Engaging the Security Community with Open Data Hacking Project

This is the first in a series of posts about the new public Click Security GitHub project Data Hacking. The project utilizes an open architecture based on Python and the most recent advances in data analysis, statistics, and machine learning. We investigate challenging security issues through a set of exercises that use open datasources and popular python modules such as Pandas, Scikit Learn, and stats models. All materials are presented within a set of iPython notebooks that are shared publicly.

Exercise: Detect Algorithmically Generated Domain Names

The Data Hacking Github project page already has several posted exercises but we'll begin with an exercise to detect Algorithmically Generated Domain Names.

Resources

- Main Page
- DGA GitHub
- DGA Notebook

Python Modules Used:

- iPython: Architecture for interactive computing and presentation
- Pandas: Python Data Analysis Library
- Scikit Learn Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
- Matplotlib: Python 2D plotting library
- StatsModels: descriptive statistics, statistical tests, and plotting functions.

In this notebook we're going to use some great python modules to explore, understand and classify domains as being 'legit' or having a high probability of being generated by a DGA (Dynamic Generation Algorithm). We have 'legit' in quotes as we're using the domains in Alexa as the 'legit' set. The primary motivation is to explore the nexus of iPython, Pandas and scikit-learn with DGA classification as a vehicle for that exploration. The exercise intentionally shows common missteps, warts in the data, paths that didn't work out that well and results that could definitely be improved upon. In general capturing what worked and what didn't is not only more realistic but often much more informative. :)

The DGA Notebook contains all the code and details of the exercise but we'll summarize the work and approach here.

Datasources

- Alexa 100k top domains (we also show results for top 1 Million).
- A mixture of ~3500 domains that were known to come from DGA sources.

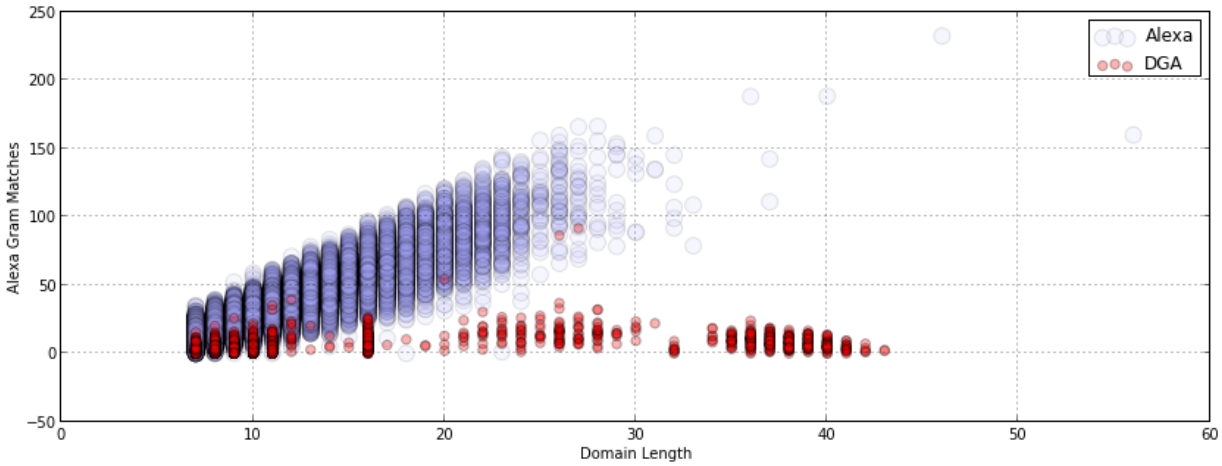
Approach

- Data Ingestion, Cleanup and Understanding
    - Show the power/flexibility of Pandas python module by reading in, processing and cleaning the input data with a couple lines of python.
    - We compute both length and entropy and add those to our Pandas data frame.
    - Demonstrate the nice integration of iPython/Pandas/Matplotlib by showing several plots of the resulting data (BoxPlots, histograms, scatter plots).
- Utilize Scikit Learn Machine Learning Library
    - Random Forest: popular ensemble machine learning classifier
    - Train/Classify: We demonstrate the classification results on feature vectors containing just the length and entropy. The results show that prediction performance is extremely poor given just those features.
- Incorporate NGrams:
    - We show the use of scikit learn's CountVectorizer to compute NGrams on both the Alexa domains and on the english dictionary.
    - We perform some Numpy matrix operations to capture a count vector
    - Those new features are added to data frame and feature matrix for scikit learn.

Results

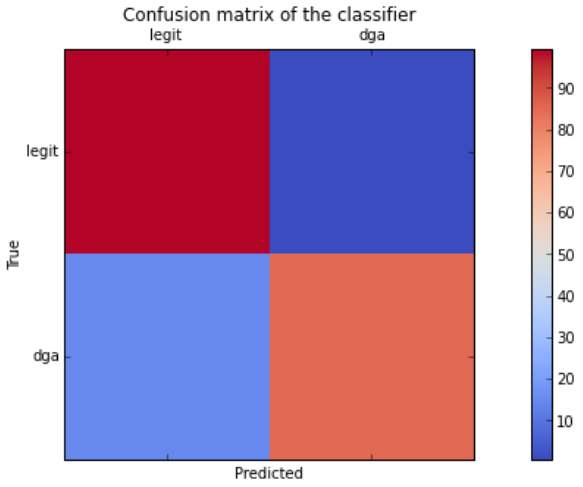
For an exercise where the focus was to demonstrate the utilization of iPython, Pandas, Scikit Learn and Matplotlib, the results were reasonably good.

We can plot our new NGram features to determine differentiation between classes:



Given a feature matrix of length, entropy, alexa_ngrams, and dict_ngrams our classifier had a predictive performance on our holdout set of the following:

```
Confusion Matrix Stats
legit/legit: 99.38% (6723/6765)
legit/dga: 0.62% (42/6765)
dga/legit: 14.61% (39/267)
dga/dga: 85.39% (228/267)
```



We can see that 'false positives' (legit domains classified as DGA) is quite small at 0.62%. This is critical in a large scale system where you don't want false alerts going off for legitimate domains.

Well that summarizes the results in a nutshell but the DGA Notebook gives a thorough, in-depth treatment of the data, features, analysis and machine learning done for this exercise.

Please visit the new Click Security Data Hacking GitHub site for additional exercises, code, and iPython notebooks.

-Cheers
Brian Wylie (Lifeform at Click Security)