Cyber Attack Predictions Using AlienVault OTX Threat Intel

Cody Brown
Virginia Tech 800 Washington St. SW Blacksburg, VA 24061
(707) 228-7420
cody@securityresearch.us

Introduction

I am looking to establish a prediction algorithm for cyber-attacks based on previously observed behavior. I believe it is possible to predict attacks based on reconnaissance. For example, if an actor scans port 23 from an attributed IP address and then throws a telnet exploit 30 days later from a different IP address, then it is possible to predict the attack based on this technique.

Project Problem Statement

Currently, attack prediction is based almost entirely on the occurrence of other attacks in the world. For example, if businesses A, B, C, D, and E get hit with a certain exploit and business F is like the others in several ways, then it is possible to guess that business F should be worried. Because this method involves classifying the target of an attack, the resulting data becomes nonapplicable to dissimilar potential targets.

Problem 1:

I will have to cluster the data into various reconnaissance and attack categories using K-means clustering or classification. The clustering/classification will rely on a comprehensive bag of words to distinguish clusters/classes.

Problem 2:

Once clustered, I will have to process the data a second time to create a combination class that shows the relationships between reconnaissance and attacks from the same entities. I will use linear regression on the output from this to determine whether an incident is imminent based off a reconnaissance event.

Datasets

For this project, I chose AlienVault OTX data because I knew it would be easy to check for accuracy. OTX collects data from partners in the form of objects called “pulses”. These “pulses” have a type or category associated with them. Based on their type, they will have a set of tags associated with them. There are 23 types, and several thousand tags. I knew that I could standardize the types, and that the tags were strongly associated with each type.

Dataset #1: Raw data from AlienVault OTX

Number of data points: 176,456

Number of features: 193 (Bag of Words)

The results of Dataset #1 were taken and preprocessed into a class that maps scan recency to attack types. If scans have occurred within a time threshold, I would like to know if this has any relation to certain kinds of attacks.

Dataset #2: OTX Data that has been classified into reconnaissance/attack categories then mapped to a recon-date to attack type class

Number of data points: Random, depending on specific adversary activity

Number of features: 21

Notes

Unfortunately, after spending enormous amounts of time preprocessing the data, I found that it was not a very good dataset for my original classification problem. It seems that there is a hard relationship between tags and types. In other words, the classification problem had already been solved. For example, if you submit a "category 8 event" type object, you can only tag it with a selection of 13 tags that overlap heavily with other similar objects. I discovered this after I started getting near perfect numbers.

The post-classified data was helpful in reaching a conclusion to the second regression problem, however.

Preprocessing steps

Preprocessing this data was extremely difficult. I received an original dataset that consisted of 12 fields. There was one main field that I concentrated on for the classification problem. This field, labeled "tags", contained a randomly sized list of tags for that data point. After using AlienVault's API to download more than 176,000 entries, I selected a set of 193 popular tags in a random sampling of all 23 types of events. I then used a PowerShell algorithm to preprocess the data efficiently, converting each list of tags to a set of fields named after each popular tag. Sadly, I could not get Python to do this efficiently, no matter what I tried.

If the "tags" field in the original data contained a specific tag, I would set that specific tag's field to 1. This resulted in a dataset containing more than 176,000 entries, each having 193 binary features. I then mapped this to several non-binary "metadata" type fields that would help me in the future.

Each entry had an adversary ID, a date of creation, a description, and a type field. The type field had the data classified into 23 types, consisting of technical terms such as "cat 7 event". I standardized these into "recon" and "attack" categories based on what they were.

After I solved problem 1, I took the resulting class definitions and stored them in the last column of the original CSV. I then processed this CSV containing classified datapoints into a new dataset containing a

mapping of how recently an adversary performed recon to a specific type of attack. Theoretically, this would allow me to see if there is any time relationship between recon and certain attacks (per adversary).

Methods and Models

1. K-means clustering
2. Naïve Bayes
3. Support Vector Classifier
4. Decision Tree
5. Logistic Regression
6. MLPClassifier
7. Linear Regression
8. KNN Regression

Problem 1 Results

For this assignment, it took many iterations to figure out the best way to do this. I tried clustering and all kinds of classification. Below are descriptions of each attempt.

Clustering

My original idea was to use K-means clustering for the first component of this assignment. This was because I did not originally think that I could derive strict classes from the dataset. By standardizing the 23 original "classes" into "attack" and "recon", this issue was solved.

Upon importing the data and using a KMeans object with 2 clusters, I trained the object using the first 193 columns in my dataset (all features), then printed out the resulting clusters. Here are my results for K-Means clustering:

Clusters (result of k-means)
{1: 133560, 0: 42896}
Ground truth
{1: 106279, 0: 70177}

This shows that close to 30,000 items were mislabeled. As we can see, this did not turn out as expected, so I switched back to classification.

Naïve Bayes Classification

I then tried Naïve Bayes classification on the dataset. This approach was far more successful, and the algorithm was able to correctly classify around 95% of the data. Because Naïve Bayes relies on well-chosen features to be successful, I had guessed that this would work the best for this problem. I found out that I was wrong, and perhaps my feature selection out of the tags wasn't perfect. I performed 5-fold cross validation on this, and noted these results:

Naïve Bayes

	Precision	Recall	F1	Accuracy
Mean	1.0	0.932394134019	0.965011530941	0.959275979622
StDev	0.0	0.003246855740	0.0017390331179	0.002067620802

Support Vector Classifier

I tried out a support vector classifier on this data, with exceptional results. I had never used this model before, and it turned out to work very well. However, it took me more than 20 minutes to run (approximately 5 times as long as others). This is likely because although SVCs perform well with large numbers of features, that performance drops off as the number of datapoints increases. Regardless, the SVC demonstrated near-perfect numbers when classifying my data. I performed 5-fold cross validation on this, and noted these results:

Support Vector Classifier				
	Precision	Recall	F1	Accuracy
Mean	0.999953004151	1.0	0.999976501302	0.999971664324
StDev	2.97656324e-05	0.0	1.48835171e-05	1.79211564e-05

Decision Tree

For this dataset, the decision tree model provided the best numbers. This is likely because the dataset is perfect for a decision tree. There are many datapoints and there is a hard delineation between the two classes based on the tagging system on AlienVault OTX. I performed 5-fold cross validation on this, and noted these results:

Decision Tree				
	Precision	Recall	F1	Accuracy
Mean	0.999924772843	1.0	0.999962383322	0.999954662822
StDev	8.20678987e-05	0.0	4.10392343e-05	4.94052740e-05

Logistic Regression

Logistic regression provided great results and was extremely efficient. The results were nearly perfect, in fact. This is surprising, because Logistic regression is really overkill for this application. I have purely binary features, no categorical ones. In addition, logistic regression does not perform well when there are large numbers of features. This model also uses the entirety of the data that is fed into it, so the performance speed was impressive and probably lucky.

Residual sum of squares: 0.00
Variance score: 1.00

I performed 5-fold cross validation on this, and noted these results:

Logistic Regression				
	Precision	Recall	F1	Accuracy
Mean	0.999953004151	1.0	0.999976501302	0.999971664324
StDev	2.97656324e-05	0.0	1.48835171e-05	1.79211564e-05

The residual sum of squares indicates a perfect fit of the model to the data. This likely has to do with the hard relationship between tags and types in the original dataset.

MLP Classification

MLP classification provided near-perfect numbers, but average performance. This model used up 100% CPU for several minutes as it ran. Although the CPU was at 100% for all 8 cores, it still took the same amount of time to complete as some of the single-threaded models. This means that in this case, the overhead of multi-threading and aggregating the results is not worth it.

For the following results, I ran the MLP classifier using the *relu* activation function, and two layers with 20 nodes each. I tried several variations of this, but there were no discernable changes. There was no need to use StandardScaler as the features were all binary.

MLP Classification			
Precision	Recall	F1	Accuracy
0.999990590892	1.0	0.999995295424	0.999994332865

Problem 2 Results

After problem 1 was solved, I was able to take the results and add them to the original dataset as "solutions" to the problem.

I then processed the classified CSV into a new dataset containing a mapping of how recently an adversary performed recon to one of 5 types of attack. For each increment of 30 days from the time of an attack, I mapped all instances of recon to a feature.

Linear Regression

Because I had non-binary classes, I needed to use linear regression. Unfortunately, what I found was discouraging. I ran a linear regression model against 50 different adversaries' data and none of them had any good results. Here is an example after training the model with 80% of the data and testing on 20%:

```
Residual sum of squares: 2.23
Variance score: -1.32
```

After dropping my test size down to 10%, I saw no real change in results

This was disappointing to say the least. I was hoping to use the coefficients from this model to establish which timeframe was the "best" indicator for which attack type. These results were very likely impacted by the sample size, but it is also probable that there simply is no correlation between reconnaissance timeframe and types attack events.

KNN Regression

When I ran the KNN regression model against the same data, I always got slightly better results, but there is still no indication that there is a correlation within the data as I had hoped. Here is an example of a KNN regression run against the same data after training with 80% of the data and testing on 20%:

```
Residual sum of squares: 4.53
Variance score: -0.08
```

Conclusion

All tests indicate that there is no way to correlate reconnaissance events with attack events using this data. Unfortunately, this means that this operation has been unsuccessful. Personally, I really believed this would work so I am very disappointed. It seems that either malicious cyber actors are random in their behavior, or we just do not have enough attributed data on what they are doing.

Although this experiment was unsuccessful, I do not plan to stop trying to find use for this data. I now have the environment set up to easily continue experimentation with this and other data.