OPERATIONALIZE MACHINE LEARNING TO DETECT MALICIOUS DOMAIN NAMES
Introduction

In today's security landscape the detection of advanced threats is getting more difficult as patterns of attacks are getting faster. Classic approaches that mainly rely on static matching, such as searches against blacklists or regex patterns, can have certain limitations in terms of flexibility or fuzziness to detect malicious artifacts in machine data. This is where machine learning techniques can add value and provide new insights and higher detection rates. This paper describes a step-by-step process on how machine learning can be leveraged to detect malicious domains and help expand existing security use cases with the Splunk platform.

Recognizing non legit domain names is helpful to detect indicators of compromise due to typical malware communications such as botnets. This paper will walk through the steps to operationalize machine learning with Splunk to automatically detect domain generated algorithms (DGA). The first part of the paper will cover data exploration and feature engineering to get everything in place to continue with machine learning. The second half of the paper will create, evaluate and operationalize machine learning for DGA detection.

1. Data Exploration

The first dataset starts with labeled domain names that indicate whether a domain is legit or created by some DGA. We have around 60 percent domain names from legit domains and the remaining 40 percent split across three DGA subclasses that correspond to different botnet types, like cryptolocker, goz and newgoz.

Based on a text mining approach we can explore the domain name strings with n-gram analysis and principal component analysis (PCA) to see if there are certain patterns in the structure of domain names. Using the TFIDF algorithm with ngram parameter set to length of 1-3 characters we obtain a matrix of most frequent character groups in the domain name strings. This yields a high dimensional result which is hard to interpret (by default 100 dimensions). We can reduce the dimensions by applying a PCA to get a useful representation that can be visualized in a 3D scatter plot. The SPL is just five lines to achieve this:

```
| inputlookup domains_pro.csv  
| sample 0.5  
| fit TFIDF analyzer=char ngram_range=1-3 domain into “dga_tfidf”  
| fit PCA domain_tfidf* k=3  
| table label PC_*
```
The image above shows legit domain names represented by the blue cluster and malicious ones that are shown as separated clusters (yellow, red and purple). This indicates that some DGA subclasses like newgoz (yellow) are more separable than others (red and purple) which means that newgoz can be detected with higher accuracy compared to cryptolocker and goz from the given structure. The results of this approach can already be used as first numeric features that are calculated from the domain name string to determine how the domain is related to the given characteristics of DGA subclasses.

2. Feature Engineering and Selection

Because it is hard to create a machine learning model just from the given domain name strings, we have to create additional features that further characterize the domain names. Using the URL Toolbox app and SPL, it is easy to calculate various metrics like shannon entropy, rate of known words in a dictionary, or even simple things like the length of the string or consonant ratio. We also add the persisted TFIDF model from the data exploration part.

```
| inputlookup dga_domains
| eval label=class.".".subclass
| `ut_shannon(domain)`
| `ut_meaning(domain)`
| eval ut_digit_ratio = 0.0
| eval ut_vowel_ratio = 0.0
| eval ut_domain_length = max(1,len(domain))
| rex field=domain max_match=0 "(?<digits>\d)"
| rex field=domain max_match=0 "(?<vowels>[aeiou])"
| eval ut_digit_ratio=if(isnull(digits),0.0,mvcount(digits) / ut_domain_length)
| eval ut_vowel_ratio=if(isnull(vowels),0.0,mvcount(vowels) / ut_domain_length)
| eval ut_consonant_ratio = max(0.0, 1.000000 - ut_digit_ratio - ut_vowel_ratio)
| eval ut_vc_ratio = ut_vowel_ratio / ut_consonant_ratio
| fields - digits - vowels
| apply "dga_tfidf"
| fit PCA domain_tfidf* k=3
| fields - domain_tfidf*
| rename PC_* as TFIDF_PCA_*
| outputlookup dga_domains_features
```
As part of this data preprocessing, we save the results in a temporary lookup table (dga_domains_features). To identify features that are the most promising ones for prediction, we used the SPL command analyzefields to get a rating of all the features and pick the top ones with highest accuracy for creating machine learning models later on (bar chart in the image on bottom right). If domain names are more complicated, the URL Parser app can be helpful to extract relevant parts.

The image above shows the table domain names which have been enriched with additional features. The bubble chart on the lower left explains how 2 selected features (Shannon entropy and meaning ratio) correspond to legit domains and those from DGA subclasses. To show the distribution of classes and subclasses according to the selected features, we can quickly visualize the dependencies in a parallel coordinate chart. This helps explore the relationship of features and detect patterns in a multidimensional dataset.
Another helpful method to identify useful feature combinations is to plot them in a scatterplot matrix. Here we can quickly spot features that show a distinct distribution and therefore are more likely to lead to an accurate prediction. For example, we can plot the features of the URL Toolbox results against each other:

In the feature engineering and selection phase we have seen how the given data set can be explored and enriched with additional features. Of course, more features like Alexa ranking, domain age, custom blacklists and whitelists can be added to give even more precision.

We described how we prepared the dataset and identified features for machine learning. In the next section we walk through the creation, evaluation and operationalization of machine learning to automatically detect DGA generated malicious domain names.
3. Create and Evaluate Machine Learning Models

Our goal is to classify domain names as malicious or legit based on the features we created. Machine learning models can be trained with different algorithms to evaluate which ones deliver the best accuracy. In our case, we will examine Logistic Regression, Support Vector Machine, Random Forest and Decision Tree.

First we equally split our dataset with a random sampling in a training and test set. Second we fit the algorithms above and persist into models that can later be reused with the apply command. Third we evaluate the results based on how successful each algorithm behaves in terms of the correct predictions and false positive true negative rates, which can easily be read from the confusion matrices of each algorithm. To get started, we use the assistant to predict categorical fields that ships with Splunk’s Machine Learning App and try logistic regression:
As we can see, our results are not too bad for a first run. We achieved a combined error rate of false classifications around 2 percent. Precision, recall, accuracy and the F1 score is at 98 percent, so there is still some headroom for improvements. Let's consider the other algorithms and evaluate how these performed under the same conditions:

The charts show that Random Forest performs best in terms of the lowest number of false positives and combined results in terms of lowest prediction error rate. However, the Support Vector Machine performs in a similar range and may also be considered equally suitable for this data set after further tests. For example, after considering cross validation of both algorithms. The least accurate algorithm is logistic regression in this case. Now let's see how we can operationalize our model to work with the most recent data.

4. Operationalize the Model
After we trained our models, we can use them with the apply command to make them work with DNS data from production systems. To apply our model, we need to calculate the features we engineered in the first part. Depending on the desired latency we use a scheduled search to apply our model to the data in the defined timespan. For demonstration purposes, in the app available on splunkbase, we generate a random set of sampled domain names every minute and apply machine learning in a real-time search. We can see the trend of true and false predictions over time and display the results that can further be checked by an security analyst in detail.
With our goal in mind to reduce the information that needs to be analyzed, we can filter the list of our results to show only the predicted DGAs to investigate if those are a security concern. We extended the GUI with actionable buttons to empower the security analyst to provide feedback into the system. By clicking LEGIT, the analyst can whitelist results that have been classified as DGA incorrectly and reduce further false positives.
With this mechanism we generate a second level classification that can be used in different ways. First, we can add the manually verified results to our training data set and increase the detection quality of our models as we train them continuously. This helps us keep our models up to date and it can grow our training dataset over time. Second, we can use the legit flagged domains in a whitelist and pre-filter or reuse those lists in other areas. Third, we can use the verified classification to grow our blacklist for exact matching. All these approaches combined lead to a living thread list that we can maintain and accommodate to our environment and organizational specifications. The following scheme summarizes the above outlined logic.

5. Wrap Up, Disclaimer and Credits

It is important to emphasize that this approach is an example how machine learning can be applied in a specific security use case. The presented results are not meant to be complete in terms of a ready to use solution that works out of the box in any environment and they are not intended to detect 100 percent of unknown malicious domains. Machine learning approaches heavily depend on the quality of the dataset that is used for model training. The above example would be a good starting point to review available data sets that can be used for this purpose and take the first steps. The Splunk app can be found on Splunkbase.

To get started download the app from Splunkbase and reach out to your sales rep for further questions.