Identifying Malicious URLs

AN APPLICATION OF LARGE-SCALE ONLINE LEARNING

Vasu Jain | 40057063 | 10th December 2017
Jain.vasu92@gmail.com

Abstract

Malicious URLs have become a serious threat to user privacy, security and monetary loss. It has led to billions of dollars loss to the victims of mistaken clicks on these URLs. Thus, it has become imperative to detect suspected URLs that we come across while our online exposure. There are several blacklisting websites present online which helps categorize these malicious URLs. These websites are not exhaustive and can fail in categorizing or blacklisting newly created URLs. Thus, there is a need for some machine learning techniques to learn the common patterns present in all the currently identified malicious URLs such that it will become easier in future to label any URL as malicious or benign based on the previously learned common patterns. These patterns are derived from the URLs, and are mainly categorized as lexical and host-based. This project will involve identifying and studying these patterns continuously by getting a continuous feed and learning them to be able to predict for the newly fed unlabeled feeds through machine learning algorithm logistic and perceptron using SGD classifier and then comparing the result.

Introduction and Motivation

With a continuously increasing number of reported scams like monetary loss, theft of personal information, malwares etc. by unintentional access to malicious URLs online, it requires the user to become aware of the suspected links that he is clicking before-hand.

Below are some stats represented in pie chart and bar graph related to clicks on malicious URLs:

The above pie chart tells about the percentage of clicks on malicious or fraud URLs across the world in 2015 with US topping the chart with more than 1/3rd of the clicks.

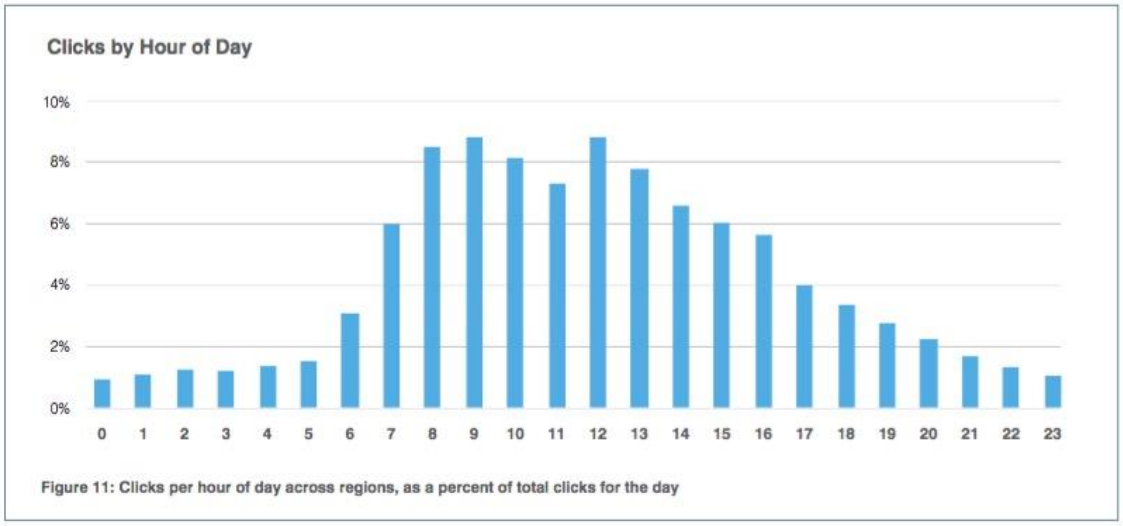


Figure 11: Clicks per hour of day across regions, as a percent of total clicks for the day

The above bar-chart describes the percentage of clicks on the malicious links on web every hour of the day.

According to another analysis, approximately one-third of all the websites are malicious. This makes it important for the presence of a rigid system to be able to predict with a decent accuracy whether the given URL is benign or malicious.

There are several blacklisting websites available which contains a database of all the URLs which are submitted by the users and voted by some others on the same. They have different mechanisms of building their database like by user reviews or self-confirmation. But still, blacklisting is not that effective that it should have been. This can be majorly due to the addition of countless new malicious URLs daily. The blacklisting doesn't get updated so quickly in contrast to the onset of malicious URLs. It takes time for the addition of a suspected new URL to be added in the blacklist database after user reviews and confirmation and before that the URL already affects thousands of victims.

Thus, numerous machine learning algorithms have been applied to counter this issue and be able to make predictions whether a new URL is safe or not. The algorithms require training to be done on a prespecified data containing both malicious and benign URLs labelled respectively. Before the URLs can be fed to the algorithm, feature generation and data preprocessing is required to be done.

But first of all, I required a big dataset of labelled URLs which I received from various online sources which keep a set of all malicious or benign URLs. From this set of thousands of labelled URLs, I will create or extract multiple features which will be fed to the algorithm. There is a lot of information present in the URLs which can be put to use in leaning. On a broad level, there are 2 types of features - lexical and host-based features. I will discuss about both one by one in this report and also how each types of features were exactly extracted and put to use. After the feature generation is done, I will train Logistic regression with stochastic gradient descent algorithm to be able to predict any future unlabeled URL. The training will be online i.e., Newer data will be continuously fed to the algorithm to retrain and update it regarding any new information for any feature that can be useful for it to make better predictions. Finally, my algorithm will be able to make predictions with a decent accuracy which can be maximized in future using various techniques of optimization which is out of scope for this project.

Feature Extraction

Initially the data was extracted from multiple sources in the form of URLs. A set of both malicious and benign URLs were taken and labelled accordingly. There are multiple sources which provides a list of identified and verified malicious/benign URLs:

There are websites which have an input and review systems for verification and classification of malicious URLs. Submissions by multiple users are verified by registered users of the forum and then reviewed by the forum team. Then finally those URLs are included in the online database of malicious URLs accessible to everyone.

Examples of such web applications include: Malware domains, malware patrols, phishtank and some more.

(*Links to the above data sources are mentioned in the References section*)

After collecting the data, the most important part i.e., feature extraction from the data(URLs) is to be done. There are multiple types of features that can be extracted from URLs:

1) **Lexical**

   Lexical features are derived from the URL name. It contains a lot of useful information about a URL that can be put to use for our predictions.

   a) URL name – total length of the URL was considered as a feature. As the length of the URL is an important feature which separates a malicious URL from a benign one.
   Bag of words were extracted based on special characters in the URL name. These words or tokens were further analyzed for token length, average token length and largest token.
   Hostname and the corresponding URL path from the URL was segregated and the same procedure was done individually as well.
   URL was also checked if it contained any exe in it. As the presence of any exe file can cause serious malware attack when the URL is hit.
   Number of dots present in the URL is also counted. As more the number of special character like a dot, the more the URL is suspicious.

   b) Host name: the name of the host was extracted which was further used to check the size of the host. Also, presence of any IP address in the hostname was analyzed as it can also be a reason of suspicion.
   c) Path – it is the additional path inside the URL after the host name containing usually the path for any page or file which might be present in the host. Length

of the path along with the average and largest token length in the path was evaluated.

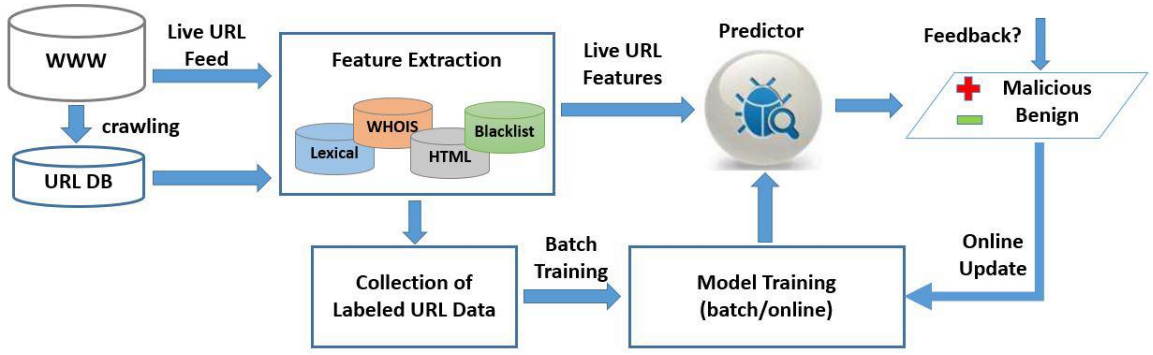
d) Ranking – ranking of all the registered web URLs is maintained by alexa which is a subsidiary of amazon. So, the host name is sent to the alexa for the rank of that particular URL. There are two ranking system returned namely: host overall rank and country rank. This is also a major feature which makes a distinction between malicious and benign URLs as most of the malicious URLs have very poor rank or are unranked.

2) **Host-Based features**
   a) ASN – Autonomous system number is the official registration number for each ISP. These are important as these uniquely identify each network on the internet. So, ASN number for each host is fetched to check whether there exists a registration number for it. There is a list of ASNs provided by maxmind *(references section).* So, for each URL ASN number is fetched using GEOIP in library *pygeoip* and matched against its presence in the verified list of ASNs provided by maxmind.
   b) IP address exist check – This is a Boolean feature to indicate whether an IP address is linked to the given domain name or not.
   c) Google safe browsing – Google Safe Browsing is a blacklist service provided by Google that provides lists of URLs for web resources that contain malware or phishing content (*References Section*). I used the API to return google API result for all the URL and it is also a strong feature for categorization.
   d) Domain age – domain age is calculated in number of years for the domain to become active. It is a good indicator for benignity of a URL as the more the number of years since it was created, more is the trust factor on it. The information for the creation date is fetched using whois library.

3) **Content-based features**
   a) HTML based – There is a lot of information that the HTML of any URL can give which can be useful in classifying whether a URL is malicious or benign. The number of HTML tags present, hidden or zero-sized iframes are also used by malicious programmers to hide links to some malicious websites which will be opened when one opens the original URL. Also, presence of large number of hyperlink tags is considered as a feature.
   b) JavaScript-based – there are various JavaScript functions that are executed without the client permission to execute unwanted procedures by malicious programmers. Some of the functions are: eval(), unescape() which can indicate the execution of encrypted code within the HTML. Other such methods are escape(), link(), exec() and search(). So, a count of usage of these functions is maintained as a feature.

After a list of all these features for all the URLs is obtained. I need to remove from the list those data points which does not have the desired values returned from the operations done above. There might be number of such cases as for a lot of URLs, host based information is not available. Thus, such data points can become a hindrance in the accuracy of training and prediction.

The below diagram indicates the process of fetching URLs and then converting them into a feature set, then sending the same to the predictor algorithm to get the label for the same.



Data Preprocessing and Classification

Classification Algorithms

1) **Perceptron:** It is a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time. It gives a decent output prediction for my model. But it suffers from a drawback that the update rate for a perceptron is fixed and thus it does not take into account the magnitude of classification error. Hence, while making the prediction it may not adapt well to the change in data and make a drastic change even when the error is small or vice versa.

$$w_{t+1} \leftarrow w_t + \mathrm{y_t}\, x_t$$

Here the learning rate is fixed.

2) **Logistic Regression:** Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. The logistic regression model is simply a non-linear transformation of

the linear regression. The "logistic" distribution is an S-shaped distribution function which is similar to the standard-normal distribution (which results in a probit regression model) but easier to work with in most applications (the probabilities are easier to calculate). The logit distribution constrains the estimated probabilities to lie between 0 and 1. The update resembles a Perceptron, except with a learning rate that is proportional to the difference between the actual and predicted likelihood that the label is +1.

$$w_{t+1} \leftarrow w_t + \Upsilon \Delta_t x_t$$

where $\Delta_t = \frac{y_t+1}{2} - \sigma(w.x_t)$ and Υ is the varying learning rate.

I have used the Stochastic gradient descent classifier function for classifying using both the above classifiers. SGD, also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions.

With the raw data I obtained as per the previous section of "Feature extraction", I tried classifying using the both the classifiers using SGD function of *sklearn* library. I got the below results:

Classifier	**Test Error**	**Classification**
Logistic Regression	28%	42%
Perceptron	27%	42%
Logistic with SGD	35%	45%

The results mentioned in the above table is not at par with the expectations from the classifiers. It was thus required to tune the data. Below are some of the actions performed to the raw data to improve the performance:

1) **Online Learning**
   What is online Learning?
   Online learning is a method of machine learning in which data becomes available in a sequential order and is used to update our best predictor for future data at each step, as opposed to batch learning techniques which generate the best predictor by learning on the entire training data set at once. Online learning is a common technique used in areas of machine learning where it is computationally infeasible to train over the entire dataset, requiring the need of out-of-core algorithms. It is also used in situations where it is necessary for the algorithm to dynamically adapt to new patterns in the data, or when the data itself is generated as a function of time.
   How is it useful in context of my project?

The data in case of malicious URL keeps updating online i.e. there are various types of new malicious URL that keeps coming up daily. Each malicious programmer uses some new tricks or techniques to safe himself from getting caught by multiple online applications which filters these URLs. Also, there are a lot of URLs which becomes inactive continuously. So, there is a need for getting the updated data to be fed to already trained algorithm so that it can learn new feature values from new data.

In the scope of my project, Logistic regression and Perceptron was trained using online learning technique wherein chunks of data will be fed to both the algorithms and accuracy is calculated. The coefficient for the Stochastic gradient descent algorithm was stored after execution for each data chunk. It is stored in a pickle file format in python. Consequently, for each subsequent training dataset, it is checked if the value of coefficient from the previously trained algorithm (for both logistic and perceptron respectively) exists, if it does then that coefficient is sent as a parameter to the fit function which takes that coefficient as the initial coefficient and calculates the new coefficient starting from the it with a constant learning rate in case of Perceptron and varying learning rate as in the case of Logistic regression as mentioned above.

2) **Data selection** – The data that I extracted for all the URLs had some data points for which complete information was not received from the various online sources like API and libraries that I was using to fetch the data as there are various Malicious or benign URLs which are either not active or not properly registered to get all the parameters for it. Hence, I segregated all those data points which had '-1' as the output which was set for any exception or any feature returning no output majorly for host-based or content-based data.
3) **Feature Conversion** – The values for the different features that is created requires to be brought into a range or a value which can be beneficial or more useful for the predictor. For example: For the feature "rank" there were many rows for which I did not get any rank from the alexis API as those might be unranked by it. Hence for those features a value should be assigned such that it does not impact poorly the performance of the classifier. It means a value of zero there would be misleading in that case as it would mean for a malicious URL lower the value, more is the chance of it being a benign URL. Hence, a value of '-1' is assigned to it.
4) **Data Scaling** – Feature scaling is a method used to standardize the range of independent variables or features of data. The data features that I generated had range values from a single integer zero (count of null iframe) to thousands (ranking of websites). In order to get a good prediction feature scaling was impertinent in this case. Thus, all the features were normalized in the ranges 0 to 1 by diving each value of every feature by its mean.
5) **Data Shuffling** – It was also observed that while training the algorithms with the data, shuffling of data points increased the accuracy as the dividing the data into

training, testing and validations points required a mix of both the classes to allow a good training on both the classes as well as points for validation of classification.

Final Results

Below are the results obtained after applying above techniques but using batch learning, i.e. giving the data only once to the trainer and without using any initial coefficient:

Classifier	**Test Error**	**Classification**
Logistic Regression	96%	94%
Perceptron	95%	91%
Logistic with SGD	97%	95%

Below are the results obtained after applying all of the above techniques and when the algorithm is executed in online learning where in the coefficient from previous run is stored and taken as the initial coefficient for the next dataset:

Classifier	**Test Error**	**Classification**
Logistic Regression	93%	98%
Perceptron SGD	96%	98%
Logistic with SGD	95%	98%

Please note that the results might change a little when executed again due to the learning coefficient which is updated at each execution and k-fold which randomly selects data for each fold. To test the application again, for fresh set of data without the existing learning coefficient, you will need to delete the files *logis.pk* and *percep.pk* which is for storing individual coefficients of logistic and perceptron respectively.

Execution time is varying. For feature generation the program will take approximately 1 hour for 1500 URLs as the data needs to be fetched from online sources. For classification algorithm, it will take approximately 5 minutes for the same number of records.

Conclusion and Future Work

Malicious Web sites are a prominent and undesirable Internet scourge. To protect end users from visiting those sites, the identification of suspicious URLs using lexical and host-based features is an important part of a suite of defenses. By seeing the output of the

algorithms, it can be safely concluded that online learning technique is most suitable with URL classification coupled with other optimization techniques as per the results and reports.

Future work may include collection of more features to gain a higher accuracy coupled with applying various other classification algorithms.

References

1) Identifying Suspicious URLs: An Application of Large-Scale Online Learning, **Justin Ma, Lawrence K. Saul, Stefan Savage, Geoffrey M. Voelker**
2) Malicious URL Detection using Machine Learning: A Survey, **Doyen Sahoo, Chenghao Liu, and Steven C.H. Hoi**
3) Detecting Malicious Web Links and Identifying Their Attack Types, **Hyunsang Choi, Bin B. Zhu, Heejo Lee**
4) PhishDef: URL Names Say It All, **Anh Le, Athina Markopoulou, Michalis Faloutsos**
5) Sources of malicious and benign URLs
   http://apac.trendmicro.com/apac/security-intelligence/current-threat-activity/malicious-top-ten/
   https://www.phishtank.com/phish_archive.php
   http://www.malwaredomains.com
   https://www.malwarepatrol.net
   http://www.joewein.de/sw/blacklist.htm
   https://db.aa419.org/fakebankslist.php?start=21
   https://moz.com/top500
6) Ranking feature of URLs
   http://data.alexa.com
7) Source for List of ASNs
   Maxmind's database – *www.maxmind.com*
8) Google safe browsing API for safe browse feature
   https://sb-ssl.google.com/safebrowsing/api