments that can be used:

- -h: will display the help
- -v: (verbose) will display all usernames and passwords attempted
- -o: will output the results to a specific output file in the json format

I have added a few cosmetic touches to make the results easier to read. Successful results are printed in green. When verbose is used, unsuccessful results are printed in red. If the script is unable to connect to a host an error message is printed in yellow. Another error message, printed in red, informs you if there are no username or password pairs found for a host.

Conclusion

Working on this project has been a great learning experience. Even though it was frustrating at times, I did enjoy it. I like working on a problem and finally finding a solution to it. House Snake is not very pretty or very useful, but it works as intended, and it is mine.