REAL-TIME DETECTION OF PHISHING TWEETS

Nilesh Sharma¹, Nishant Sharma², Vishakha Tiwari³, Shweta Chahar⁴, Smriti Maheshwari⁵

¹Software Engineer at Dubizzle, Dubai, UAE
nilesh.sharma7@gmail.com

²Integrated Dual Degree (B.Tech+M.tech), Electronics & Communication
Department, Indian Institute of Technology, Roorkee, India
nishantsharma.iit@gmail.com, nish5uec@iitr.ac.in

³, ⁴B.Tech, Computer Science Department,
Hindustan College of Science & Technology, India
vish.twr26@gmail.com
chaharshweta@gmail.com
stellermsr19@gmail.com

ABSTRACT

Twitter is an immensely popular social networking site and micro blogging service where people post short messages of 140 characters called tweets. Phishers have started using Twitter as a medium to spread phishing scams because of the fast spread of information.

We deployed our system for end users by providing an easy to use “Web framework” which takes the tweet id and the specific keyword, and in return it will give tweets indicating legitimate or unsafe. We have a green background indicating a legitimate or safe URL and red symbols indicating the malicious or phishing URL with the help of APIs and machine learning algorithm.

KEYWORDS

Phishing Detection, Python, Machine Learning Algorithm, Twitter, Web framework

1. INTRODUCTION

Phishing is the act of attempting to acquire information by masquerading as a trustworthy entity in an electronic communication. Twitter, due to its large audience and information reach, attracts Spammers.

There has been an increase in phishing attacks through social media due to ease spread of information on social networks. Such a rise in phishing attacks on social media presents a dire need for technological solutions to detect these attacks and protect users from phishing scams.

Natarajan Meghanathan et al. (Eds): ICCSEA, SPPR, VLSI, WiMoA, SCAI, CNSA, WeST - 2014
Detecting phishing on social media is a challenge because of (i) large volume of data – social media allow users to easily share their opinions and interests and large volume of data make it difficult to analyse (ii) limited space – social media often impose character limitation (such as Twitter’s 140 character limit) on the content due to which users use shorthand notations. Such shorthand notation is difficult to parse since the text is usually not well-formed; (iii) fast change – content on social media changes very rapidly making phishing detection difficult; and (iv) Shortened URLs – researchers have observed that more than half of the phishing URLs are shortened to obfuscate the target URL and to hide malignant intentions rather than to gain character space.

Recent statistics show that on an average, 8% tweets contain spam and other malicious content. It has been estimated that in the Kaspersky Lab report 37.3 million users experienced phishing in 2013 [1]. Also, 45% of bank customers are redirected to a phishing site divulge their personal credentials.

In our paper, we have developed a “Web Framework” in Python using Django through which we put a tweet id from Twitter as input, which processes as HTTP request for all the tweet with URLs and HTTP response as a result with legitimate and phishing URLs. We have been using several APIs integrated with the framework.

Along with these APIs we have also created another database for the “new URL” which has not been found in any of the above API. For classifying this “new URL” we use machine learning algorithm. The whole system is user friendly as user can easily input the tweet id or any keyword and we can get all the URLs with classification.

Further, this “Phishing Detection System” is time efficient, taking an average of only 0.501 seconds to detect phishing tweets with high accuracy of 94.56%. Such low computation times make it ideal for real world use.

2. NEED OF REAL TIME PHISHING DETECTION

Phishing is a harmful form of spam. These Phishing attacks not only cause the leakage of confidential information, but it also results in a huge amount of monetary losses [3]. Hence, it is important to build a realtime phishing detection mechanism for every OSM to protect its users.
As there is an increase in phishing attacks so we have to deploy a system in which we have built a Web framework for finding the tweet as phishing or safe.

In April 2013, an AP journalist clicked on a spear phishing email disguised as a Twitter email. The phisher, then hacked AP’s Twitter account. Stock markets plunged after a phony tweet about an explosion at the White House, erasing $136.5 billion of value from the S&P 500 index [4].

A most preferred solution used for Twitter by Lee et al. is the Warning-Bird system whose main focus is on suspicious URLs in general but not on detecting phishing. However, Warning Bird may fail if the spammers use a short redirect chain or multiple page-level redirects [5].

There is PhishAri technique which also works in real time, but now it is not in use because the extension could not work for the PhishAri API as now to access the Twitter data we have to firstly have authentication from Twitter using oauth and due to the appliance of various security parameters allowable to Twitter make it difficult to use [6].

After reviewing the above techniques, it was evident that there was very little work done to detect phishing on Twitter in real-time. To fill this gap, we designed and developed a “Web framework” to identify the particular tweet is phishing or safe.

3. TWYTHON

Twython is primary python wrapper for Twitter API, so that we can access the Twitter data easily and supports both normal and streaming Twitter APIs. In other words, Twython is used to query Twitter using the Representational State Transfer (REST) web API to get incoming replies and direct messages.

Twython, has two main interfaces:

* Twitter’s Core API (GET statuses/sample request)
* Twitter’s Streaming API (GET users/lookuprequest)

Search API is used for finding the tweets of a user, finding the tweets with specified keywords. We have a large number of queries, for that purpose Twitter streaming API is used. The RESTful API is useful for getting things like lists of followers and those who follow a particular user, and is what most Twitter clients are built off. In order to allow Twitter to monitor the number of requests we make, we need to follow an authentication protocol, Oauth [7].

Twitter API version 1.1 uses the concept of oauth, in which we need an authentication key whereas in version 1.0 we can access the data of Twitter directly [8]. This provides us the facility of making chrome extension using PhishAri API, which in version 1.1 is not possible.

```python
APP_KEY = 'YOUR_APP_KEY'
APP_SECRET= 'YOUR_APP_SECRET'
TWITTER=Twython (APP_KEY, APP_SECRET)
Oauth=TWITTER.get_authentication_tokens(callback_URL='http://abc.com/callback') [9]
```
4. API INTEGRATION

After getting the permission from Twitter using oauth, web framework is integrated with several APIs which are working in the background to detect whether the extracted tweets with URL are phishing or not. The APIs used are Phish tank API, Google safe browsing API, Mywot API. These APIs check for the phishing URLs using their own realtime database.

As new phishing techniques contain the short URLs, we can't detect them directly. LongURL API converts the short URLs from long URL, with the extra information added to it [10].

An addition tab “MORE” is used to load more tweets. It takes some time, to overcome with this limitation, we have used the concept of caching. We can store user requested tweets in cache memory, and checks for the update so that the data in cache memory on client site doesn’t become outdated.

Web frameworks also gives the facility of searching the tweets containing a specific keyword along with the specified user. Two text buttons are used for that purpose, and then GET request is sent to the Twitter using Twython, which helps us in accessing the user data, JSON type request is returned. “check_phish.py” file checks for the phishing URLs and gives the result. It shows a red background for phishing URLs and a green background for good URLs, and tweets which do not contain any URL have white background. Database of phishing URLs is stored, which is also referred whenever processing is done.

The machine learning algorithm we used is:

**Random Forest** - It is the most effective method of machine learning algorithms, it is a collection of CART-like trees specific rules for tree growing, tree combination, self-testing Trees are growing using binary partitioning[11]. This chooses some important set of features which makes it more accurate for classification[12].
As figure 3 specifies how a URL is classified as a phishing or safe URL. Tweets are fetched from the Twitter using Twitter Streaming API. LongURL API is used to get the long URL of the shortened URL, which give URL with additional information. It is first checked with the database, if it exist in database, it is labelled as phishing URL, if not then it is further send to different APIs, which check the URL in their database and gives output. Then, final verification is done by machine learning algorithm.

5. FEATURE EXTRACTION

5.1 Features for phishing Detection

Phishing is like a plague in social media. Past studies show that the phishing website can be detected through the analysis of the URL and the content of the website. Phishing websites, often appear identical to the legitimate website, but will generally have one or more characteristics by which we can find out that site are phishing. However, from the past studied it has been observed that the malicious users keeps changing the techniques they used for Phishing, making detection more difficult [13].

Database of phishing URL is used to extract more features and new URLs are classified accordingly. In short, it is learning from previous data.

5.1.1 URL based features

URL based features are defined for the analysis of the suspicious website. The length of the URL, no of dots, length of domain and subdomain, spelling, position of slashes used in the URL. These are some of the common features that help in detecting phishing websites (Table 1). The length of the URL of the phishing websites is normally longer than the legitimate website. A phishing URL contains more number of dots and sub domain than legitimate [14].
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of URL</td>
<td>Length of expanded URL in number of characters.</td>
</tr>
<tr>
<td>Number of dots</td>
<td>Number of dots (.) used</td>
</tr>
<tr>
<td>Number of subdomains</td>
<td>Number of subdomains (marked by /) in the expanded URL</td>
</tr>
<tr>
<td>Number of Redirections</td>
<td>Number of hops between the posted URL and the Landing page</td>
</tr>
<tr>
<td>Presence of conditional redirects</td>
<td>Whether the URL is redirected to different landing page for browser or an automated program</td>
</tr>
<tr>
<td>Spelling</td>
<td>use of “1” instead of “I” in URL</td>
</tr>
<tr>
<td>Slash</td>
<td>Number of (/) used</td>
</tr>
</tbody>
</table>

5.1.2 Tweet based features

Phishing tweets are designed in such a way so that they can get high visibility by carefully using tags. Twitter specific features are tweet content and its characteristics like length, hashtags, and mentions tags. Other Twitter Features used are the characteristics of the Twitter user posting the tweet such as the age of the account, number of tweets, and the follower-followee ratio (Table 2).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of @tags</td>
<td>Number of Twitter users mentioned in tweet</td>
</tr>
<tr>
<td>Presence of trending #tags</td>
<td>Number of topics mentioned which were trending at that time</td>
</tr>
<tr>
<td>Number of RTs</td>
<td>Number of times the tweet was reposted</td>
</tr>
<tr>
<td>Length of Tweet</td>
<td>Length of tweet in number of characters</td>
</tr>
<tr>
<td>Position of #tags</td>
<td>Number of characters of tweets after which the #tag appears</td>
</tr>
</tbody>
</table>

5.1.3 WHOIS based feature

WHOIS is a TCP based transaction-oriented query/response protocol that is widely used to provide information services to Internet users. It is widely used for querying databases that stores
the registered users or assigns of an internet resource, such as domain name, an IP address block (Table 3). Most malicious users register domains of websites from the same registrar, hence tracking the registrar may aid in detecting phishing. Therefore, we use WHOIS based features to further enhance our phishing detection methodology.

Table 3: WHOIS based feature

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ownership period</td>
<td>Age of the domain</td>
</tr>
<tr>
<td>Time taken to create TwitterAccount</td>
<td>How much time elapsed between creation of domain and the Twitter account</td>
</tr>
</tbody>
</table>

6. RESULT

As we developed a phishing detection system which uses several APIs as well as some features like URL based and Twitter based features to classify tweets accordingly as phishing or safe. In the next step, we create a real time phishing detection system by deploying a Web framework which makes a call to different APIs like Google safe browsing, Web of Trust API, etc. and then marks each tweet as phishing or safe.

In this section, we elaborate the results and observations based on the classification mechanism using the five set of feature sets with the database of phishing URLs. We have implemented random forest algorithm which learns from database.

6.1 Evaluation Metrics

In order to evaluate the effectiveness of our classification method based on the features described, we use the standard information retrieval metrics viz. accuracy, precision and recall. Precision of a class is the proportion of predicted positives in that class that are actually positive. Recall of a class is the proportion of the actual positives in that class which are predicted positive.

Table 4: Results of Classification experiment. We observe that Random forest has the best accuracy of 94.56%.

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Random Forest Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>94.56%</td>
</tr>
<tr>
<td>Precision (phishing)</td>
<td>96.24%</td>
</tr>
<tr>
<td>Precision (Safe)</td>
<td>98.23%</td>
</tr>
<tr>
<td>Recall (phishing)</td>
<td>93.21%</td>
</tr>
<tr>
<td>Recall (safe)</td>
<td>96.54%</td>
</tr>
</tbody>
</table>
Each entry in the table 5 indicates the number of elements of a class and how they were classified by our classification method. For example, ‘TP’ is the number of phishing tweets which were correctly classified as phishing. Using this confusion matrix, we can compute the precision and recall for both ‘phishing’ and ‘safe’ classes.

We also use the confusion matrix to compute the overall ‘accuracy’ of the classifier [15]. It is the ratio of the correctly classified elements of either class to the total number of elements.

Precision phishing = $\frac{TP}{TP + FP}$
Recall phishing = $\frac{TP}{TP + FN}$
Accuracy = $\frac{TP + TN}{TP + FP + TN + FN}$

6.2 Classification Results

We now describe the results of our classification experiment as described. We use the classification method for our study which is Random Forest. We present the results of classification task using all these methods.

From the 1,689 phishing tweets, we found that 1,573 tweets had unique text. Therefore, is our true positive dataset, we consider these 1,573 phishing tweets and 1,400 safe tweets chosen randomly from the tweets marked as ‘safe’ during our data collection process. We use this dataset for the rest of our classification experiments. We found that Random Forest classifier works best for phishing tweet detection on our dataset with a high accuracy of 94.56%. We also obtain a recall of 94.21% for phishing class and 95.82% for safe class. The results from the classification technique are described in the table 5.

We find that the superior performance of Random Forest for phishing detection on TWITTER also holds true with a high accuracy.

In table 5 we show that we could detect 94.31% phishing tweets correctly. However, we misclassified 8.5% of legitimate tweets as phishing tweets. The false negative percentage is low indicating that we classified only 6.78% phishing tweets as legitimate.

Table 5: Precision and recall for phishing detection using Random Forest based on all six feature sets

<table>
<thead>
<tr>
<th>Actual</th>
<th>Phishing</th>
<th>Prediction Safe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phishing</td>
<td>94.31%</td>
<td>8.5%</td>
</tr>
<tr>
<td>Safe</td>
<td>6.78%</td>
<td>95.41%</td>
</tr>
</tbody>
</table>

6.3 Evaluation of various Feature Sets

Most of the previous studies to detect phishing have used features based on the URL of the suspicious page and the HTML source of the landing page. In this study, we propose to use Twitter based features along with URL based features to quickly detect phishing on Twitter at zero-hour.
As described we have used six sets of features in table 6. To evaluate the impact of each feature set, we performed classification task by taking one feature set at a time and then added the other one in the next iteration.

Table 6: Informative features which we found for phishing tweet detection using Random Forest classification

<table>
<thead>
<tr>
<th>Feature Sets</th>
<th>Precision (Phishing)</th>
<th>Precision (Safe)</th>
<th>Recall (Phishing)</th>
<th>Recall (Safe)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>82.27%</td>
<td>88.72%</td>
<td>79.67%</td>
<td>92.35%</td>
<td>84.22%</td>
</tr>
<tr>
<td>F1+F2</td>
<td>87.11%</td>
<td>89.34%</td>
<td>82.89%</td>
<td>92.78%</td>
<td>89.35%</td>
</tr>
<tr>
<td>F1+F2+F3</td>
<td>92.21%</td>
<td>90.12%</td>
<td>85.29%</td>
<td>93.45%</td>
<td>91.18%</td>
</tr>
<tr>
<td>F1+F2+F3+F4</td>
<td>95.85%</td>
<td>92.35%</td>
<td>91.14%</td>
<td>94.35%</td>
<td>92.52%</td>
</tr>
<tr>
<td>F1+F2+F3+F4+F5</td>
<td>96.21%</td>
<td>94.62%</td>
<td>92.34%</td>
<td>95.45%</td>
<td>92.87%</td>
</tr>
<tr>
<td>F1+F2+F3+F4+F5+F6</td>
<td>97.85%</td>
<td>95.84%</td>
<td>93.56%</td>
<td>95.90%</td>
<td>93.15%</td>
</tr>
<tr>
<td>F1+F2+F3+F4+F5+F6</td>
<td>98.43%</td>
<td>96.21%</td>
<td>94.87%</td>
<td>96.56%</td>
<td>94.56%</td>
</tr>
</tbody>
</table>

We observe that when we use only URL based features, we get an overall accuracy of 84.22% and a low precision and recall for ‘phishing’ class. The addition of Twitter based feature sets, user based features and network based features significantly improve the performance of phishing detection and boost the precision of identifying phishing tweets significantly. Hence, Twitter based features are helpful in increasing the performance of classifying phishing tweets.

6.4 Most Informative Features

We now evaluate the most important features which help to decide whether a tweet is phishing or not. We use ‘scikit’ library to find out the most informative features. Random Forests deploy ensemble learning to evaluate the feature importance.

After each random tree is constructed using a set of features, its performance (misclassification rate) is calculated. Then the values of each features is randomly permuted (for each feature) and the new misclassification rate is evaluated.

The best performing features are then chosen as the most informative features.

The domains of malicious and phishing URLs tend to be short lived when compared to the domains of legitimate URLs in order to avoid detection. Similarly the age of Twitter account of the user posting phishing tweets is also generally less. Such users are often detected by Twitter and their accounts are suspended. However, using Phishing Detection System, we could detect a large number of phishing tweets by such users before they were suspended by Twitter[16].

6.5 Test Cases

Here are some test cases, to check the functioning of the webframework.

UserId- shwetachahar1
Keyword- India
Now we discuss how we can further improve Phishing detection for more efficient and robust phishing detection.

* Backend database for faster lookup: In future, we can maintain a cache backend database to capture tweets which have already been marked as either phishing or safe on Twitter. So, if the same tweet appears on Twitter, then we can skip the entire process of feature extraction and classification and lookup in our dataset of phishing URL and safe URL. This will also help us increase our own database of phishing tweets.

* As future work, it would be interesting to evaluate other feature ranking and selection techniques such as principle component analysis, latent semantic analysis, chi-squared attribute evaluation, etc. and other feature space search methods such as greedy backward elimination, best first, etc.

8. CONCLUSION

This phishing detection realtime web framework allows its user to easily access the tweets using tweet id or using any specific keyword containing in tweets using various APIs. Twython is used for accessing the Twitter data using oauth. This may take some time, for that concept of cache memory is used. For shortened URLs there is the need of longURLs API. Integration of APIs and machine learning algorithm together gives us result with an accuracy of 95.56%. Time taken for detection is a maximum of 0.501 Sec for a tweet. Various features including WHOIS, tweet, network based are under main considerations. White background is used for
tweets without URLs, red background for phishing URL, and green background for safe URL. This method can be improved by using more advanced features and database.

ACKNOWLEDGEMENT

We are very grateful to everyone who has given their valuable feedback and suggestions. This project has been done as a final year B.Tech project.

REFERENCES

AUTHORS

Nilesh Sharma received the degree in Computer Science from Hindustan College of Science & Technology, India and degree in Information Technology from IIIT Delhi, India. Currently, he is working as a software engineer at Dubizzle, Dubai. His interests are in python (Django web framework), Databases, JavaScript, JQuery and web designing.

Nishant Sharma is pursuing 5th year of Integrated Dual Degree (B.tech+M.tech) in Electronics & Communication from Indian Institute Of Technology, Roorkee, India. His interests include python, networking, communication and algorithms.

Vishakha Tiwari has completed her B.Tech (1st division with honors) in Computer Science branch from Hindustan College of Science and Technology, Mathura, India. She is placed as software developer in Contata Solutions, Noida, India. Her interests are in Java, Python and Databases.

Shweta Chahar has completed her B.Tech in Computer Science branch from Hindustan College of Science and Technology, Mathura, India. She is placed as program analyst in Cognizant Technology. Her interests are in Java, Python and Databases.

Smriti Maheshwari has completed her B.Tech in Computer Science branch from Hindustan College of Science and Technology, Mathura, India. She is placed as a graduate trainee engineer at HCL comnet Limited. Her interests are in Java and Python.