Detecting Malicious Links at a Glance

Andrew D. Barlow
University of Maryland,
Baltimore County
drew.barlow@umbc.edu

Sadi M Jawad Ahsan
University of Maryland,
Baltimore County
wh11893@umbc.edu

Paul Ledala
University of Maryland,
Baltimore County
pledala1@umbc.edu

Sivanag Maddineni
University of Maryland,
Baltimore County
smaddin1@umbc.edu

December 15, 2022

ABSTRACT

Computers being able to communicate with each other over a network has introduced a large attack vector. Users are free to visit any website on the internet that they would like to. Likewise, users are free to host said websites. The problem with these freedoms is the fact that those with malicious intentions exist, and they will attempt to fulfill them. A common method of doing so is via phishing attacks, which may involve coercing a user into clicking a URL that redirects to a malicious website. We propose a two-phase approach to determining if a URL is malicious. The first phase analyzes the characteristics of the URL, as well as some other defining features such as top-level domain and geographic location of the host. If phase one is uncertain about its classification of a URL, then it will pass it off to phase two. In phase two, the HTML and JavaScript content of the page is analyzed. The first phase does not require visiting the webpage at all. The second phase does require making a request to the webpage, though there are various tools that allow for one to retrieve the HTML and JavaScript content of a page without revealing vulnerable information to the host.

1. INTRODUCTION

Ever since the conception of computers, the idea of being able to use them maliciously has existed. When computers first became able to communicate with each other in the late 1960s, an entirely new attack vector was introduced. That is, one was able to attack a computer remotely instead of physically. However, these networks were typically on a very small scale. It is no secret that with the advent of the internet in 1989 came the ability to attack any computer connected to it from any location.

When a client makes a connection to a website, the website's host obtains a lot of data about it. Aside from every bit of information obtained through the HTTP request header, the host also obtains the client's IP address. Having access to an IP address allows one to not only scan the client machine for vulnerabilities, but also get an approximate location of where that client is geographically located.

On top of having access to a client's IP address, the malicious actor can also design the website itself to be malicious. One example of such design would be masquerading as a different website entirely–

one that users already trust. Through a simple login page that looks close enough to the original, a malicious actor can harvest login credentials from users. Another example of malicious behavior is making the claim that a piece of downloadable software is safe and legitimate, when in reality it may do any number of things to a client's computer when executed.

A number of services exist just to mitigate this attack vector. An example of such is Google keeping a list of websites that could leave a user susceptible to phishing or malware attacks. When a user goes to visit one of these websites, Google Chrome may outright prevent them from doing so. Similarly, other services such as Malwarebytes Premium exist that perform the exact same task.

The problem with the aforementioned services is that they must have already encountered these malicious websites at least once before. This implies that there has already been one victim of the website, which is unacceptable.

2. MOTIVATION

It is no secret that the internet is widely used by all demographics in this day and age, ranging from children that can barely read to very elderly seniors. While not every demographic is safe from visiting malicious webpages, it is typically these two demographics that are the most susceptible to scams. The Federal Bureau of Investigation (FBI) estimates that seniors lose more than $3 billion annually to varying types of scams [1].

Phishing attacks are very commonly used in cyberspace today. Roughly 1% of all emails sent contain some kind of phishing attack– 30% of these emails are opened [2]. In 2021 alone, there were 214,345 unique phishing websites discovered and reported [2]. 90% of all data breaches happen because of somebody falling victim to a phishing attack [2].

We wish to provide users with confidence that the website they are about to visit is not malicious– preventing them from becoming just another victim. It is our hope that the approach we propose in this paper can be implemented into a browser extension that runs a check on any hyperlink a user clicks.

3. RELATED WORK

Significant research has been carried out on Malicious Website detection using machine learning techniques. However, these papers have restricted themselves to relatively few features. Ma et al. in their paper on detecting malicious websites from suspicious URLs have considered only URL related attributes [4]. While, Wressnegger et al. have discussed Flash based malwares [5], Mavrommatis et al. have discussed iFrames [6], Ganesh et al. have analyzed Java Applet based malwares [7], Mao et al. have discussed HTML content based analysis of malicious websites [8], and Cova et al. have analyzed few JavaScript based attributes [9]. Thus, we find that a holistic analysis of all possible attributes that are required for Malicious Website detection is lacking.

4. DATASET

The dataset for this research was prepared by crawling and scraping websites on the internet using MalCrawler. The labels have been verified using the Google Safe Browsing API. The basic information captured during the crawl included URL, IP address and web content. The features were then extracted through further preprocessing. The choice of features used in this research was based on their relevance in malicious web page classification. While more features could have been selected, the ones picked up in this paper were ranked best predictors in earlier research [5].

The dataset consists of over 1.2 million training samples, and over 200,000 test samples. The ratio of benign (good) to malicious (bad) samples is given in the following figure:

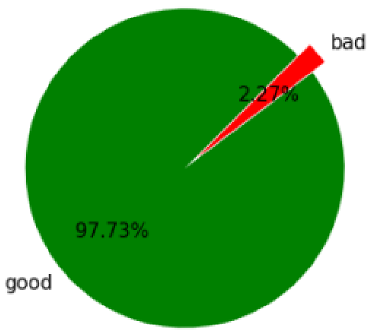

Figure 1: Pie chart for malicious and benign samples.

5. PHASE 1

The goal of the first phase is to completely avoid visiting the potentially malicious website entirely. If the resulting confidence level of the classification is within some chosen threshold, then we cycle the sample over to the second phase.

5.1 Chosen Features

Because we would like to avoid giving the potentially malicious website any information at all, we strictly make use of resources that it must provide us itself. That is attributes resulting from the URL, including:

- Length of URL,
- Length of netloc (network location),
- Length of path,
- Length of queries,
- Number of parameters,
- Length of fragment

We have also chosen to make use of the top-level domain, geographic location, whether or not a who-is lookup can be performed on the domain, and whether the connection uses HTTP or HTTPS. Using Seaborn, the correlation matrix for these given features is given in Figure 2:

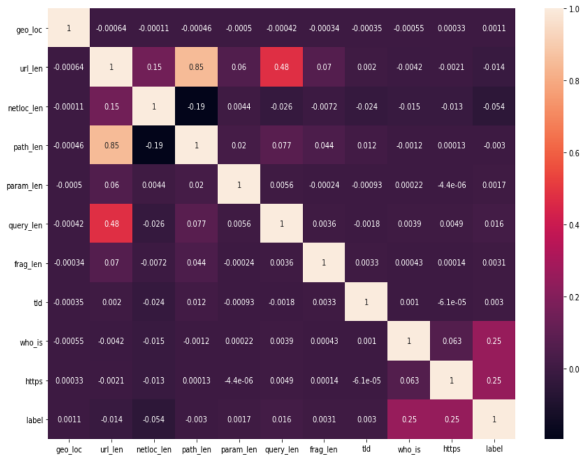

Figure 2: Correlation matrix for phase one features.

5.2 Experiments

For classification of the dataset, we used Random Forest classifier with Grid Search with hyper parameter tuning as baselines. The classifier was 98% accurate. Its hyperparameters are given below:

- Max. Features: 4,
- N Estimators: 130

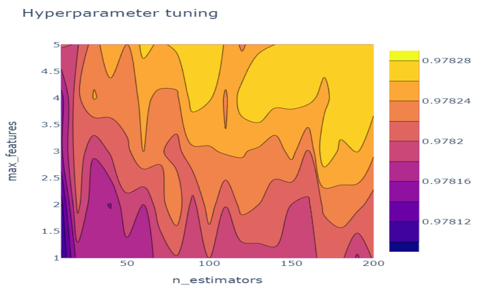

Figure 3: Hyperparameter tuning, grid search.

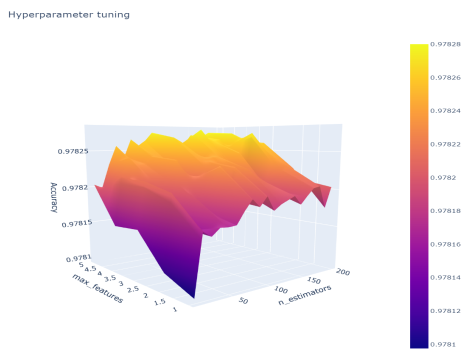

Figure 4: Hyperparameter tuning, grid search (3d).

Evaluation of the samples (1.2 million) is given below in Table 1:

Metric	Result
Precision	98%
Recall	100%
F1	99%
Accuracy	98%

Table 1: Evaluation on the dataset.

A high accuracy of 98% was achieved on the test dataset, thereby vindicating the effectiveness of this model in detecting malicious web pages. Precision, which denotes percentage of predicted positives that were correctly classified, is at 98%. Recall, which gives the percentage of actual positives that were correctly classified is also convincing at 100%. The Confusion Matrix that gives us the distribution of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN), is also given on the next page in Figure 5:

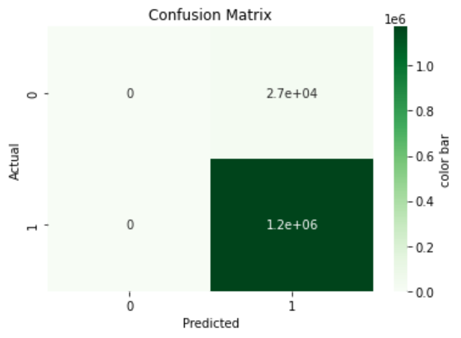

Figure 5: Confusion matrix of dataset.

Because of the tabular nature of the chosen feature set, a standard neural network is sufficient for this task. The neural network consists of three linear layers and two Leaky ReLU activation layers. It was implemented in PyTorch, and was trained on the University of Maryland, Baltimore County's (UMBC) Discovery, Research, and Experimental Analysis of Malware (DREAM) Lab's "Fantasy" server using a single NVIDIA Titan RTX GPU. It made use of the following hyperparameters:

- Epochs: 25
- Hidden Dimension: 237
- Loss Function: Cross Entropy Loss
- Optimizer: SGD
    - Learning Rate: 0.0001
    - Momentum: 0.99

The output of the neural network is a vector of confidence values representing confidence in whether the URL is benign or malicious. To determine what samples need to be passed onto the second phase, we must define a minimum and a maximum acceptable confidence value. In our case, we determined this interval to be ±2.5% of the mean through trial and error.

5.3 Results

The resulting output of the neural network is shown in the following classification report on the next page (Table 2).

Metric	Benign	Malicious
Precision	99%	31%
Recall	96%	74%
F1	98%	44%
Accuracy	96%	

Table 2: Phase one classification report.

It is obvious that the phase one neural network has relatively little confidence in its classification of malicious URLs. However, it achieves high performance when it comes to benign web pages. This shows that having a confidence interval to pass samples to phase two is indeed the correct choice.

Via the aforementioned interval defined in section 5.3, the neural network passed a smaller dataset with the following statistics to phase two in Table 3:

Class	Number of Samples
Benign	11,851
Malicious	5,503
Total	17,354

Table 3: Samples sent to second phase for evaluation.

6. PHASE 2

The goal of the second phase is to get samples from phase one which are susceptible to misclassification and classify them as malicious or benign based on the JavaScript content. This phase requires natural language processing of the content.

6.1 Chosen Features

The chosen feature for phase two is the JavaScript content of the website. We obtain the key words from the content to form a vector. Word embeddings are used to convert the text into feature vectors.

6.2 Experiments

A benchmark model is implemented using Word2Vec on the data, then using Random Forest on top of that to achieve a precision of 99%.

Word2vec attempts to learn relations between words, then embeds them in a lower-dimensional vector space using a neural network with only a few layers. In order to accomplish this, Word2Vec trains words against nearby words in the input corpus, thereby capturing some of the meaning of the word sequence. The parameters in the model such as window size, vector size, and vector count are adjusted by trial and error to tune for maximum accuracy.

The hyperparameters used for our model are:

- Window Size: 5,
- Vector Size: 100,
- Min. Count: 2

We used the Bidirectional Encoder Representations from Transformers (BERT) introduced by Google AI [3] to retain the contextual meaning of the sentences in the JavaScript content and get the embedding to feed as input to the ML model. We use BERT as a method of extracting pre-trained language representation that can further be used as input to create models for classification. We use a deep learning sequential network as our model with three Linear layers and two Tanh activation layers.

The hyperparameters chosen for the BERT model are:

- Encoder Model: bert-base-uncased,
- Encoder Learning Rate: 1e-07,
- Learning Rate: 3e-07,
- Batch Size: 32,
- Epochs: 5-10,
- Loss Function: Cross Entropy Loss,
- Optimizer: Adam

6.3 Results

The phase two model takes in the JavaScript content of samples passed through phase one.

The classification for the benchmark, Word2Vec, is given in Table 4:

Metric	**Result**
Precision	99.8%
Recall	100%
Accuracy	99.8%

Table 4: Phase two benchmark classification report.

The full classification report for BERT is also given here in Table 5:

Class	**Precision**	**Recall**	**F1**
Malicious	26%	97%	41%
Benign	100%	94%	97%
Macro	63%	96%	69%
Weighted	98%	94%	96%
Accuracy	94%		

Table 5: Phase two BERT classification report.

We focused on getting better recall scores in the model for our use-case. These final results give us a strong level of confidence in our predictions, on real world websites.

7. FUTURE WORK

In this experiment we applied the Random Forest classifier for phase one and Word2Vec with Random Forest for phase two, which gave us good accuracies for baseline models. We intend to try comparisons with other ML models. For interpretability, Local Interpretable Model-agnostic Explanations (LIME) will be used in future work. In our work, the threshold for sending data samples to phase two is decided by trial and error method and determined to be at ±2.5% of the mean. In future work, a constructive method like Support Vector Machines (SVM) could be used to determine the threshold used in phase one.

8. CONCLUSION

We used machine learning techniques currently used in research to come up with an efficient way to detect malicious links. Although we faced the problem of skewed data, we corrected it by taking equal proportions of benign and malicious links in the training data. Most of the links are correctly classified in phase one, which allows the user to skip opening a malicious link. To improve the accuracy of the model, we've also incorporated a second phase which involves parsing the web content and running an NLP classification task. This method proves to be more robust and improves the model's overall F1-score.

9. REFERENCES

[1] FBI. (2020, June 15). *Elder fraud*. FBI. Retrieved December 15, 2022, from https://www.fbi.gov/how-we-can-help-you/safety-resources/scams-and-safety/common-scams-and-crimes/elder-fraud

[2] Slandau. (2022, April 5). *Phishing attack statistics 2022*. CyberTalk. Retrieved December 15, 2022, from https://www.cybertalk.org/2022/03/30/top-15-phishing-attack-statistics-and-they-might-scare-you/
[3] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). *Bert: Pre-training of deep bidirectional Transformers for language understanding*. arXiv.org. Retrieved December 15, 2022, from https://arxiv.org/abs/1810.04805
[4] J. Ma, L. K. Saul, S. Savage, and G. M. VoRegional Conference on - ACM SE '14, Detect Malicious Web Sites from Suspicious URLs," World Wide Web Internet And Web Information Systems, 2009.
[5] C. Wressnegger, F. Yamaguchi, D. Arp, and K. Rieck, "— Technical Report — Available:http://christian.wressnegger.info/content/projects/gordon/2015-tr.pdf.
[6] P. Mavrommatis and N. Provos, "All Your iFRAMEs Point to Us," Symposium
[7] N. Ganesh, F. Di Troia, V. A. Corrado,T. H. Austin, and M. Stamp, "Static Analysis of Malicious Java Applets," Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics, 2016.
[8] J. Mao, W. Tian, P. Li, T. Wei, and Z. Liang, "Phishing-Alarm: Robust and Efficient Phishing Detection via Page Component Similarity," IEEE Access, vol. 5, pp. 17 020–17 030, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8015116/.
[9] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of driveby-download attacks and malicious JavaScript code," in Proceedings of the 19th international conference on World wide web - WWW '10, 2010.
[10] A. Gorji and M. Abadi, "Detecting Obfuscated JavaScript Malware Using Sequences of Internal Function Calls," in Proceedings of the 2014 ACM Southeast 2014.elker, "Beyond Blacklists : Learning to Germany ¨ , no. December, 2015. [Online]. Analyzing and Detecting Flash-based Malware using Lightweight Multi-Path Exploration," University of Gottingen