Developing an Anti-Phishing Tool –

“Here Phishy Phishy”

Eric Holguin | Mohammad Hossain

Introduction

Phishing is a prime example of social engineering and can cause a lot of harm to an end user or organization and it is an attack that can be difficult to defend against. This project aims to build an anti-phishing tool that will help prevent such phishing attacks.

There are a numerous number of phishing attacks/techniques that are used, such as, spear phishing, session hijacking, email phishing, content injection, web-based delivery, SMS phishing, Voice phishing, Malvertising, and many others. It is difficult to build a tool that would mitigate all of these attacks, however, building a tool that can inform users while at the same time alert them of suspicious or possible phishing attacks would provide usefulness to many organizations and to the cyber security community.

There exist websites and APIs that already play a role when it comes to anti-phishing, such as checking well known phishing URLs, malware detectors, tools that check emails or mitigate against man in the middle attacks. These are a few of the already existing works and some prior research projects that we will consider and reference when building our very own anti-phishing tool.

The goals we have set in mind:

- Command-line interface
- Mitigate against Domain Name Phishing Attacks
- Monitor suspicious web-server activity
- Allow users to report phishing

The plan in mind is to use Python build the command-line interface while keeping the potential of a GUI in mind. We chose to mitigate against domain name phishing attacks because it is a very common attack and can easily deceive a user. We plan to use machine learning techniques that are used for phishing detection and in addition look into sites with well-known phishing sites lists. We also want to leverage any potential free APIs to improve the tool even further. With Python we will monitor web server logs in search of suspicious activity which might be helpful for the user to know, such as excessive traffic from one source IP. Finally, we want to allow the user to submit a report of any potential phishing to any recognized websites that help prevent phishing. We plan to make this tool extensible in order to allow for future work to be done and additional phishing mitigation techniques can be added.

Implementation

The application was built using python 3.6 in addition to some valuable python libraries and the help of the PhishTank API.

We used the PyInquirer library in order to build a simple to use command line interface for the user, adding colors that makes it easy for the user to tell the difference between commands and program output.

Using the Scapy library we were able to log the incoming queries from port 53, which logs DNS queries. This helped us and the user to see what URLs your system makes, and we can manually check if they look suspicious.

We created a developer account with PhishTank that allowed us to use their free API in order to check phishing URLs against their database. We send a POST request, formatted in JSON and we then parse the response to check whether the URL we sent is in their database or not and whether it has been flagged as phishing or not.

Finally, we used the sklearn library which provides the algorithms necessary for machine learning detection. We used the dataset downloaded from PhishTank, which included 7000+ URLs, each with their own label, 1 as phishing or 0 as not. We then defined the features we deemed important, such as:

- Presence of @
- Presence of //
- Count of dots in the URL
- Number of delimeters
- If the IP address is found inside the URL
- Presence of -,
- Count of subdirectories
- Filename extension
- Subdomain
- Count of queries.

Each feature was setup as a method that could parse a URL to extract the information, this then allowed us to create a dataframe of the extracted features and then pass this to the decision tree classifier. Roughly half of our data was labeled as a phishing URL and the other half as not. With a decision tree with a maximum depth of 10 we used a training sample of 80% and a testing sample of 20%. Our accuracy calculation was 89.6% This was a good score, but we found there still exist false positives and false negatives when testing certain URLs.

User Instructions

***This was tested and ran on MacOS Mojave**

To install the application on your own system first download it from GitHub and install the python libraries required.

```
# git clone https://github.com/ZugNZwang/HerePhishyPhishy.git
```

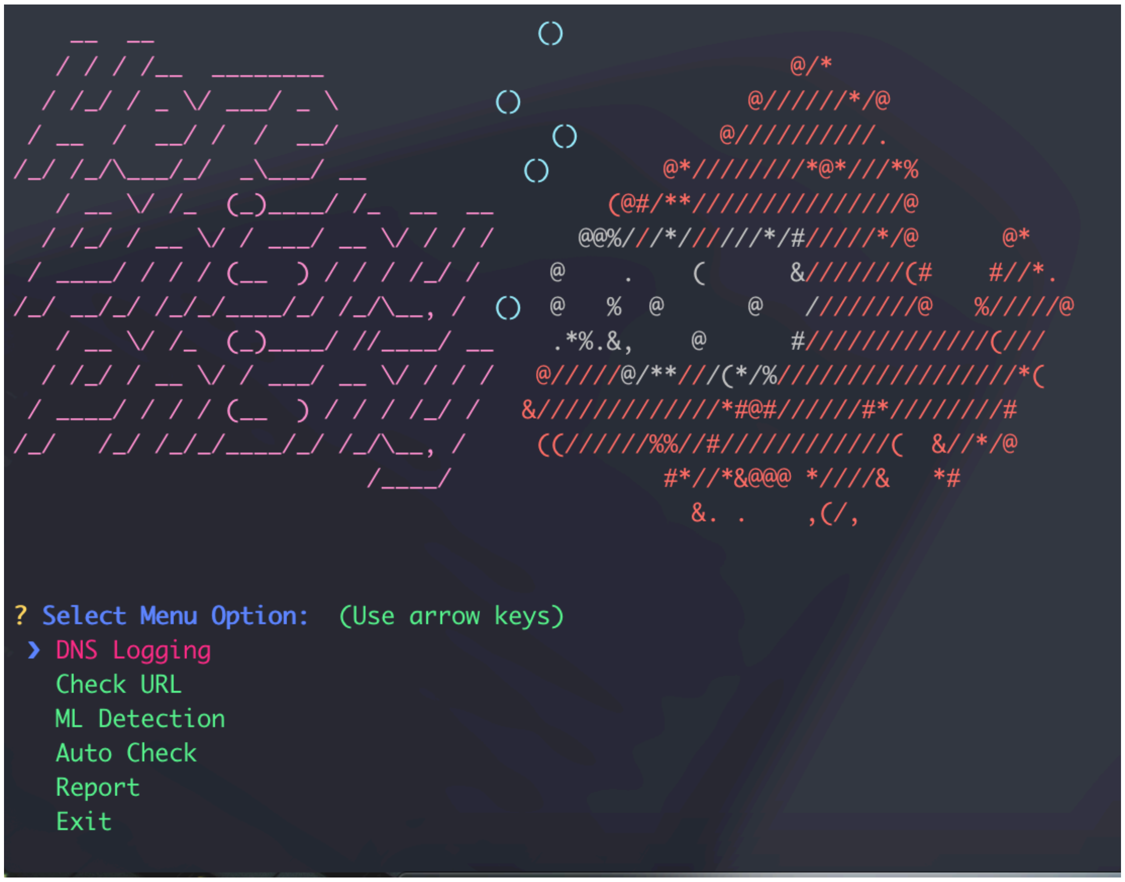
```
# cd HerePhishyPhishy
```

```
# pip install -r requirements.txt
```

Then you will be able to run the application. Please note that sudo privileges are needed in order to log the DNS traffic.

```
# sudo python3 HerePhishyPhishy
```

You will see the main menu:



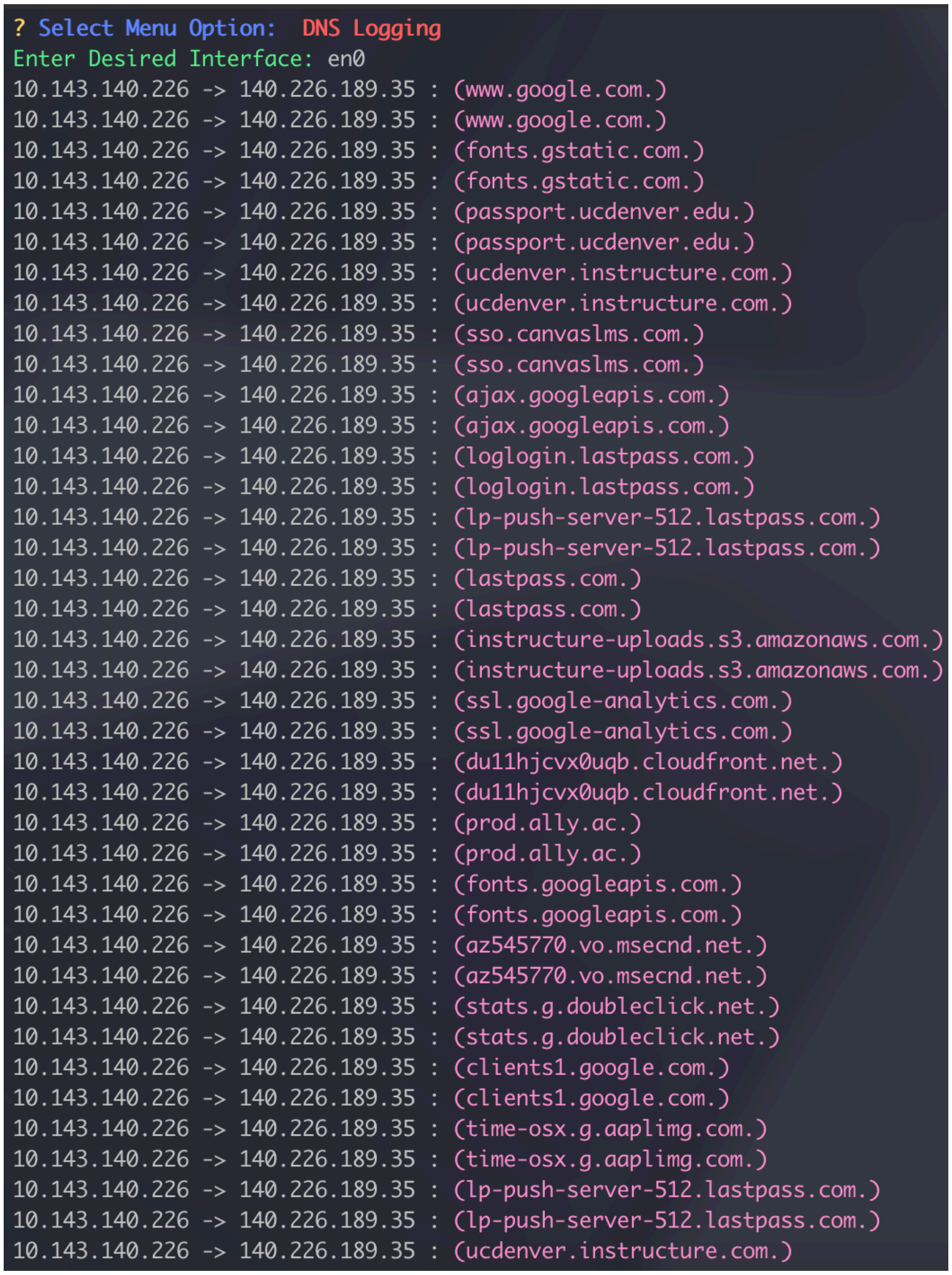
```
? Select Menu Option:  (Use arrow keys)
 ❯ DNS Logging
   Check URL
   ML Detection
   Auto Check
   Report
   Exit
```

Using the arrow keys the user is able to select a menu option and then press enter to proceed.

DNS Logging Option:

Selecting this option allows the user to see the traffic on port 53, any DNS queries made will be listed. This can help the user see if there exists any suspicious traffic. If a URL looks suspicious to the user they can then check that URL to see if it is a phishing attack. The user must know what network interface they are using.

Example of DNS Logging:

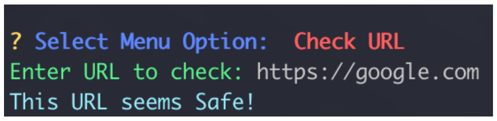


```
? Select Menu Option:  DNS Logging
Enter Desired Interface: en0
10.143.140.226 -> 140.226.189.35 : (www.google.com.)
10.143.140.226 -> 140.226.189.35 : (www.google.com.)
10.143.140.226 -> 140.226.189.35 : (fonts.gstatic.com.)
10.143.140.226 -> 140.226.189.35 : (fonts.gstatic.com.)
10.143.140.226 -> 140.226.189.35 : (passport.ucdenver.edu.)
10.143.140.226 -> 140.226.189.35 : (passport.ucdenver.edu.)
10.143.140.226 -> 140.226.189.35 : (ucdenver.instructure.com.)
10.143.140.226 -> 140.226.189.35 : (ucdenver.instructure.com.)
10.143.140.226 -> 140.226.189.35 : (sso.canvaslms.com.)
10.143.140.226 -> 140.226.189.35 : (sso.canvaslms.com.)
10.143.140.226 -> 140.226.189.35 : (ajax.googleapis.com.)
10.143.140.226 -> 140.226.189.35 : (ajax.googleapis.com.)
10.143.140.226 -> 140.226.189.35 : (loglogin.lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (loglogin.lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (lp-push-server-512.lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (lp-push-server-512.lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (instructure-uploads.s3.amazonaws.com.)
10.143.140.226 -> 140.226.189.35 : (instructure-uploads.s3.amazonaws.com.)
10.143.140.226 -> 140.226.189.35 : (ssl.google-analytics.com.)
10.143.140.226 -> 140.226.189.35 : (ssl.google-analytics.com.)
10.143.140.226 -> 140.226.189.35 : (du11hjcvx0uqb.cloudfront.net.)
10.143.140.226 -> 140.226.189.35 : (du11hjcvx0uqb.cloudfront.net.)
10.143.140.226 -> 140.226.189.35 : (prod.ally.ac.)
10.143.140.226 -> 140.226.189.35 : (prod.ally.ac.)
10.143.140.226 -> 140.226.189.35 : (fonts.googleapis.com.)
10.143.140.226 -> 140.226.189.35 : (fonts.googleapis.com.)
10.143.140.226 -> 140.226.189.35 : (az545770.vo.msecnd.net.)
10.143.140.226 -> 140.226.189.35 : (az545770.vo.msecnd.net.)
10.143.140.226 -> 140.226.189.35 : (stats.g.doubleclick.net.)
10.143.140.226 -> 140.226.189.35 : (stats.g.doubleclick.net.)
10.143.140.226 -> 140.226.189.35 : (clients1.google.com.)
10.143.140.226 -> 140.226.189.35 : (clients1.google.com.)
10.143.140.226 -> 140.226.189.35 : (time-osx.g.aaplimg.com.)
10.143.140.226 -> 140.226.189.35 : (time-osx.g.aaplimg.com.)
10.143.140.226 -> 140.226.189.35 : (lp-push-server-512.lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (lp-push-server-512.lastpass.com.)
10.143.140.226 -> 140.226.189.35 : (ucdenver.instructure.com.)
```

Check URL Option:

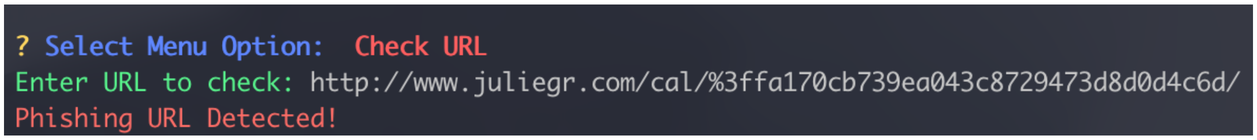
Selecting this option allows the user to check a URL of their choosing. The user will be asked to type out the URL they would like to check. The application sends an API request to the PhishTank website which checks to see if the URL already exists on their database. If the API request returns false it means the URL is not a phishing scheme and the user is notified. If it is true then the user is alerted. The user should enter the full URL.

Checking https://Google.com:



```
? Select Menu Option:  Check URL
Enter URL to check: https://google.com
This URL seems Safe!
```

Checking a known Phishing URL:

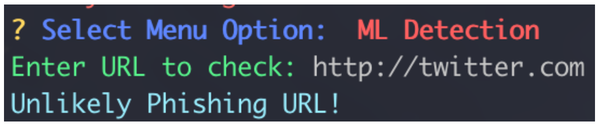


```
? Select Menu Option:  Check URL
Enter URL to check: http://www.juliegr.com/cal/%3ffa170cb739ea043c8729473d8d0d4c6d/
Phishing URL Detected!
```

ML Detection Option:

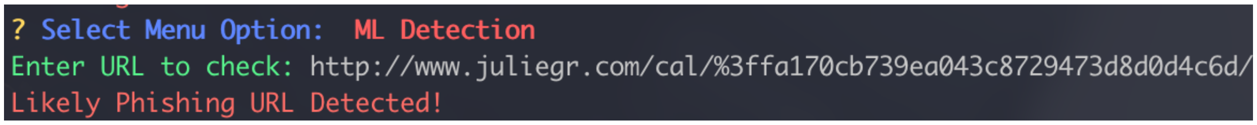
Selecting this option allows the user to check a URL of their choosing using machine learning prediction. The user will be asked to type out the URL they would like to check. The application then uses the pre-existing model, that was trained on dataset of existing phishing URLs, to then return its prediction on whether the URL could possibly be a Phishing URL or whether it is safe. The user should enter the full URL for better results.

Checking http://twitter.com:



```
? Select Menu Option:  ML Detection
Enter URL to check: http://twitter.com
Unlikely Phishing URL!
```

Checking a known Phishing URL:



```
? Select Menu Option:  ML Detection
Enter URL to check: http://www.juliegr.com/cal/%3ffa170cb739ea043c8729473d8d0d4c6d/
Likely Phishing URL Detected!
```

Immediate Future Work

Future work that can soon be implemented would be the auto-check feature. Our goal with this project was to be able to allow the user to run this application in the background and have our PhishTank API in addition to the machine learning prediction running alongside the DNS logging. This would mean that each time a DNS query is logged it would be checked against our methods and any time something seems suspicious the user could be notified. Another immediate implementation that could be done would be logging other incoming or outgoing connections from other ports. Finally, we could add more important features to our decision tree which could help improve the accuracy of the detector.

Long Term Future Work

Future work that can be implemented in the long run would be improving the already implemented features, using different machine learning methods that may improve the accuracy and precision. We would also need to use additional phishing mitigation techniques, this could possibly include checking website contents for spam, possibly using Convolutional Neural Networks to check the appearance of actual legitimate websites and those that are fake phishing sites. Looking into detecting phishing email, SMS, or even VoIP. Another implementation that would provide value would be mitigating injection attacks. Ideally we would want our application thorough enough to detect multiple different type of phishing attacks and provide an all in one platform that users could use.