Phishing Lures and Email Alerts Coding Challenge

It is strongly encouraged that you use Python for this challenge, but you can use another language if you prefer. If you have any questions while working, send an email to kristy@recordedfuture.com.

Grading Criteria

The final submission for this challenge should be your best attempt after one or two passes. We don't expect this to be a completed and perfected project, so don't spend too much time trying to get everything just right.

With this in mind, we would also like you to provide comments for parts of your solution that you believe would need improvement, briefly explaining why the improvement would be important. This will help us know what to focus on during the follow-up interview session.

We are looking to get a better idea of how you approach technical problems and analyze data. We will also grade how easy it is to understand, use, and expand on your code. Your solution to the technical pieces of this challenge will be taken into consideration but won't be our main focus.

This challenge is a substitute for the technical whiteboard portion of an interview, but be prepared to explain your solution to an interviewer and work with them to expand upon what you submit.

Background

Phishing is the fraudulent attempt to obtain sensitive information or data, such as usernames, passwords and credit card details, by disguising oneself as a trustworthy entity in an electronic communication.

One of the techniques attackers use in a phish is to register domain names that at a quick glance look legitimate but are actually malicious. For example, **paypal-login.appspot.com** looks safe at first glance, but it isn't an official PayPal website and could be operated by an attacker trying to steal PayPal login credentials. We call these types of domains Phishing Lures.

The Challenge

MartyPay is a software company that specializes in mobile payment processing. Their threat intelligence team is always on the lookout for emerging threats to the company and to the industry as a whole. To protect themselves against phishing campaigns, they're relying on your work at Recorded Future to alert them to lures they should investigate.

Your task is to implement a proof of concept that can identify potential lures in a feed of newly registered domains, disseminate alerts to the relevant members of the threat intelligence team, and come up with ways to take this proof of concept to production.

We've divided this challenge into 3 pieces to help get you started, but we would like the final deliverable to be a cohesive proof of concept rather than 3 separate solutions.

Step One

One way to identify potential lures is to curate a list of likely target terms (amazon, apple, login, paypal, facebook etc.) and check newly registered domains to see if they contain combinations of these terms before sending them for further analysis.

For this part of the challenge, we'd like you to use the following list of terms:

- cisco
- gmail
- login
- mail
- paying
- paypal
- .gov

to identify potential Phishing Lures in this list of domains: ***see Appendix One for the list*.** To qualify as a potential lure, a domain must contain at least **two** of the above terms.

Example:
Given this list of domains and using the above list of terms:

- paypal-login.appspot.com
- ciscomail.com
- cisco.heroku.com
- cisco.com

Only the following fit the two term criteria:

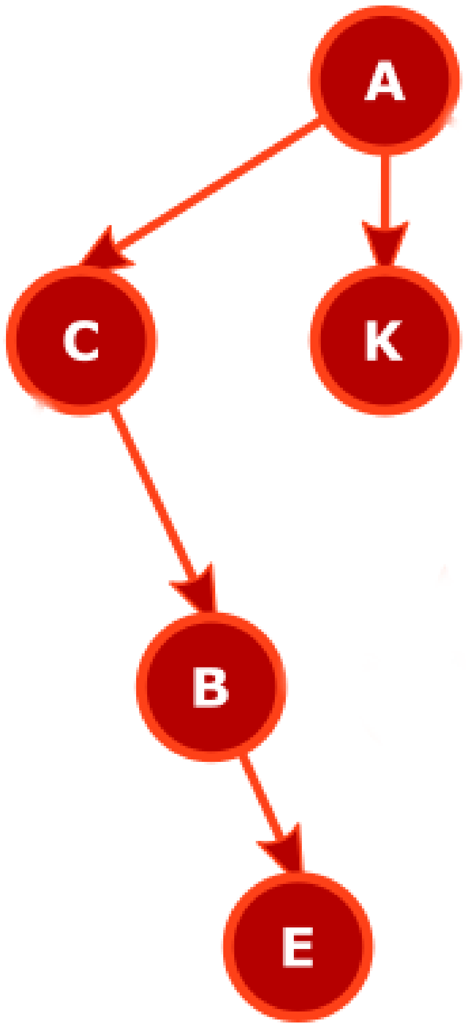
- paypal-login.appspot.com
- ciscomail.com

To help us grade, please implement the **identify_lures** method in the Python template (see **Appendix Four**).

Step Two

A security analyst at MartyPay wants to be on the lookout for Phishing Lures targeting their competitor PayPal, so they set up alerts in the Recorded Future platform to be notified whenever a potential lure is found.

The security analyst's team lead has also set up alerts for a list of terms. If a lure is found containing one of the team lead's terms, the team lead and all of their team members should be notified.



use the node's letters to represent user_ids ("A" is the director of the whole team and "E" is the lowest team member)

For example, if **“E”** sets up an alert for the term **paypal** and **“C”** sets up an alert for the term **mail**, what should happen when the lures **paypal-login.appspot.com** and **ciscomail.com** are found?
> **paypal-login.appspot.com** should send an alert to **“E”**
> **ciscomail.com** should send an alert to **“C”, “B”, “E”**

See **Appendix Two** for a file you can use to generate the team graph.
See **Appendix Three** for a file with a list of alert subscriptions.

For this challenge, we have created a simple graph to make it easier to work with. But when designing your solution, make sure to consider how your code would handle larger and more complex graphs.

To help us grade, please implement the **notify** method in the Python template (see **Appendix Four**).

Step Three

To help give us get a better understanding of how you approach problems and to give us some ideas to expand on during your interview, please answer **any two** of the following questions in a few sentences.

They're intentionally open-ended and there isn't a specific answer that we're looking for; we are interested in hearing your opinion and seeing your thought process.

1) When looking at the list of potential lures from Step One, do you notice any patterns that will likely lead to false positives?
2) What are some of the pros/cons of using a fixed list of terms to identify these Phishing Lures?
3) If you were an analyst, what additional information would you need to help prioritize which lures you should investigate first?
4) What would you need to refactor if asked to scale your solution to handle millions of domains, thousands of terms, and hundreds of users?
5) How would you architect deploying this lure alert service? What databases, class structures, tools (queues, API's etc.) would you think about using? Would it be a monolithic app or smaller microservices?

Appendix

One.

Description: A txt file with a candidate domain on each line
Download: see **coop-2020-code-challenge-domains.txt** which should be attached to an email along with the challenge

Two.

Description: A jsonlines file with an edge to the team organization chart on each line
Example: {id: K, reports_to: A}
Download: see **coop-2020-code-challenge-graph.jsonlines** which should be attached to an email along with the challenge

Three.

Description: A jsonlines file with a notification subscription on each line
Example: {id: E, term: paypal}
Download: see **coop-2020-code-challenge-subscriptions.jsonlines** which should be attached to an email along with the challenge

Four.

Description: A Python file to help you get started with your implementation. Feel free to modify and add to as needed depending on how you choose to design your solution.
Download: see **lastname_firstInitial_challenge.py** which should be attached to an email along with the challenge