Statistics of DNS Data in CTU-13

Chris Goswick
Network Analytics
University of Colorado Boulder
Boulder, Colorado
christopher.goswick@colorado.edu

***Abstract*—My research is exploring the DNS data for traffic captured by the University of Colorado on August 8th, 2022. My initial analysis includes loading the data, data cleaning, and understanding the shape and simple metrics of the data. I then replicated the natural language processing aspects of the article ”DGA-based Botnet Detection Using DNS Traffic Mining” to detect anomalous data discovered from K-Means clustering algorithms. The anomalies were detected by discovering anomalies that were two standard deviations away from the cluster centroid. Then a decision tree model was built to predict the labeled anomalies. Once the tree was built, the results show that data and features created were not beneficial in this application to use for botnet detection.**

I. Introduction

For the class project in CYBR 5320 I have chosen to do an anomaly exploration of DNS data that was captured from on campus traffic from August 8, 2022. The purpose of the exploration is to understand what normal DNS traffic looks like and to look for anomalies that can lead to the detection of threats. I chose to replicate certain aspects of the paper ”DGA-based botnet detection using DNS traffic mining.” This paper focused on bot detection using the query that was sent to DNS servers for transcription into their IP addresses. This paper provides an alternative look at detecting bots using some natural language processing techniques.

II. Background

A botnet is a network of remotely controlled machines, like computers, cameras, sensors, servers, etc., that are hijacked to be used in malicious attacks such as DDOS, phishing, and information theft. These networks are usually controlled by command and control servers (C&C). One common technique used by botnets is the creation of domain names, also known as domain generation algorithms (DGAs). These algorithms generate large numbers of unique domain names that the botnet can use to communicate with its C&C servers. DGA's are designed to create domain names that are difficult to predict and therefore difficult to block. This makes it hard for law enforcement officials to track down the origin of the botnet. These domain names can vary wildly depending on the algorithm that created them.

One technique, in cases of phishing, is to create domains that mimic popular websites or companies that can trick people into visiting these fake websites to steal sensitive data. Other algorithms can be abnormally short to long and contain special characters in an effort to be able to change them quickly to avoid detection. These bots are programmed to communicate with their C&C server to send DNS queries to be resolved until a live IP address is found. Their ability to dynamically change and quickly query new addresses makes it difficult for these bots to be detected and stopped.

”DGA-based botnet detection” suggests that these DGA queries differ from known benign sites enough that malicious activity can be predicted on the query alone. Their theory is that normal, benign domains are meant to be human-readable. Things such as 'amazon.com' or 'bankofamerica.com' are meant to derive a certain meaning and be easy to remember. DGA botnet domains generated by algorithms are usually of the same length and contain meaningless words, misspelled words, numbers and special characters that can lead to their detection. [5]

III. Related Works

Several studies have focused on the detection of botnets using DNS traffic analysis. In a recent study by Mansrah et al. (2022), the authors propose a machine learning-based approach to detect botnets using DNS traffic data. Their approach involves clustering DNS query data using the K-Means algorithm, and using decision trees to detect anomalies in the resulting clusters.

The authors claim that their approach achieves high accuracy in detecting both known and unknown botnets, and outperforms existing botnet detection methods. However, their study does not provide a detailed comparison with other approaches, and it is unclear how their approach would perform on larger and more diverse datasets.

Overall, the study by Mansrah et al. presents a promising approach to botnet detection using DNS traffic analysis, and highlights the potential of machine learning techniques in this field. However, further research is needed to evaluate the effectiveness and scalability of their approach, and to explore other potential avenues for improving botnet detection using DNS traffic analysis. [5]

IV. Data

A. Data Access

The data is being provided by the University of Colorado Boulder and being accessed through the class server. Per an agreement with CU, the data has to stay on the server so that access to the data can be controlled and proprietary information is not shared. Analysis will be done on the file '2022-08-13_dns.09:00:00-10:00:00.log'. The notebook used for analysis can be found at data/dns/Goswick_project/dns_project.ipynb

B. Data Tools

I will be using a Jupyter notebook run on a virtual environment through the class server to do my analysis. I will be using the pandas package to read in the data to a data frame and perform filtering and aggregations. I will also be using the matplotlib and seaborn packages for data visualizations. Other packages like, numpy and datetime for data cleaning procedures. Further I will be using several packages from Scikit Learn to pre-process the data and run the machine learning models. The Natural Language Took Kit (NLTK) will be utilized for feature engineering to compare words with real words and misspellings. Regex was also utilized for cleaning purposes.

C. Data Features

The original data includes 24 columns. Using the open source community Zeek.org, I will use their information to learn more about which columns could be the most useful for analysis. The following features taken from the class website should be the most helpful and will provide as a basis for initial exploration: [1]

- **id.orig_h**: the ip address of the host in either ipv4 or ipv6 format
- **id.orig_p**: the port number that the host is connected to
- **id.resp_h**: the ip address of the DNS server responds from
- **id.resp_p**: the port number that the DNS server responds from
- **trans_id**: a 16-bit identifier assigned by the program that generated the DNS query. Also used in responses to match replies to queries.
- **query**: the domain name that is being requested
- **qtype_name**: descriptive name for the type of query
- **answers**: set of resource descriptions in the query answer

V. Initial Analysis

In the data there were 5,903,036 records. There are 70,548 unique host ip addresses and 19,905 unique response ip addresses. It would be expected that there are more hosts than response since a lot of the DNS requests would be sent to a common DNS server. It can be inferred then that address 128.138.213.13 would be the main DNS server for CU since it was sent 1,884,139 of the almost 6 million requests. It was sent 1,505,073 more requests than the next most used address of 128.138.240.1 which is on the same network.

Since most DNS requests are fulfilled on port 53 I thought that it would interesting to see how many were found to use a different port. I found that there were 2006 requests that were sent to other ports with over half of them going to port 137. According to www.speedguide.net, port 137 is used by NetBIOS naming service which is a file and print sharing protocol and also Windows Internet Naming Service (WINS) which is similar to DNS for NetBIOS systems [2] [3]. This is a legacy system meaning that the devices using this protocol are probably old and could be a point of interest for further analysis. The overwhelming majority of DNS requests are fulfilled using UDP.

In this instance of data there were 5,815,094 resolutions through UDP and 87,942 resolutions through TCP. The most common reason why TCP would be used in a request is when a transfer from a primary to a secondary DNS server is needed. The TCP ensures that data is being shared consistently and accurately. There is not enough evidence that TCP protocols should be explored further. [4]

VI. Data Cleaning

Of the 5,903,036 rows in the data set, 3,222,777 contained instances of blank queries, queries that contained a single hyphen, or recursive DNS requests that were removed. The blanks and hyphens are believed to be the reply's from the DNS server back to the host. Once those were removed, periods, hyphens, and underscores were removed from the query to separate words and phrases with a space for easier analysis. Then the top level domains were stripped from the queries as well. This left several thousand instances of queries that contained single letters. A reg-ex filter was applied to remove single and double letters which aren't useful in processing words. There is a large majority of domain names that contain compound words such as 'stackoverflow.com' An attempt was made to separate these words to be analyzed separately but a lack of compute power prevented this from being successfully cleaned. The new cleaned data was stored in the column 'clean_query'.

A. Feature Engineering

In 'DGA-based Botnet Detection', they created several feature engineered columns to assist with the analysis. I was able to replicate most of the same features for my analysis. 'domain_length' stores the total number of characters in the raw query including punctuation. 'num_words' is the total number of words to be found within the query. 'num_hypens', 'num_periods', and 'num_numeric_tokens' all contain a count for the total number of designated characters contained from the raw query. 'rare_words_ratio' is a count of words that are rare to this instance of data by word counts. 'common_words_ratio' is the opposite of rare words where the common words appear more than once in this instance. 'misspelled_words_ratio' is the ratio of words that were misspelled when compared to the 'wordnet' dictionary found in NLTK. If the word was found in the dictionary, then word was considered spelled correctly, incorrectly if not found. From there, I created a binary column for each of the main TLD's found in the US; .com, .org, .edu, .gov, .net, and .biz. Lastly, I compiled an entropy score for the raw query to get a randomness score based on what was typed into the search bar.

VII. Finding Anomalies

In 'DGA-based Botnet Detection', they created their own instances of malicious data by using known algorithms of discovered DGA botnets. They combined this data with real DNS query captures to create a labeled data set that could be used directly with machine learning models. In my case, I first needed to use the features that I created to discover anomalies to test for. I did this by utilizing a K-Means clustering algorithm to cluster all the data points and then finding the ones that were further than 2 standard deviations away from their cluster centriod's. Anything that was further than 2 standard deviations away was labeled as anomalous. I first found the optimal number of clusters using the elbow method, which was 6 as shown in figure 1. Once the optimum number of clusters was found, I re-ran K-means with the optimal parameters. Once clustered there were 2654 anomalies detected. Now that the anomalies were detected I could then build a machine learning model to try and predict the anomalies.

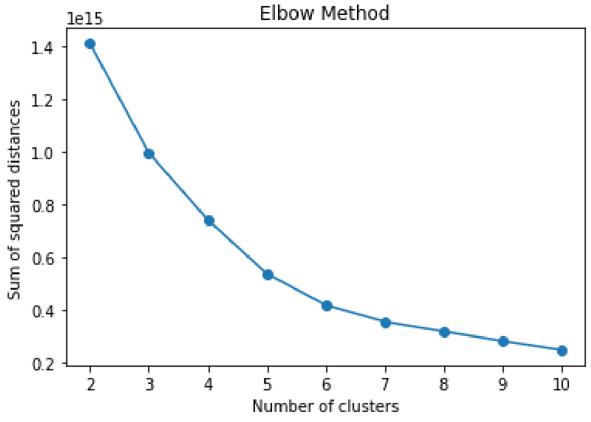

Fig. 1. Feature Importance

VIII. Machine Learning

I decided to use a decision tree algorithm to train my model for several reasons. Decision trees are known to be very accurate model due to its ability to split decisions based on certain criteria. It is also easy to extract feature importance to see what is useful in making predictions. This is highly valuable since this project is exploratory in nature. I can see what factors led to my models scores which allows me to get rid of columns that aren't useful.

During the preprocessing of the data, I split the data into training and test sets with a 80/20 ratio and balanced with the positive instances being stratified. I then used StandardScaler from Scikit Learn to standardize the data to be on the same scale. This way the model doesn't suffer from biases or outliers.

Finally once the data was prepared I could train the decision tree model on the training data. I used the hyper parameter 'class_weights' with values of $\{0:1, 1:50\}$ to account for the imbalance of positive instances. Once the data was train I used the test set to make predictions to see how well the model performed.

IX. Results

The decision tree model did not yield the results that I had hoped. Despite having an accuracy of 99%, precision was less than 1%. The model seems to suffer from imbalanced data. After looking at the confusion matrix in figure 1, it is apparent that the model is poor at predicting the positive class.

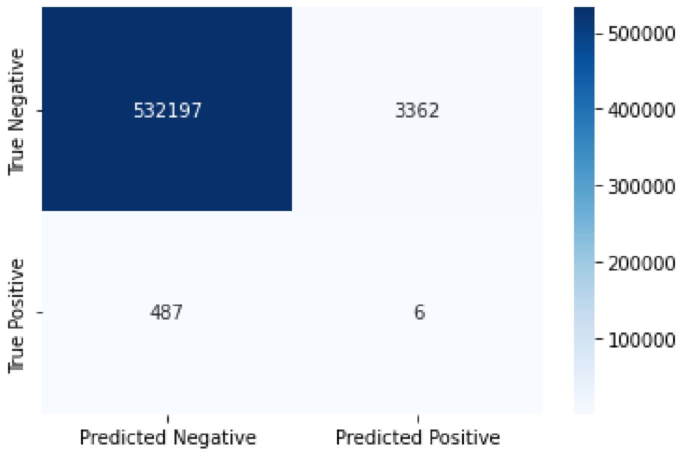

Fig. 2. Confusion Matrix

After looking at the feature importance of the columns, shown in figure 2, I have determined that some of the columns that I featured engineered are not as helpful as I would have hoped. The top 5 features were the source ip address, the destination ip address, clean query, entropy score, and the domain length. The other features contribute very little to the classification.

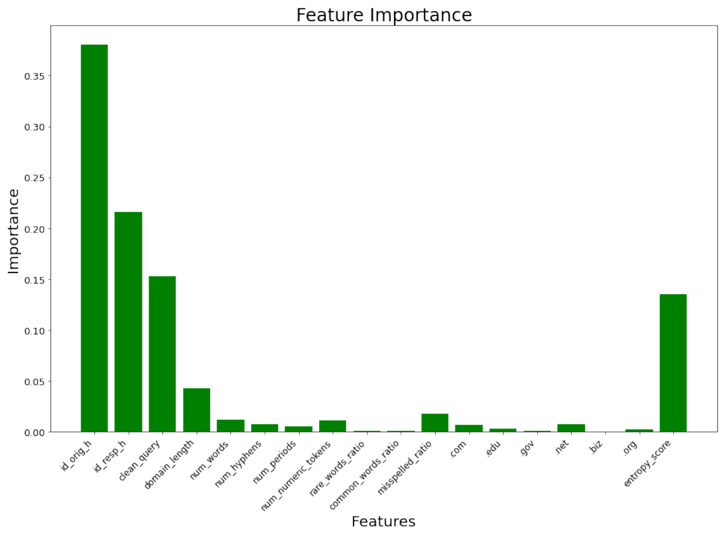

Fig. 3. Feature Importance

I believe this is due to a few factors. First, is the class imbalance. With only 2464 anomalies, there just are not enough of the positive class to learn from. I do believe that since CU is a large institution, it has many protections in place to detect bots and keep them out. There is no way for certain to know if any of the anomalies are actually bots. This is leading to the class imbalance. Second, is the quality of the feature engineering. Some of the features like rare_words_ratio and misspelled_words_ratio suffer from some the queries complexity. Because of the nature of some of the queries coming from a institution of higher education, the queries are not the usual type of query you would see in a normal use case. For example, 'utcnist2.colorado.edu' is a popular domain that was queried 27,898 times. It is a national time server which explains why it was queried so many times in one hour. The 'utcnist2' is not a word that would appear in a dictionary of words. So the misspelled_words_ratio is counting that as a misspelled word even though it passes the benign test. Therefore the scores for this column are not accurate. I believe that on normal data that would be captured in a normal setting, these issues would not be a problem. Lastly, is the issue of the use of compound words in domain names. I did not have the access to as much computational power as they did in the paper. The attempts that I made at splitting compound words did not find success as it took too long to run to be effective so I scrapped the feature. Therefore, the features of common_words_ratio also took a hit in quality because of the inability to split these compound words.

X. Future Considerations

Having learned a lot about the botnet detection process through implementation there are several things that I would change to improve the performance. First would to be find an efficient way to split the compound words to be able to accurately account for them. I believe this is a main factor contributing the poor results of the model. Once that was resolved I could find a better data set to test on. One with labeled data so I could be certain to train the model on actual cases of bots. Lastly, I would try to implement the typing difficulty score that the DGA paper created. This was their number one feature in determining bots by 0.2 over the second place common_words_ratio by information gain.

References

[1] Base/protocols/DNS/main.zeek. base/protocols/dns/main.zeek - Book of Zeek (v5.2.0). (2023, March 1). Retrieved March 7, 2023, from https://docs.zeek.org/en/current/scripts/base/protocols/dns/main.zeek.htmltype-DNS::Info

[2] SpeedGuide. (n.d.). Port 137 (TCP/UDP). SpeedGuide. Retrieved March 7, 2023, from https://www.speedguide.net/port.php?port=137

[3] Rashmi Bhardwaj. (2022, September 14). When does DNS use TCP or UDP? " . Network Interview. Retrieved March 7, 2023, from https://networkinterview.com/when-does-dns-use-tcp-or-udp/

[4] DNSSEC – what is it and why is it important? ICANN. (n.d.). Retrieved March 7, 2023, from https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-05-en

[5] Manasrah, A. M., Khdour, T., amp; Freehat, R. (2022). DGA-based botnets detection using DNS traffic mining. DGA-Based Botnets Detection Using DNS Traffic Mining, 34(5), 2045–2061. https://doi.org/10.1016/j.jksuci.2022.03.001