Phishing Detection Using Machine Learning

Aravind Anand

Foundational Work

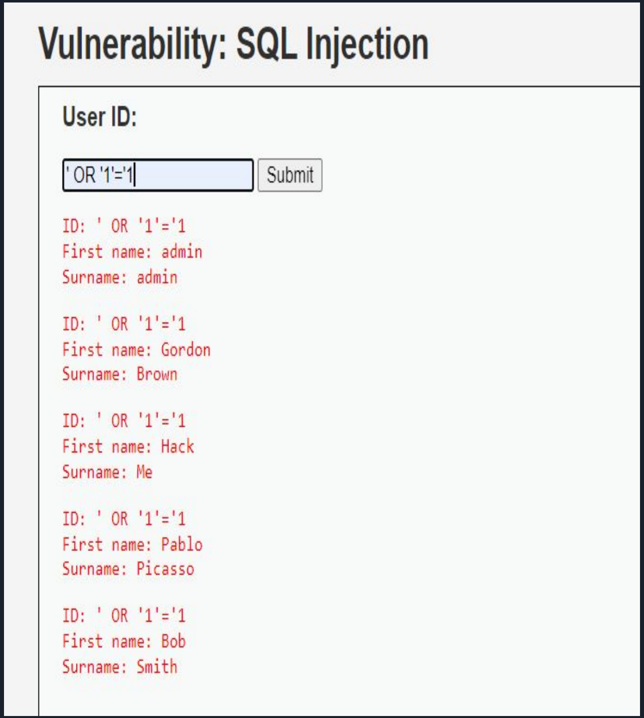
In order to detect network vulnerabilities , we learnt the basics of Machine Learning and simulated common network concerns.

Within the domain of Machine Learning, we learnt

- the basic algorithms of ML
- how to evaluate the performance of an algorithm.

Within the domain of network vulnerabilities, we simulated common network concerns using Metaspolitable. Some are listed below:

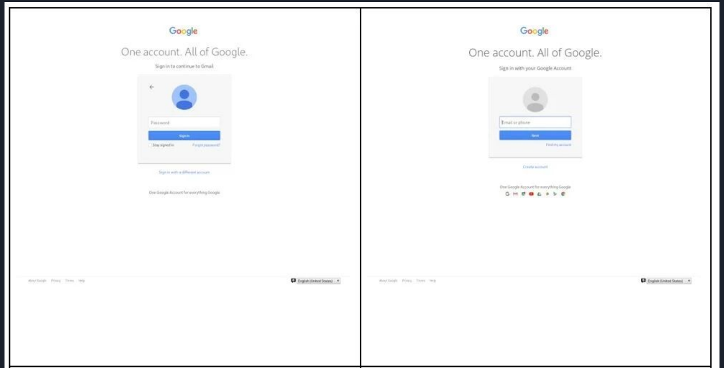
SQL injection



Command Injection vulnerability

Motivation

Phishing detection is vital in cybersecurity for several reasons. It safeguards sensitive data, preventing its compromise through deceitful methods. This is crucial for personal and organizational data security.It plays a pivotal role in preventing identity theft, safeguarding individuals from fraudulent use of their information. In summary, phishing detection is essential for protecting data, finances, identity, and reputation in our interconnected digital world.



Phishing site | Older version of the Google login page

Objective

This project's core objective is the creation of a powerful machine learning model to detect phishing URLs, addressing a critical cybersecurity challenge. The plan involves data gathering and preprocessing, algorithm selection, feature enhancement, and rigorous model evaluation. Once completed, the model will be seamlessly integrated into a user-friendly application to proactively identify and block phishing links. Continuous updates and improvements will ensure its effectiveness in countering evolving phishing techniques, ultimately contributing to a safer digital landscape and enhanced user security.

Research Paper

Title: Phishing Detection: A Literature Survey

Authors: Mahmoud Khonji, Youssef Iraqi and Andrew Jones

Published in: IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 15, NO. 4, FOURTH QUARTER 2013

Summary:

- Blacklists(MX toolbox): database of identified phishing URls
- Rule-based heuristics: pre-defined rules to judge the legitimacy of an URL
- Visual similarity comparisons: comparing the front-ends of sites to judge their legitimacy.

Research Paper

Title: Phishing Website Detection using Machine Learning Algorithms

Authors: Rishikesh Mahajan , Irfan Siddavatam

Published in: International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 23, October 2011

Summary: The phishing URL detection relies on a diverse set of features, including IP addresses, @ symbols, excessive dots, dashes in domains, URL redirections, deceptive HTTPS tokens, email-related functions, URL shorteners, lengthy hostnames, sensitive words, slash abundance, Unicode characters, SSL certificate age, anchor tag hyperlinks.These features serve as vital parameters for spotting potential phishing URLs and bolstering online security. After categorizing URLs by these traits, our Machine Learning model is trained to discern between legitimate and potentially malicious URLs.

Techniques and Algorithms Used

We utilize a K nearest neighbours from scikit-learn to detect potential phishing URLs based on various URL characteristics. We employ feature extraction techniques, including character counting, length-based features, pattern matching, and suspicious word detection, to create informative features from URLs. These features are used to train the classifier, which categorizes URLs into classes such as phishing or safe. Data preprocessing, including label encoding and data splitting, is performed to prepare the dataset for model training and evaluation. We evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.

Datasets

1. The dataset used consists of 549,346 instances out of which 392,924 are safe and 156,422 are malicious.

   URL: [2] https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls

Feature Engineering

To find length, check for anchor, https and count of ('.', '/', '//', '-'):

```
df['url_length'] = df['URL'].apply(len)
df['num_dots'] = df['URL'].apply(lambda x: x.count('.'))
df['num_slash'] = df['URL'].apply(lambda x: x.count('/'))
df['num_redir'] = df['URL'].apply(lambda x: x.count('//'))
df['num_dash'] = df['URL'].apply(lambda x: x.count('-'))
df['contains_anchor'] = df['URL'].str.contains('#')
df['has_https'] = df['URL'].str.contains("https")
```

```python
df['www']=df['URL'].apply(lambda x: x.count('www'))

def counter(x: str)-> int:
    count=0
    for i in x:
        if i.isdigit():
            count+=1
    return count

df['digit']=df['URL'].apply(lambda x: counter(x))
df['num_Qmarks']=df['URL'].apply(lambda x: x.count('?'))
df['num_undscr']=df['URL'].apply(lambda x: x.count('_'))
df['num_and']=df['URL'].apply(lambda x: x.count('&'))
df['num_per']=df['URL'].apply(lambda x: x.count('%'))
df['num_com']=df['URL'].apply(lambda x:
x.count('.com')+x.count('.org')+x.count('.in')+x.count('.html')+x.count('.php'))
```

Label	url_length	num_dots	num_slash	num_redir	num_dash	contains_anchor	has_https	www	digit	num_Qmarks	num_undscr
0	225	6	10	0	4	False	False	0	58	1	4
0	81	5	4	0	2	False	False	1	1	0	1
0	177	7	11	0	1	False	False	0	47	0	0
0	60	6	2	0	0	False	False	1	0	0	0
0	116	1	10	1	1	False	False	0	21	1	0

The DataFrame created will have the following columns:

url: This column contains the URLs that are being analyzed for phishing detection.
Label: This column represents the type of the URL, bad or good.
Url_length: Length of the url.
Num_dots: number of dots in the url
Num_slash: number of slashes
Num_redir: number of redirections identified by double slashes
Num_dash: number of hyphens
Contains_anchors: 0 or 1 based on existence of pound sign ‘#’ in url
Has_https: 0 or 1 based on existence of https request
Www: number of ‘www’ in the url
Digit: number of digits in the url
num_Qmarks: number of question marks
Num_undscr: number of underscores ‘_’
Num_and: number of aprisond ‘&’
Num_per: number of percentage signs ‘%’
Num_com: number of .com,.org,.in ,.php,.html in the url

K Nearest Neighbours

```
input=df.drop(['URL', 'Label'], axis='columns')
target = df.Label
X_train, X_test, y_train, y_test = train_test_split(input, target, test_size=0.15)
knn = KNeighborsClassifier(n_neighbors=9)
```

With 85:15 split

	precision	recall	f1-score	support
0	0.80	0.67	0.73	23413
1	0.88	0.93	0.90	58989
accuracy			0.86	82402
macro avg	0.84	0.80	0.82	82402
weighted avg	0.86	0.86	0.86	82402

Decision Tree

With 80:20 split

Accuracy Score: 0.9576317385729313

	precision	recall	f1-score	support
benign	0.97	0.98	0.97	85621
defacement	0.98	0.99	0.98	19292
phishing	0.95	0.94	0.95	6504
malware	0.87	0.84	0.86	18822
accuracy			0.96	130239
macro avg	0.94	0.94	0.94	130239
weighted avg	0.96	0.96	0.96	130239

Results

With 80:20 split

Decision Tree: 85%

With 60:40 split

Random Forest: 79%

With 85:15 split

K Nearest neighbours: 86%

References

Technical Report:

Mahajan, Rishikesh & Siddavatam, Irfan. (2018). Phishing Website Detection using Machine Learning Algorithms. International Journal of Computer Applications. 181. 45-47. 10.5120/ijca2018918026.

Dataset:

https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls

https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset

GitHub Repository:

https://github.com/arvindashok/Phishing_Detection

https://github.com/Surag21/Phishing_detection