Online Traffic Anomaly Exploration and Models

Fiskewold, M., Mikkola, F., Nakae, J., Schwaiger, J., Thorne, J., Wyman, L.

April 23, 2023

I. Introduction

Intrusion Incorporated is a cyber security company that specializes in advanced threat intelligence, malicious traffic identification, and automated threat response. When it comes to all of the domains that exist on the world wide web, two of the key questions this company seeks to answer about such domains are, “What does normal activity look like” and “What are the indicators of malicious intent”. Our project focused on providing insight to these questions by utilizing machine learning, and other techniques.

Over the past several years, Intrusion noticed a trend in anomalous web traffic rates for multiple web domains. The domains can be viewed as a collection of floors in a skyscraper and the amount of foot traffic each floor experiences is akin to the web traffic rate. Over time, patterns of foot traffic have been extracted to determine what ‘normal activity’ looks like for a floor at any given time. The traffic is constantly monitored and there have been multiple events of floors receiving a lot more foot traffic than what is ‘normal’, and then traffic suddenly returns to normal within a few days. Such events are anomalous and these are the events our team investigated.

The goal of our project was to use information about web traffic flow for all domains to help classify and predict potential anomalies using machine learning, and other techniques. The challenge comes from anomalies being so scarce relative to the amount of domains and overall traffic that need to be monitored. It is analogous to the skyscraper being millions of floors tall and one entity is tasked with monitoring all floors simultaneously while trying to find anomalous traffic. Another challenge is that not all anomalies are malicious. For example, a domain related to a celebrity may experience an unusual amount of traffic when that celebrity passes away.

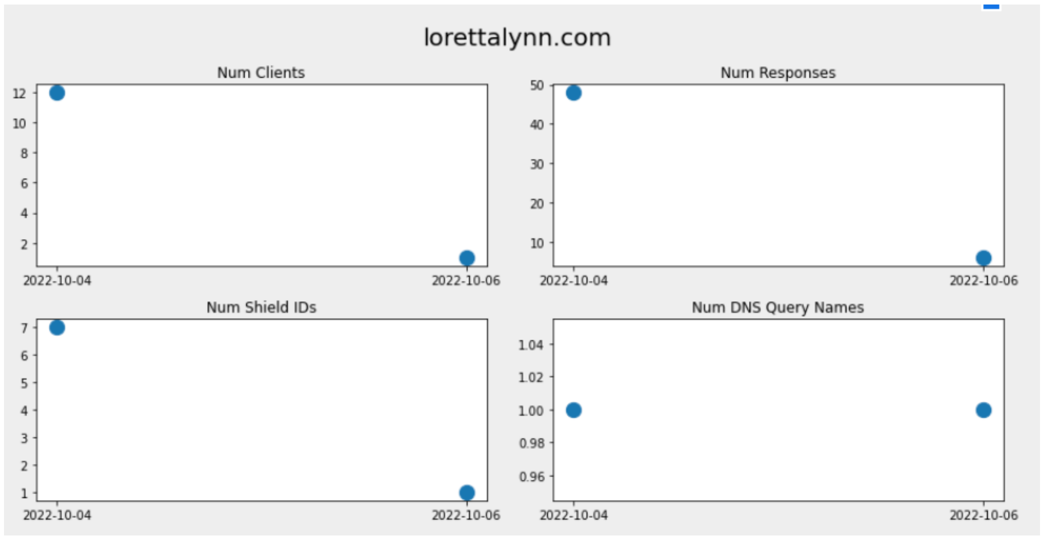
II. Data

The data we recieved represented an aggregated summary of DNS logs per site, per day. This data was collected between the months of October 2022 and January 2023. The original data frame consisted of eight features; day, shieldID, dns_qname, dns_qdomainname, num_clients, num_responses, timefirstseen, and timelastseen. ShieldIDs are unique site codes for each sensor deployment. Dns_qname and dns_qdomainname are the FQDN requested of the DNS resolver, and the registered domain of the dns_qname (eg. www.intrusion.com and intrusion.com). Num_clients are the number of distinct IPs that requested resolution of that dns_qname on the given day, at that ShieldID. Num_responses is a sum, across all IPs, of the DNS answers returned, for that dns_qname, on the given day, at that ShieldID. Timefirstseen and timelastseen are first and last seen timestamps that the dns_qname was requested, on that day, at the shieldID. This data contained 89 unique shieldIDs, 466686 unique domains. We later received some additional data that contained statistics on when and with who the domain was created. This data contained the domain name, the whois host, date created, date updated, and date the domain expires.

III. Anomalies

a. Method 1

When first starting this project we were faced with the obvious questions of "what is an anomaly?" With only a couple of provided examples and a vast dataset, we needed a way to get a list of domains we knew were anomalies. One thing we knew that was a sure anomaly indicator was if the domain name had appeared on no Shield-IDs and then all the sudden appeared on multiple at once. To isolate these in the dataset, we need to group by both the domain name, day, and shield–id. Then we need to count how many days each domain name occurred in the dataset. This was important because if the domain name appeared only a few times throughout the dataset, it was more likely to be an anomaly than a domain that appeared almost everyday. Then, we wanted to make sure that it occurred on at least 3 different shield-id's. This technique was highly effective in identifying a list of 200 anomalies. One interesting one that it found was "lorretalynn.com". Below is an example plot of each of the categories (Num Clients, Num Responses, Num Shield IDs, and Num DNS query Names).

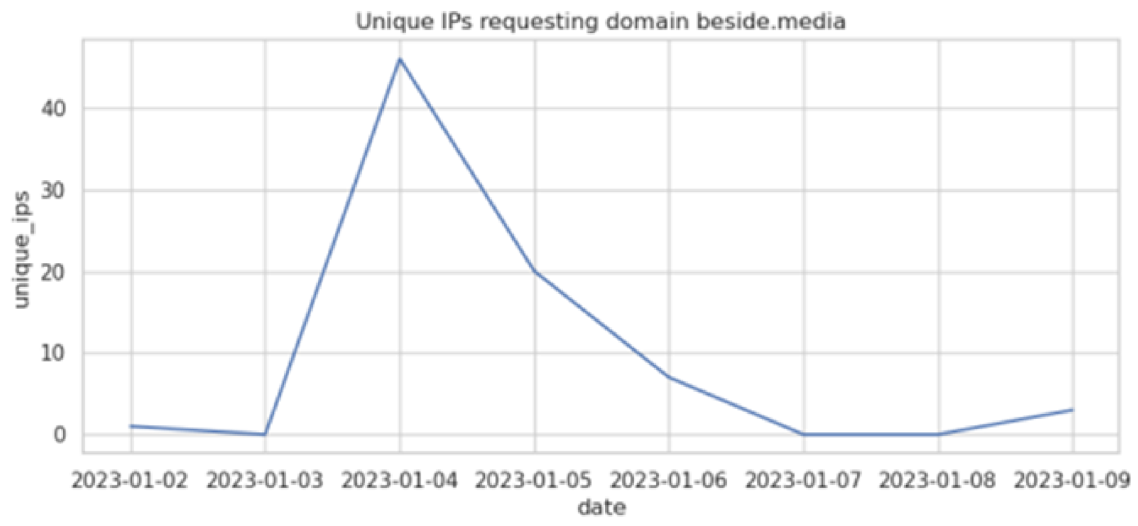


As we can see, there is a "random" spike on Num Shield Ids on 2022-10-04. After looking this up, we find out that this turns out to be the exact day that Loretta Lynn passed away. So this anomaly is not necessarily a bad anomaly, but still is an anomaly.

There were a couple of flaws to this approach. First, anomalies are not necessarily just a domain name, but rather a domain name and day paired together. Second, this method would not pick up domain names that were frequently occurring in the data set but had specific days that were anomalies. This method also did not account for any other type of anomalies than those defined by shield-id. As we will see anomalies can be classified using a variety of different types.

b. Method 2

The second type of anomalies that were researched came from examining the shape of the example anomy given in 'beside.media'. When the dates and traffic for this website were graphed, it formed a very distinct pattern which was quite different from most other websites. The traffic would be very low for a long period of time, and then shoot up suddenly to its peak amount of traffic. Then the traffic would slowly die back down to essentially no traffic.

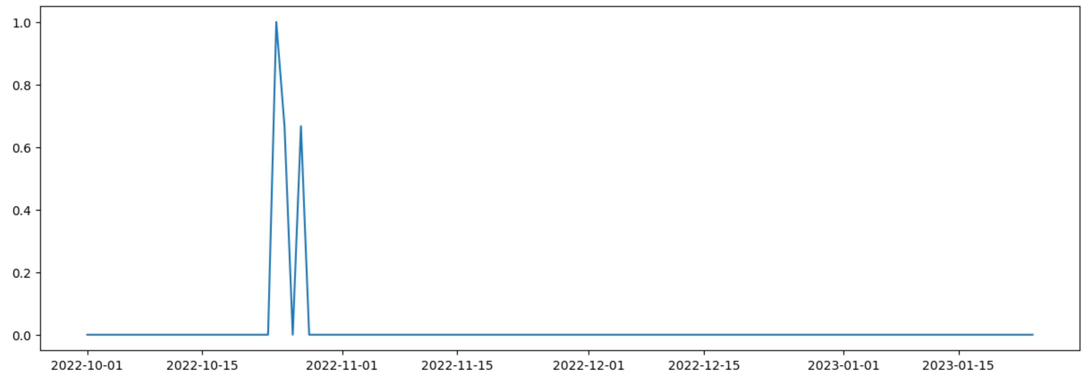


Intrusion believed that this site was involved in unwanted data collection, and thus was the perfect example to base my work on. The first thing we did was run a linear regression program using this data as the base to find other sites like it. It was successful in the way that it found other sites with relatively similar traffic values, but it only found sites that had traffic at the exact same time as ‘beside.media’. This was a huge problem as it would only find anomalies that were anomalous on the same dates. The solution to this came from importing a python library called tslearn, which had a specific function called DTW or dynamic time warping. This dynamic time warping would compare two graphs and then return a value based on how close those graphs match up, even when they are on different x axes. The lower the value, the more similar the graphs. With this we were able to compare websites with traffic that occurred on completely different days. We ran the DTW function comparing all websites with beside.media and only saved those who had a DTW value below a specific number that we chose. We chose a DTW value of 0.6 based on the number of websites we wanted to return. Anything less than .6 and too few websites would be returned, and too many websites that had minor differences were excluded. Anything more than 0.6 seemed to return too many websites that only shared minor similarities. From a sample of about 600 thousand unique domain names only around 1200 were below .6. Around 100 thousand were below 0.7.

One of the websites that this process found that stuck out was called ‘123w0w.com’. Intrusion notified us that this particular website had reports of shady behavior involved with phishing for sensitive information. Upon further examination this website had an interesting characteristic that we would use to find further anomalies. It turns out this website was created just a couple days before it had its big spike in traffic that is common with these types of anomalies. Intrusion had already provided us with a new data set that had the creation dates for all the websites within the data. We wrote a function that found the peak traffic date and then compared the two dates to find how many days the 2 dates were separated by. From the domains that were found using the DTW function, we could now see if any of those domains were created within X amount of days from their peak traffic date. So now we could find all domains that resembled an anomalous website in shape, and also see if they are anomalous in their quick success to maximize traffic within days of being created. Here a few of the websites that were found using these methods:

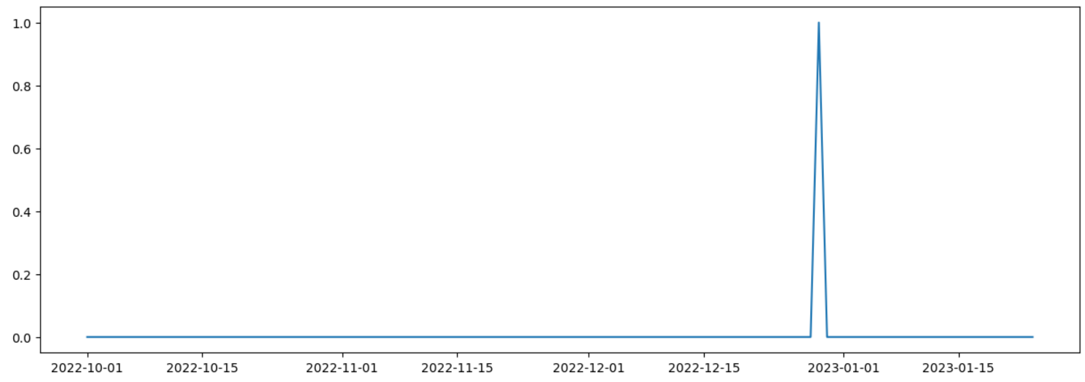
‘lowhardboiledadjoin.com’

Created: '2022-10-21'



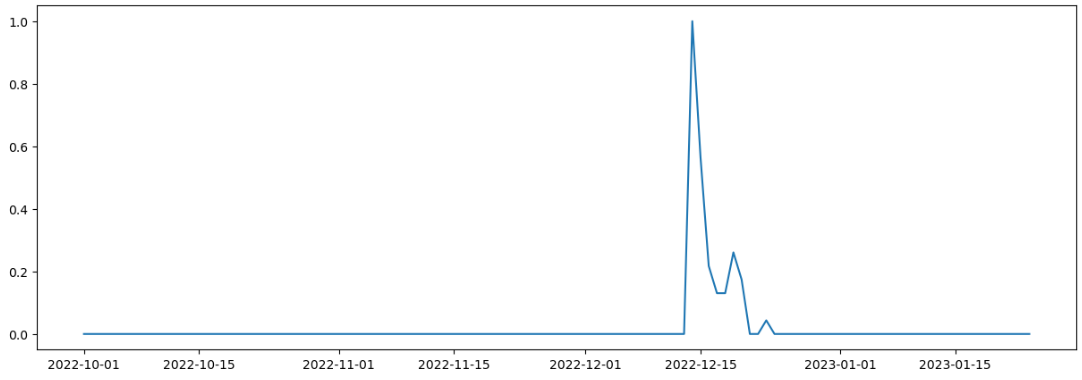
‘Grand-prize-add2.live’

Created: 2022-12-24



‘123w0w.com’

Created: 2022-12-12



The websites that are found using these methods have been classified as type 2 anomalies. We believe it is a great start in the right direction for identifying websites that are looking to illegally phish for data. The limitations that hold these methods back are the fact that not all phishing websites resemble ‘beside.media’ and not all phishing websites are created relatively close to their peak online traffic date. However, we think this is a good start, and with further parameters and improvements these methods could become much more effective with time.

IV. Models

a. Classification

Once we were able to identify domains that were likely anomalies we began attempting to predict these domains using various different ML classification models. Our initial model attempts included logistic regression, random forest, and XGboost. Unfortunately these initial attempts at classification proved very poor due to such an unbalanced dataset. This sparsity was to be expected as the occurrence of anomalies is rare in relation to normal web traffic. However, since so few potential domains were identified using both methods, our initial models could simply classify everything as not an anomaly and still have accuracy metrics of above 90%. To combat this, we combined techniques of undersampling and oversampling. We took the data and oversampled the classified anomalies and undersampled the non-anomalous rows. We arbitrarily chose a 1:1 correspondence of anomalies to non-anomalies for a total of around 750,000 rows.

We then split this data into a training and testing set. After running our models, we realized that our models will be used to predict anomalies on unbalanced data sets, but our testing sets were balanced. This caused issues in determining the effectiveness of the models. We then repeated our process but in a different order. The data was first split into training and testing sets. The training set was then balanced, again by obtaining a 1:1 correspondence of anomalies to non-anomalies with a total of around 500,000 rows.

While the features of the initial data provided valuable information we chose to add more features to give our models more data to enhance classification. Using 'www.domcop.com/top-10-million-domains' to identify the top million most popular domains we created a column that identified whether or not the given domain existed in the top million and where it ranked. If it ranked in the top ten thousand, we removed it from the data. Our assumption was that malicious or anomalous domains would likely not be ranked in this list, and those ranked above ten thousand would be well known domains. Using the additional whois data we created month, date, and year columns based on the dates the domain was created and expired. Additionally we created a time existed column which was the total number of days between created and existed. After researching similarities between malicious domains we learned creators often register their domains using certain tld's (spamhaus.org). We therefore created a column isolating the tld of each dns_qdomainname.

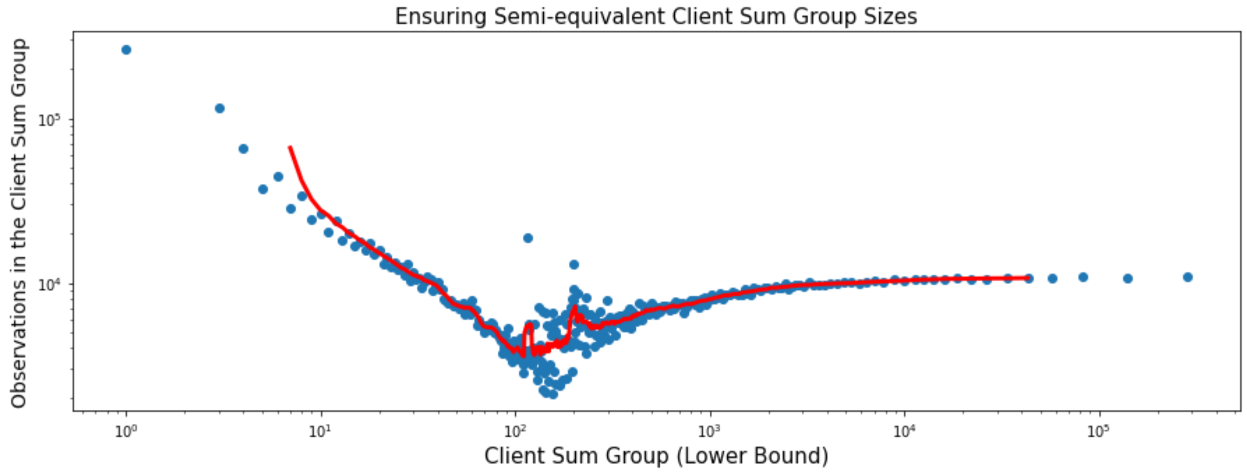
Results

Once we had more robust and balanced data we once again tried our random forest and XGBoost models. We trained our models with the balanced data and then tested on the unbalanced, as well as a single day of the data. We used the Optuna library in python for hyperparameter optimization as well as AUC metrics instead of accuracy. Unfortunately our random forest model still performed poorly assigning probabilities of approximately 0.5 to each domain. However, the XGBoost model performed much better, classifying 100% of the assumed anomalies correctly, with a false positive rate of 2% on unbalanced test data. Using the data from a single day returned the same result, this time with a false positive rate of 1%. It should be noted that the data misclassified as false positives are not necessarily misclassified. It is possible our method of identifying anomalies in the first place is not completely accurate and therefore the model may be finding more anomalies than originally identified.

b. Statistics-Based Model

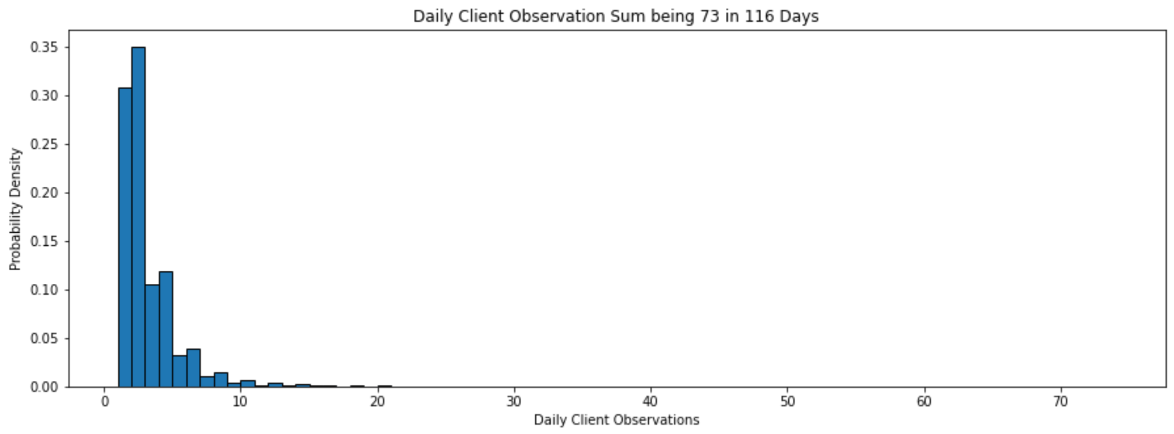
The second type of model that we explored investigates finding anomalous instances in sets of domain names and days rather than just domains. Some domains exhibit normal behavior but may have a spike in one or two days that could be considered anomalous. To identify these instances, we create a statistical model for the number of clients, number of responses, and number of shield ids for each domain on each day. By comparing domains with similar levels of total traffic across sensors, we can analyze how sensor traffic is spread out in comparative domains. We build a normality score that shows how likely we are to see an observation as large as the one seen in the data for a sensor sum group. Then, we make suggestions for taking these normality scores and applying a second means of classification to reinforce our first method of anomaly score assignment.

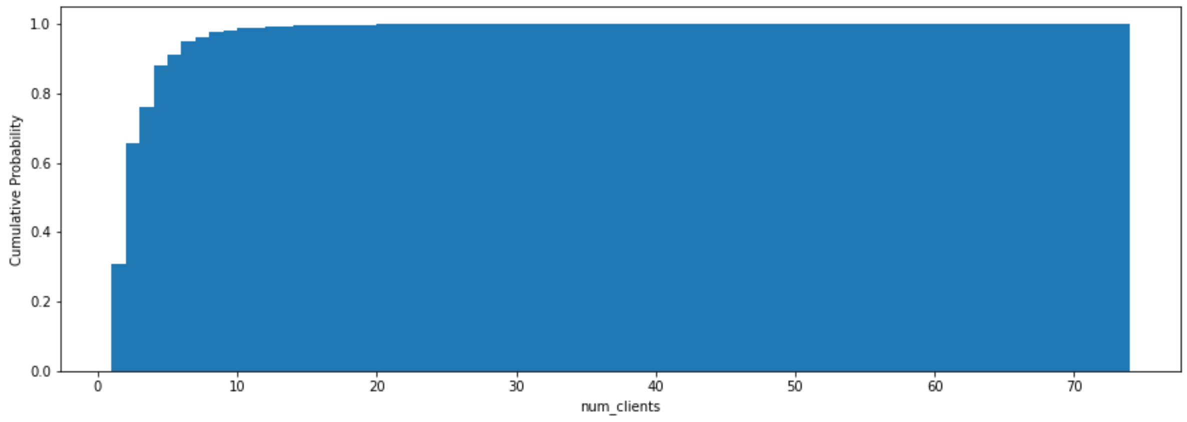
To motivate the model, we explain the methods by using the daily number of clients columns presented in the dataframe. Take each domain's total of daily clients over the data's timeframe for each domain name. We initially assign each domain into a client sum group, indicated by its sum over the data's timeframe. In doing so, we notice disparities in the size of the sum groups as the sums get larger. This makes intuitive sense, as we would expect to have many domains with small period sums and few domains with large period sums. To avoid this issue, we attempt to break the client sum groups up into ranges that accurately represent the spread of the data.



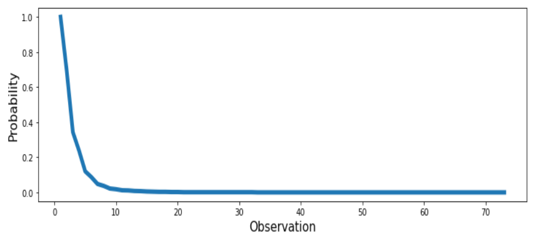
As shown in the figure above, we have come up with a strong method of splitting domains into different sum groups. We expect more observations in client sum groups that are smaller. The number of observations in a client sum group never drops below 1000 which indicates that each sum group has adequate samples to explore empirical distributions of how the number of clients is being summed over the period. It's important to note here that when sum groups get larger, the ranges in which the sum groups cover also gets larger. This helps in keeping sum groups of equivalent sizes and thus, avoid chaos and randomness in the total daily number of clients over the time period, e.g. a sum of 1238 is essentially the same as 1240.

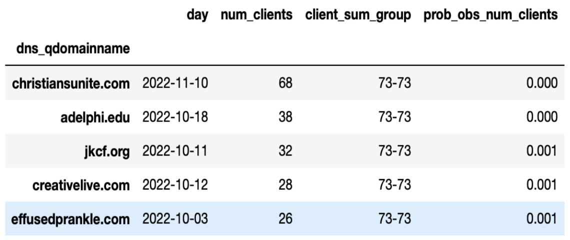
To further motivate the model, we explore the client sum group which covers all domains that have the number of clients column sum to 73. The figure below shows the empirical distribution of daily clients when the sum of daily clients over the period for a domain was 73. As we can see in general, as the daily client observations increase, the probability density decreases.





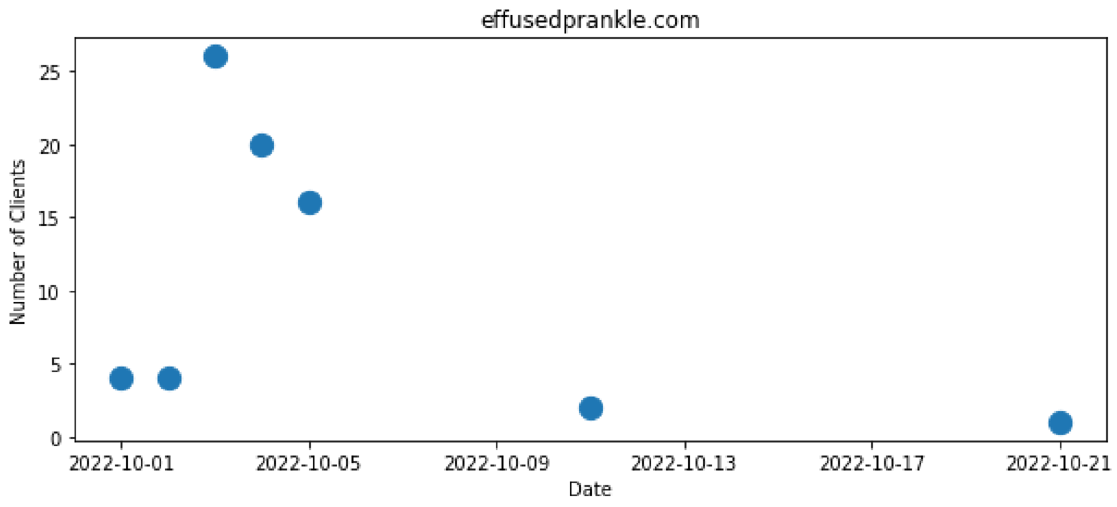
Taking a look at the cumulative probability density function, we see that observations above approximately 10 seem highly unlikely when the sum is 73. Therefore, to capture this information, we assign each observation with the probability of seeing an observation as large as the one seen, or one minus its cumulative probability. So, the probability values will be inversely related to the daily number of clients. Let's take a look at 'effusedprankle.com.'



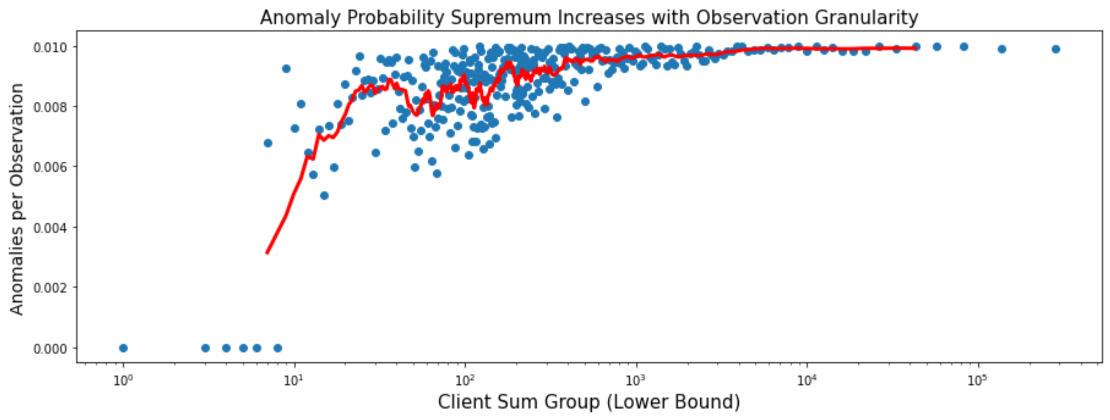


	day	num_clients	client_sum_group	prob_obs_num_clients
dns_qdomainname				
christiansunite.com	2022-11-10	68	73-73	0.000
adelphi.edu	2022-10-18	38	73-73	0.000
jkcf.org	2022-10-11	32	73-73	0.001
creativelive.com	2022-10-12	28	73-73	0.001
effusedprankle.com	2022-10-03	26	73-73	0.001

The domain 'effusedprankle.com' has a large spike at the start of October, but has little-to-no sensor traffic shown in the rest of the data. This is a malicious anomaly, of which we are satisfied with labeling as such with a low probability score.



Now, we take and apply the same process for each client sum group and examine the observations with probabilities less than .01. This implies that in a given sum group, the probability of observing a value as large as the one that was seen is less than 1%. While this anomaly probability cutoff is arbitrary, we also see that doing so avoids labeling any instances in client sum groups strictly less than 8. As our client sum group increases, we note more anomalies per observation, which speaks to the improved level of accuracy as probability granularity improves as observations are more differentiable, e.g. the probability loss going from a 5 to a 6 daily client observations in a small client sum group will be much larger than the probability loss going from a 190 to a 191 daily client observations in a large sum group.



Now that we have established a means of differentiating anomalies in the number of clients, we extend the same logic to the number of responses and number of shield ids. We follow the same checks for sum group sizes to keep an effort towards statistical accuracy in the model. After doing so, we normalize over total daily traffic to avoid oversampling and seeing too many anomalies from specific days with high levels of sensor traffic with which to begin. While we find normalizing important, the model does not necessarily require it if the data was balanced from different sources. In our data, we had shields that only reported in October with high levels of traffic, but then dropped out of the data, leaving the data to seem unbalanced. One way to avoid this would be staying consistent in the shield ids used over the time period.

Now that we have established a way to identify anomalies based on the number of clients columns, we perform the same process on the number of responses and the number of shield ids. Now, we would like to take each of these probability columns and create an interaction function between them that tells us more about how anomalous these observations really are. We would like observations that have low probabilities for each column. Since probabilities were assigned on different sum groups, we need to rescale each column to avoid oversampling from one column versus another column. To do so, we sort the data by probabilities and then replace the lowest with 0 and the highest with 1, evenly spacing each normalized probability. Then, to create the interaction, we take the product of each probability column raised to the one half power:

$$\sqrt{\mathrm{ClientProb}_i} \cdot \sqrt{\mathrm{ResponseProb}_i} \cdot \sqrt{\mathrm{ShieldIdProb}_i} = \mathrm{NormalityScore}$$

Now, we also take the interaction function, sort by the values, and rank the lowest with 0 and the highest with 1, evenly spacing each interaction probability. While ranking seems like an added unnecessary step, we are actually interested in the sensor observations' relationships with each other rather than the actual values. By ranking from 0 to 1, we have a way to easily identify the percentiles of the probabilities rather than the strict probabilities. Doing so addresses any imbalances in what we are claiming to be a probability from column to column.

When investigating the outcomes of this model, specifically the instances with the lowest normality scores, we note moderate accuracy. After thorough investigation of network behavior of the top 100 identified by the model, we note at least 10 that are marked as malicious with a quick internet search, e.g. 'so1cool.com', 'healthy-tracker.com', 'mc-lci1.com' and 'msh-cdap-tat.com'. Additionally, we note many domains involved in sharing news articles, which we speculate short-lived popularity due to feature articles, e.g. 'romper.com', 'mgtv.com', and 'thewalrus.ca'. Here are a random sample of twenty domains from the top 1000 observations with low normality scores:

Random sample of Domains with Low Normality Scores
aaxdetect.com, self.com, adxyield.com, videowalldirect.com, yale.edu, luxebook.in, loreal.net, planespotters.net, bestreviews.com, z2data.com, fidelity-media.com, avocet.io, cengage.com, hdsex.org, biggreenegg.com, 17173.com, nymag.com, spokenlayer.com, joinhandshake.com, bangwo8.com

This provides a great start in labeling a training dataset that strictly identifies malicious versus non-malicious observations. We could then use that training dataset to investigate the accuracy of new models.

Discussion/Future Projects

When we first began classifying anomalies, the classification was based on only a handful of characteristics. Defining more characteristics to the data became our first big hurdle in improving our models. Data regarding the creation of domains and the registration period of the data helped to add more characteristics. We also looked at common malicious top-level-domains (ie .com, .org, .net, etc.) and added that information into the data. For future investigations of this data with new project teams, having this information already included may be of use. Also, for the first couple weeks, we lacked an example of an anomaly making our search rather difficult. Lastly, we did not anticipate that unbalanced data would pose a problem due to lack of experience. Investigating methods of techniques to handle unbalanced data took time away from investigating the models themselves.

Overall, the biggest challenges we faced were the unbalanced data, the sheer size of the data, and marking desirable ‘malicious’ anomalies. In terms of balancing the data, a lot of the choices of how much to undersample and oversample were arbitrary. It is not known if a 1:1 correspondence of anomalies to non-anomalies is desirable as well as having a training set of about 500,000 rows. The size of the data forced us to look at subsets due to limited computing power. Most analysis used the subsets to perform model training and it is not known how our models work full scale. Lastly, we found many examples of ‘anomalies’ that were not malicious in nature. As mentioned, the ‘lorrettalynn’ anomaly appeared due to her passing and resulted in high web traffic on her website. Finding a way to determine what predicted anomalies are malicious and which aren’t is another area for future investigation.

In terms of models, there are still lots of techniques that we did not have the time or ability to implement. Isolation forests were used, but the results were very broad whereas PCA and tSNE methods resulted in overfocused results. Finding a way to combine these methods together to achieve balance may lead to interesting findings. There are also more complex methods that were recently created such as Robust Subspace Recovery Autoencoder Layer (RSR Neural Network). The complexity of the new methods require more time to understand the theory and mechanics of them. But, if time can be used to learn these, they may also lead to better models for anomaly detection.

Lastly, with our statistical approach to defining anomalies, we are curious how it can be incorporated into a machine learning approach. Our first idea is to use the generated probabilities of an anomaly to become another characteristic for machine learning to use to better predict anomalies. Another route would be to use the probabilities as the method to classify the anomalies themselves. The machine learning models can then find patterns in the data that determine those probabilities we may not be able to detect. One last route would be to turn the classification problem into a regression problem. The model could be trained to predict the probabilities themselves leading to predicted probability scores on new data.

In conclusion, we were tasked with answering the questions of “What does normal activity look like” and “What are the indicators of malicious intent”. Our process began by understanding the data set provided and gaining a general sense of the problem. We then began searching for anomalies, appending other characteristics to the data, and employing other methods of machine learning. Lastly, we developed a statistics based approach and then refined the models developed. Our vision for future projects include tuning hyperparameters for the various models, finding other ways to balance data, and employing new machine learning methods.