**Detecting Phishing Attacks Using Natural Language Processing and Machine Learning**

**Abstract:**

Malicious Web sites largely promote the growth of Internet criminal activities and constrain the development of Web services. As a result, there has been strong motivation to develop systemic solution to stopping the user from visiting such Web sites. We propose a learning based approach to classifying Web sites into 3 classes: Benign, Spam and Malicious. Our mechanism only analyses the Uniform Resource Locator (URL) itself without accessing the content of Web sites. Thus, it eliminates the run-time latency and the possibility of exposing users to the browser based vulnerabilities. By employing learning algorithms, our scheme achieves better performance on generality and coverage compared with blacklisting service.

URLs of the websites are separated into 3 classes:

* Benign: Safe websites with normal services
* Spam: Website performs the act of attempting to flood the user with advertising or sites such as fake surveys and online dating etc.
* Malware: Website created by attackers to disrupt computer operation, gather sensitive information, or gain access to private computer systems.

**Existing System:**

A poorly structured NN model may cause the model to under fit the training dataset. On the other hand, exaggeration in restructuring the system to suit every single item in the training dataset may cause the system to be over fitted. One possible solution to avoid the Over fitting problem is by restructuring the NN model in terms of tuning some parameters, adding new neurons to the hidden layer or sometimes adding a new layer to the network. A NN with a small number of hidden neurons may not have a satisfactory representational power to model the complexity and diversity inherent in the data. On the other hand, networks with too many hidden neurons could over fit the data. However, at a certain stage the model can no longer be improved, therefore, the structuring process should be terminated. Hence, an acceptable error rate should be specified when creating any NN model, which itself is considered a problem since it is difficult to determine the acceptable error rate a priori. For instance, the model designer may set the acceptable error rate to a value that is unreachable which causes the model to stick in local minima or sometimes the model designer may set the acceptable error rate to a value that can further be improved.

**Disadvantage:**

1. It will take time to load all the dataset.
2. Process is not accuracy.
3. It will analyse slowly.

**Proposed System:**

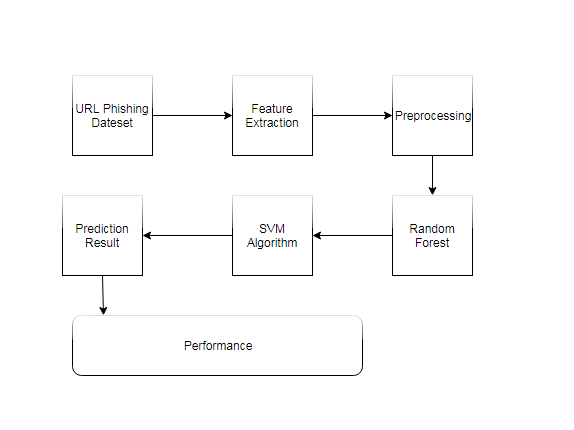
Lexical features are based on the observation that the URLs of many illegal sites look different, compared with legitimate sites. Analysing lexical features enables us to capture the property for classification purposes. We first distinguish the two parts of a URL: the host name and the path, from which we extract bag-of-words (strings delimited by ‘/’, ‘?’, ‘.’, ‘=’, ‘-’ and ‘’).

We find that phishing website prefers to have longer URL, more levels (delimited by dot), more tokens in domain and path, longer token. Besides, phishing and malware websites could pretend to be a benign one by containing popular brand names as tokens other than those in second-level domain. Considering phishing websites and malware websites may use IP address directly so as to cover the suspicious URL, which is very rare in benign case. Also, phishing URLs are found to contain several suggestive word tokens we check the presence of these security sensitive words and include the binary value in our features. Intuitively, malicious sites are always less popular than benign ones. For this reason, site popularity can be considered as an important feature. Traffic rank feature is acquired from Alexa.com. Host-based features are based on the observation that malicious sites are always registered in less reputable hosting centres or regions.

**Advantage:**

1. All of URLs in the dataset are labelled.
2. We used two supervised learning algorithms random forest and support vector machine to train using scikit-learn library.

**System Architecture:**



**Hardware and Software Requirements:**

**Hardware:**

1. Windows 7,8,10 64 bit
2. RAM 4GB

**Software:**

1. Data Set
2. Python 2.7
3. Anaconda Navigator

Python’s standard library

* Pandas
* Numpy
* Sklearn
* tkMessageBox
* matplotlib