URL Semantic Graph Analysis for Phishing Detection

Jack Heavey, Matt Walsh
CS6501: Network Security and Privacy

Abstract

In recent years, phishing attacks have becone more prevalent, using well-made fake websites to steal user information and bypass security measures by tricking individuals into believing their legitimacy. The accurate and speedy detection of phishing sites proposes to greatly reduce the number of unintended accesses to systems. While the web page of a phishing site may look legitimate, a common method of identifying these sites is to examine the URL itself. These URLs will often use variations on common words or popular domains to try and give themselves legitimacy but will not match the true domains exactly. In this paper, we propose a number of machine learning models, including a basic neural network and a graph convolutional network (GCN) to classify phishing websites using only the URL. This text analysis allows classification to occur without any additional traffic information. Our models all exceed 75% accuracy in classification, with our two neural network models approaching 80% accuracy using only created features from the text of the URL.

For this project, Jack was in charge of the initial analysis and graph construction while Matt was in charge of writing our our classification models. Jack ran the models and produced the analysis.

Introduction

Over the past decade, phishing attacks have surged in popularity, with more than 445,000 incidents of financial phishing in 2013 [2], and experts predict that the total number of phishing attempts will continue to increase by more than 50% per month as technology for reaching large audiences improves[4]. Phishing attacks focus on creating web pages that mimic authentic web pages, convincing users to submit confidential information that is then used by the attackers to compromise accounts and bypass security measures. Phishing attacks are most commonly performed through email, where attackers mimic an email sent from a reliable source with a link to a malicious site where confidential information can be entered. Because this confidential information often allows attackers to bypass other security measures, there is a large incentive by security professionals to find ways to highlight or prevent phishing attacks. One of the most successful measures is to highlight possible attacks before they happen, with many companies and institutions training people on how to recognize phishing attacks. These trainings will highlight the suspicious nature of phishing links, which differ from the proper domains that they are attempting to mimic. In this work, we present classifiers that are trained to recognize suspicious links. These classifiers function purely on the URL semantics, which allows them to classify the site before any traffic has been sent and warn individuals before they visit the site. We present the accuracies of a number of different classifier, including a naive linear classifier, a basic neural network, and a graph convolutional network (GCN). These classifiers can be applied within large institutional networks to preemptively stop phishing attacks before they occur, saving institutions time and possibly money.

Prior Work

Much of the prior work in identifying phishing web pages has focused on either the information on the website, analyzing web page content such as text and images[1], or analyzing the path that the traffic takes, identifying Autonomous Systems or domain groups that are commonly used to detect new phishing attempts before they can get started[5]. In these classification methods, the models require additional information that causes the classification of an individual website to require more information. As such, they would generally be used by total web scanners or hosting sites that rather than at the individual level, identifying malicious websites independent of the individual. Because of this, these models must be used beforehand and offer little practical use for individuals or companies in their day to day, who would see too much overhead as they travel from web page to web page to use these classifiers regularly. Instead, we make the decision to only use the URL semantics in our classification. While this requires initial overhead to train, the models that we present can be deployed with relatively low space overhead, similar to a company level firewall, and can classify web pages with minimal interruption in normal internet travel, keeping productivity while improving safety. We think that our models complement existing methods well, which may have higher accuracies due to the expensive training and classifying but are less practical in day to day use, providing IT specialists and security re-

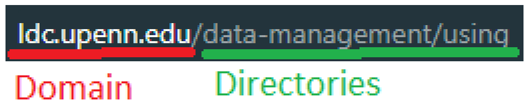

Figure 1: The domain of a URL versus the directories.

searchers another tool with which to defend against the ever changing realm of phishing attacks.

Methodology

We obtained a labeled data set of 96,005 labeled data sets from Aalto University in Espoo, Finland[6], which contained 47,904 phishing URLs and 48,101 legitimate URLs. While this data did come pre-labeled with features that they have generated, these features generally were related to syntax of the URL or other features that they had identified within their analysis rather than the semantics that we wanted to explore. In our initial review of the data, we noticed that many URLs contained words to make them seem more legitimate – words such as "payment," "login," "recover," etc. were extremely common. Seeing this, our goal was see what sorts of similarities existed between the words contained in the URLs of legitimate web pages and malicious web pages. To do this, we separated URLs between their domain, which must be registered with a host and is extremely difficult (or impossible) to hide, and their directory structure, which can be arbitrarily set to attempt to fool individuals. Figure 1 illustrates where we made this separation, taking all characters after the first "/" symbol. Throughout the rest of the paper, we refer to this as the directory structure, while we will still use URL to refer to the entire text. Within the directory structure, we created a dictionary that linked each URL with the set of words in the top 10,000 English words of length 4 or more letters. We limited our search to longer words and only the top 10,000 as defined by the Google Research Team[8] in order to remove noise caused by shorter words that do not provide semantic information, such as "a, "I," "of," etc. Once all of these sets were created, we went through each set of words found the three most similar words to it, as defined by the Gensim library for natural language processing[7]. These similar words give us a larger data set to work with while also linking together URLs that share similar, but not exact, semantic meaning with each other.

Graph Construction

After finding the sets of words and similar words to the URL, we construct a graph that links together the URL with the words that are within it, which are then linked to the first three returns from Gensim for similar words. Figure 2 shows how this would be constructed for a single URL that has two words within its directory structure. This construction allows for URLs to be linked if they have either the same word within them (directories "/payment_authorization/" and "/GooglePayment/" would share the "payment" node) at distance 2 or if the similar words

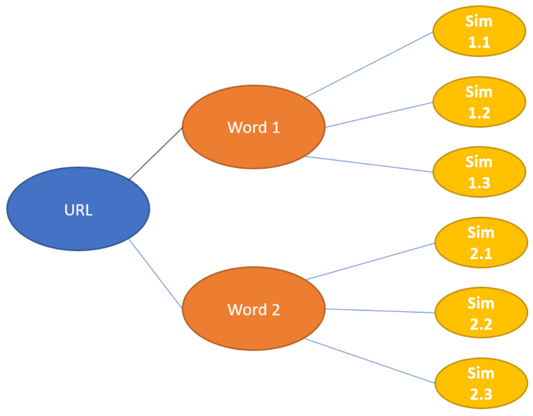

Figure 2: Structure of a singleton URL with two words in the directory structure.

to a word within a URL is contained within another URL, which would be at distance 3. Finally, a URL is distance 4 from another URL if they share no exact same words within their directory structure, but one word in each URL has the same similar word.

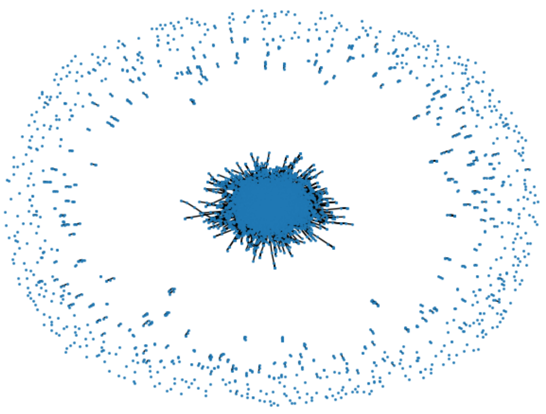

Figure 3: A visualization of the URLs being broken down by the words within their directory structure.

Creating this graph structure initially allows for analysis of our data set. Figure 3 shows a visualization of our graph.

Graph Analysis

When looking at the graph structure, we see that there is an extremely dense and connected center in the middle and a large number of disconnected nodes on the outside. The dense center shows that a large number of websites either share similar or the same words within their directory structure, while the singletons or low-connectivity graphs on the outside show that many websites either have no or very few English words within. Initial analyses of this show that phishing websites are much more likely to be connected

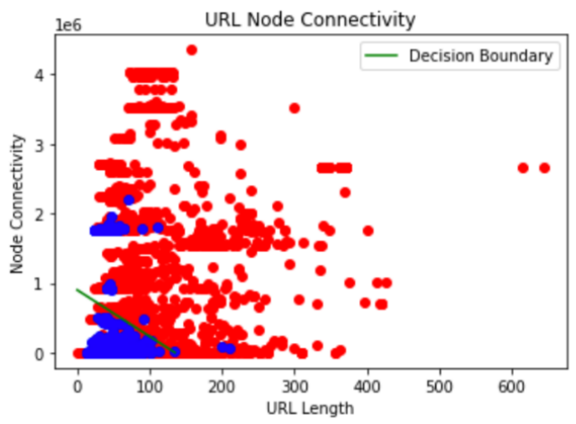

Figure 4: The linear decision boundary between the URL length and $m(n)$ for URLs n.

to other URLs and are much more connected. The average phishing website was connected to 800,000 other nodes through at least 1 path, which shows that these websites are extremely likely to share common words or similar words to other phishing sites. The high degree of connectivity also stems from the length – phishing websites are more than 2.5 times as long as legitimate websites within our data set, with an average length of 92.2 compared to 36.8 characters. Being so much longer means that these phishing websites are going to be much more dense, on average, than legitimate websites.

Model Creation

Initially, we wanted to have a baseline classification method using our graph structure that we could then compare more advanced models to. Using our semantic graph, we created a metric for each URL based on how many words it shared with other URLs. To do this, we say $m(n) =$ Number of malicious nodes at distance 2 or less from node n, which means that Node n will share at least one similar word with $m(n)$ nodes. With our independent variable being the length of the URL and our dependent variable being our metric m, we construct a linear decision boundary in 2-dimension, as shown in Figure 4.

We can see an immediate positive correlation between the connectivity to other malicious nodes and the length of the URL. This naive decision boundary boasts a 76% accuracy at predicting whether a URL is going to be malicious or benign.

Neural Network

For our second model, we train a standard neural network using the Tensorflow framework. For features, we explore all of the unique words that were either found within the directory structure of the URL as well as all of the unique similar words that are connected to these words, and we create a vector v. The feature vector for each URL consists of a (1,0)-vector where the ith position is 1 if the word corresponding to i is in the directory structure or similar words to a given URL, or 0 otherwise. In this way, we partially encode the graph structure as our feature set. We train our model using a learning rate of 0.01 over 300 epochs, although we stop training early using Tensorflow's conditions for early stopping if accuracy is not increasing, and training accuracy generally plateaued around 60 epochs and then terminated between 120 and 150 epochs. Each training epoch took approximately one second on a system with an Nvidia GTX1080 and Intel i7-7700k 4.2GHz processor, meaning that our entire training took under 3 minutes. For a standard URL, the feature vector can generally be created and processed in under 3 seconds.

Graph Convolutional Network

Due to the structure of our graph and it's non-Euclidean nature, we felt that a Graph Convolutional Network (GCN) would perform best for classifying our data. GCNs' use has increased in recent years, where they have been used to great success for node classification. Graph Convolutional Networks are similar to Convolutional Neural Networks in the sense that they use kernals or filters to have input layers learn from neighboring data. However, GCNs are used on unstructured data to learn the features of a node based on the features of neighboring nodes. GCNs are most often used to make predictions on unstructured data such our text data, and have outperformed other graph neural networks in a number of text domains such as natural language processing and semantic analysis[9]. We used the StellarGraph package, combined with Tensorflow, in order to do our training and testing[3].

Alternative Graph Construction The graph used for this model is different from the graph previously described, as feeding in our current graph with so many unlabeled word nodes would cause a decrease in performance. In this new graph, only nodes in this graph are full URLs, and each node has a feature vector representing words related to that URL. This feature vector indicates if a URL contains certain words and is constructed from a dictionary of all words found when parsing URLs and determining similar words. Edges in this graph are defined as follows: Let W_A be the set of all words found in URL A. Let S_A be the superset of W_A containing the three most similar words to each word in W_A. Nodes A and B will have an edge if $|S_A \cap S_B| > 5$, and the weight of that edge will be $|S_A \cap S_B|$. The inputs to the GCN are then the feature vectors of all nodes as well as the adjacency matrix of the graph. Epochs took between 7 and 9 minutes to train on the same system on even a fraction of the data, a much steeper training price to pay.

Results

Our linear classifier had an accuracy measure of 76% for identifying both malicious and benign URLs. Classifying a URL in this method is dominated by the time that it takes to identify the words within the subdirectories, which was approximately 3 seconds for an average individual URL based on our string search and is performed in polynomial time based on the length of the directory structure. From here, insertion into the graph is performed in $O(1)$ time, while Node

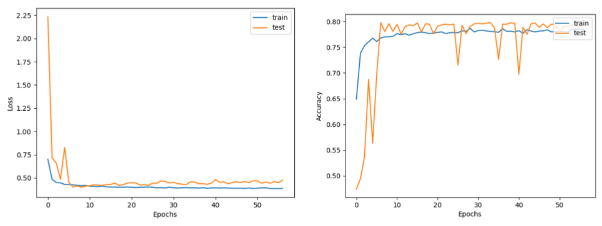

Figure 5: (Left) Loss function of the neural network through training and testing epochs, with training in orange and testing in blue. (Right) Overall model accuracy through training and testing epochs with the same colors.

Connectivity is measured in $O(n)$ time for a graph with n nodes already in it. Thus, while the linear classification is quick, it requires the memory overhead of the rest of the graph to correctly measure the connectivity of the URL to other URLs, which is not ideal for maintaining the classifier at the browser level.

Our baseline neural net had an overall accuracy of 79.36% with a loss value of approximately .389. This outperforms our naive linear classifier by approximately 4% while still offering the speed of a more simple model. Figure 5 shows the loss and accuracy graphs over training and testing.

Classification of a new single node in this neural network is nearly instantaneous given that a feature vector is provided, which means that the only overhead is how quickly this feature vector can be constructed after a link is clicked, which is solely reliant on the string search. This is actually faster than our linear classification and does not require the memory overhead of saving the lexicographic graph within the browser, meaning the neural network was actually our fastest classifier in practice. While the three seconds that we see in feature creation and classification would amount to considerable slowdown in loading times for the standard web page, we believe that future work can reduce this overhead even further to make this a viable front end option on networks.

Our Graph Convolutional Network boasts an 80.21% accuracy after training for 105 epochs, with the accuracy reaching above 79% as early as epoch 23. Classification of a single node within this network takes much longer, however – while the feature vectors are the same between the two models, the entire adjacency matrix is also required when identifying a single node, which means that there is a much higher classification overhead than the neural network. We discuss more comparisons between the two models in the Discussion and Conclusion section.

Discussion and Conclusion

While we were happy with the results of our classifiers overall, we were surprised by how little the accuracy of the Graph Convolutional Network exceeded the accuracy of a more simple neural network. While the GCN is able to take in structural relationships between different URLs, we believe that the accuracies are so similar because we fed in a majority of our graph structure into the neural network as a feature vector. The additional information provided by our secondary graph definition did not supplement the word vector very much, and additional testing of both classifiers showed that they performed worst on phishing nodes that were extremely disconnected from the rest of the graph, containing no English words. And while our classifier performed poorly on identifying these malicious sites, we believe that these are the easiest sites for humans to identify based on safety trainings. Because of this, we believe that our neural network is actually the most practical classifier to use in the real world, offering the smallest overhead for feature creation and URL classification when compared to the GCN and linear classifiers. Future work in this space revolves around the continued identification and highlighting of malicious websites. A speedup in our feature creation would allow for the cheaper deployment ofModern technology has made it easier and easier to disseminate phishing links to wider audiences, reducing the return that these attackers require to continue phishing. Only through a combination of prevention methods at all levels of the internet connection truly fully prevent phishing attacks from being successful.

References

[1] M.A. Adebowale, K.T. Lwin, E. Sánchez, and M.A. Hossain. Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text. *Expert Systems with Applications*, 115:300–313, 2019.

[2] Fadi A Aloul. The need for effective information security awareness. *Journal of advances in information technology*, 3(3):176–183, 2012.

[3] CSIRO's Data61. Stellargraph machine learning library. https://github.com/stellargraph/stellargraph, 2018.

[4] B. B. Gupta, Aakanksha Tewari, Ankit Kumar Jain, and Dharma P. Agrawal. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications*, 28(12):3629–3654, March 2016.

[5] Samaneh Mahdavifar, Nasim Maleki, Arash Habibi Lashkari, Matt Broda, and Amir H. Razavi. Classifying malicious domains using dns traffic analysis. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 60–67, 2021.

[6] Samuel Marchal, Jerome Francois, Radu State, and Thomas Engel. Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11:458–471, 12 2014.

[7] Radim Rehurek and Petr Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

[8] wordlywisdom. Google 10,000 words. https://github.com/first20hours/google-10000-english, 2013.

[9] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1), November 2019.