Predicting Malicious URLs

Proofpoint, Inc.

Aidan Cheng, Kevin Herrera, Carli Lessard, Vidushi Ojha (PM)
Liaisons: Thomas Lynam, Mike Morris '97 | Advisor: Prof. Sweedyk

Problem Statement

Proofpoint, a leader in cybersecurity, provides a service to screen client emails and detect embedded URLs that are malicious. Our task was to investigate various machine learning models to improve the accuracy of the classification process.

Goals

Proofpoint's current system scans for embedded URLs and sends suspicious ones to a sandbox, a sped-up virtual environment where any malicious effects can be observed. We aimed to:

- Assign each URL a **probability** of being malicious
- **Maximize** the number of URLs classified correctly
- Specifically, correctly classify **over 70%** of the malicious samples

Models

We investigated three different machine learning models:

- Support Vector Machines (SVM)
- Naive Bayes
- Logistic Regression

The SVM approach was deemed too slow to be a viable option.

Features

We used URLs and IP addresses as features for our models. Feature extraction consisted of tokenization on punctuation and character (for URLs) or word (for IP addresses) *n*-grams.

```
https://www.cs.hmc.edu                      134.173.193.149

'https'        'http'                       '134'    ['134', '173']
'www'          'ttps'                       '173'    ['173', '193']
'cs'           'tps:'                       '193'    ['193', '149']
'hmc'          ...                          '149'
'edu'          '.edu'
```

Testing

We tested our models **offline** as well as **online**. We developed a **pseudo-online** algorithm that tests 1000 samples at a time, and then uses those samples together with other recent malicious samples to create a new training set.

Experiments

Model:	Data:	Testing:	Parameters:
Naive Bayes	URLs	Offline	E.g. iterations, malicious set size, etc.
Logistic Regression	IP addresses	Pseudo-online	

Naive Bayes

Algorithm: This model uses Bayesian probability rules to predict, based on extracted features, whether a sample is more likely to be clean or malicious.

Tools: We use scikit-learn's Naive Bayes implementation.

Results: A number of experiments were successful at surpassing 70%, such as combining URLs and IP addresses.

	% of malicious URLs correctly classified	% of clean URLs correctly classified
Offline URL	37.7%	95.5%
Offline IP	75.8%	96.4%
Online URL	55.9%	92.7%
Online IP	85.3%	53.2%
Online URL & Offline IP	78.1%	95.1%

Logistic Regression

Algorithm: We used a logistic regression model and gradient descent optimization to determine the best coefficients for our model.

Tools: We used Google's open source library TensorFlow to construct our model.

Results: We tried different combinations of using the URL and IP address as features. The features that worked best was combining the URL and the first three octets of the IP address.

	% of malicious URLs correctly classified	% of clean URLs correctly classified
Online URL	78.6%	80.2%
Online IP	52.2%	94.2%
Online URL & IP	78.4%	90.6%

Combining Classifiers

Algorithm: We used the probability of being clean produced by Naive Bayes as a feature within the Logistic Regression model.

Results: Not significantly better than either classifier.

	% of malicious URLs correctly classified	% of clean URLs correctly classified
Bayes + URL + IP	78.4%	90.4%
Bayes + URL	80.1%	80.9%
Bayes + IP	65.2%	90.0%

We would like to thank our advisor, liaisons, and the CS staff for all their help and support on this project, and credit the TensorFlow and scikit-learn libraries.