Network Attack Clustering

CURE Algorithm Evaluation

Srini Bondalapati (sb1686), Patrick Hickey (ph402), Chaitanya Bachhav (csb152)

***Abstract*—We aim to analyze malicious network traffic captured and identified by Kitsune, a modern NIDS. We want to evaluate how well the CURE clustering algorithm can differentiate the different attack types identified by the Kitsune NIDS.**

I. Project Description

A. Overview

Network security has become increasingly concerning for many organizations. In response to this, neural networks have become a popular heuristic method for detecting network attacks. One such example of this is Kitsune, a Network Intrusion Detection System (NIDS) (5). This system gathers various statistics about packets and labels them as benign or malicious based on their attributes. We would like to evaluate how well the CURE clustering algorithm can differentiate the different attack types identified by the Kitsune NIDS. The data set obtained from the Kitsune NIDS will serve as the baseline. This will allow future researchers and network security administrators to better quantify the benefit of more sophisticated methods of network intrusion detection.

B. Tools and Technologies

- Python: The coding for this project was done in project.
- Vaex: Vaex is a python library for Out-of-Core DataFrames to visualize and explore big tabular datasets.
- PyClustering[4]: The PyClustering library was used to implement most of the CURE algorithm.
- Plotly: Plotly was used to generate the plots which assist in visualizing the data and results

II. Objective

Our objective is to be able to answer the following questions based on our data:

- How well does CURE differentiate different network attack types identified by the ANNs used in the Kitsune NIDS based on packet statistics?
- How well does the CURE algorithm differentiate benign from malicious traffic?
- What impact does the number of clusters have on the ability to differentiate between network attack types?

III. Data Description

Our data is sourced from the UCI Machine Learning Repository. It consists of 27170754 records and 115 attributes at 16 bytes per attribute (long double), for a total of roughly 55.8 GB. Each record is a collection of 23 statistics about network traffic captured during a network attack related to the packets' size, count, and jitter aggregated by various combinations of source MAC, source IP, channel, and socket. These statistics are taken over five different time periods, for a total of 115 attributes. The dataset is broken down into 9 different csv files according to distinct attack types, each of which is mixed with benign packet captures. There are also 9 csv files with Boolean values specifying whether a packet is benign or malicious. The eight of the nine attack types were evaluated, as one attack profile summary had an additional attribute that was not explained. The total size of the data that was evaluated was 20,253,452 records, each having 115 attributes at 16 bytes per attribute for a total size of 37 GB. Roughly 80 percent of this data was benign traffic, as can be seen in Figure 2. Additionally, a subset of the data containing only the packets that were labeled as malicious was evaluated separately. This set contained 4,208,758 records with 115 attributes each, for a total size of 7.5 GB. Some attack types had significantly more data than others. The total amount of data for each attack type can be seen in Figure 1.

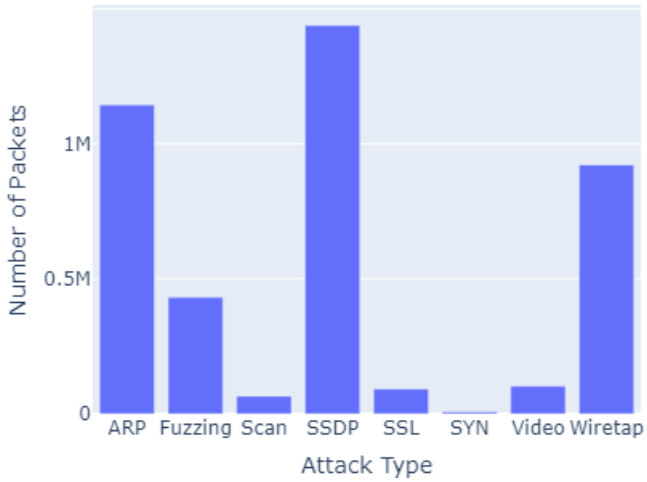

Fig. 1. Attack Type Data Distribution

IV. Modes of processing

Due to the large size of our dataset, we chose the Clustering Using Representatives (CURE) algorithm for clustering. The rough steps of the CURE algorithm can be seen below:

- Make one pass over full data set, sampling an amount that can fit in main memory
- Implement hierarchical clustering on the sampled data. Each cluster is representative by a particular amount of representative points. These points are then condensed towards the center of the cluster by a certain factor.

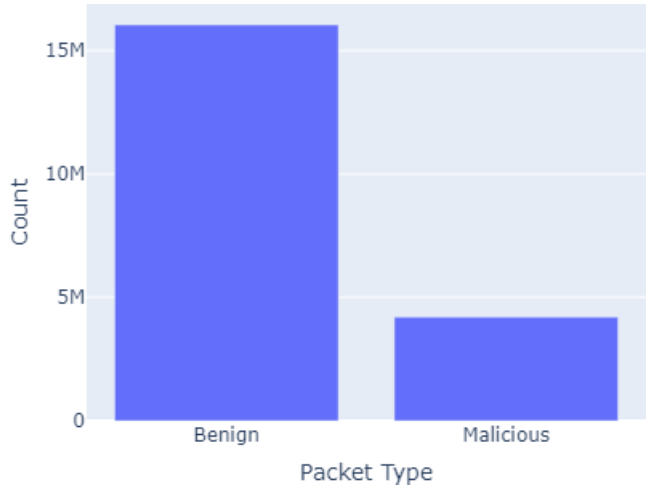

Fig. 2. Packet Types

- Make a second pass over the full data set, evaluating the distance of each datapoint to each cluster representative, and assigning the datapoint to the cluster with the closest representative.

For our data sample, we took a collection of 3000 data points from each attack type, for a total of 24,000 data points. This data in each column was then normalized so that one column wouldn't dominate the others when comparing distances in Euclidean space. This data was then clustered using two different methods. The first method was to predefine the desired amount of clusters based on the number of data types. This led to k=9 clusters for the dataset that included both the malicious and benign traffic (eight different attack types plus benign traffic) and k=8 for the malicious only traffic (eight different attack types). The second method was to cluster the sample data into 100 clusters and then observe the amount of meaningful clusters that evolved. The representatives from the clusters with a meaningful amount of data (generally greater than 500 datapoints out of the 24,000 sample size) were then used to group the full dataset on the second run through of the data. A condensing factor of 0.2 was used for the cluster representatives in both scenarios. Preliminary testing changing the condensing factor to different values yielded negligible results. Ten representative points were used to define each cluster. In all scenarios, after the clusters and cluster representatives were identified, we scanned the full datasets and counted how many packets of each attack type were assigned to each cluster for our analysis.

V. Analysis and Evaluation

Our first attempt at clustering looked at a sample of the combined dataset (both benign and malicious traffic) using a cluster size of k=9. We then created a matrix with the results, displaying how many data points of each data type were assigned to each cluster. This matrix can be seen in Figure 3.

The majority of the data (58 percent) was all grouped in the same cluster. To get a better idea of how well each attack type was clustered, we then created a Jaccard similarity matrix

	ARP	Fuzzing	Scan	SSDP	SSL	SYN	Video	Wiretap	Benign
0	50036.0	4061.0	64939.0	1429574.0	7576.0	1403.0	510.0	47748.0	1295589.0
1	0.0	0.0	0.0	0.0	66.0	0.0	0.0	44.0	329.0
2	1094176.0	421399.0	759.0	16.0	60180.0	4445.0	101578.0	874487.0	9097611.0
3	0.0	0.0	1.0	0.0	488.0	0.0	0.0	3.0	216.0
4	0.0	6285.0	0.0	9964.0	6663.0	663.0	0.0	17.0	3199548.0
5	29.0	17.0	0.0	6.0	684.0	0.0	111.0	28.0	2089.0
6	0.0	0.0	1.0	0.0	0.0	240.0	0.0	0.0	63.0
7	1030.0	486.0	0.0	19.0	7.0	0.0	299.0	887.0	7758.0
8	0.0	535.0	0.0	24.0	16987.0	286.0	0.0	1.0	2441491.0

Fig. 3. k=9 Full Set Data Matrix

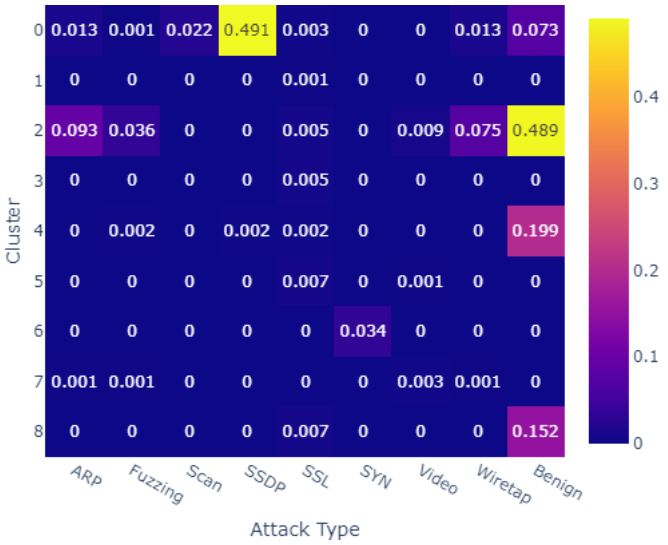

Fig. 4. k=9 Full Set Jaccard Similarity

to show how closely related each attach type was to each cluster. We then highlighted the highest similarity score with any cluster for each attack type. The Jaccard similarity matrix can be seen in Figure 4.

The Jaccard similarities were low for all attack types and clusters. The highest two were the SSDP dataset (49 percent) and benign data (49 percent).

We then looked at the combined sample dataset using a cluster size of k=100 for the sample data, selecting only the clusters with significant density for use with the full dataset. This coincidentally led to nine clusters, but with different representative points. The data matrix can be seen in Figure 5.

Here, once again, the majority of the data (62 percent) was assigned to the same cluster. The Jaccard similarity matrix can be seen in Figure 6.

The change in results were negligible for the most part. The similarity of the benign data increased slightly to 54 percent, and the similarity of the SYN DoS attack type increased from 3 percent to 10 percent. The algorithm was not able to effectively differentiate between any attack types when benign data was included.

	ARP	Fuzzing	Scan	SSDP	SSL	SYN	Video	Wiretap	Benign
0	1093234.0	427615.0	327.0	7927.0	25405.0	1543.0	101443.0	873243.0	9986549.0
1	395.0	0.0	0.0	0.0	0.0	0.0	135.0	1050.0	6664.0
2	802.0	359.0	0.0	1277.0	6626.0	663.0	299.0	609.0	3202040.0
3	251.0	205.0	0.0	516.0	367.0	0.0	104.0	305.0	2331.0
4	49787.0	4041.0	64727.0	1429883.0	7867.0	726.0	515.0	47763.0	1296275.0
5	255.0	10.0	213.0	0.0	95.0	917.0	2.0	30.0	1542.0
6	547.0	42.0	0.0	0.0	0.0	0.0	0.0	6.0	988.0
7	0.0	0.0	433.0	0.0	51007.0	3188.0	0.0	209.0	1547713.0
8	0.0	511.0	0.0	0.0	1284.0	0.0	0.0	0.0	592.0

Fig. 5. k=100 (Filtered) Full Set Data Matrix

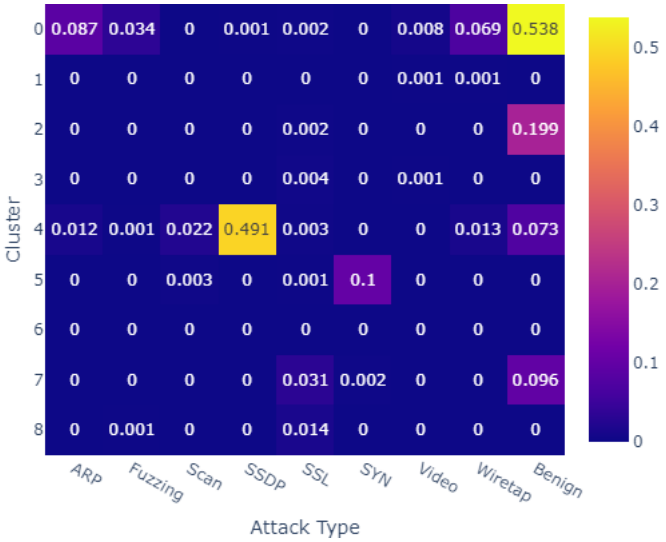

Fig. 6. k=100 (Filtered) Full Set Jaccard Similarity

Next, we went through the same process using only the malicious (non benign) data. After sampling the malicious data from all eight attack types, we ran the clustering algorithm at a set k=8 clusters (one for each attach type). The data matrix results can be seen in Figure 7.

In this scenario, nearly all of the data (99.9 percent) was grouped into two clusters, with the remaining data split

	ARP	Fuzzing	Scan	SSDP	SSL	SYN	Video	Wiretap
0	1143848.0	430457.0	1.0	0.0	88457.0	5311.0	102353.0	922009.0
1	0.0	559.0	3.0	0.0	0.0	0.0	0.0	0.0
2	9.0	1.0	8.0	0.0	15.0	490.0	2.0	7.0
3	241.0	1225.0	65686.0	1438692.0	4150.0	1236.0	12.0	101.0
4	1173.0	540.0	0.0	4.0	0.0	0.0	131.0	1097.0
5	0.0	0.0	0.0	907.0	12.0	0.0	0.0	0.0
6	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
7	0.0	0.0	0.0	0.0	17.0	0.0	0.0	0.0

Fig. 7. k=8 Malicious Data Matrix

Fig. 8. k=8 Malicious Data Jaccard Similarity

	ARP	Fuzzing	Scan	SSDP	SSL	SYN	Video	Wiretap
0	42.0	1190.0	74.0	3590.0	3299.0	1726.0	12.0	51.0
1	208.0	70.0	65621.0	1717.0	875.0	0.0	2.0	59.0
2	0.0	0.0	0.0	12943.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1421353.0	0.0	0.0	0.0	0.0
4	1143848.0	431013.0	3.0	0.0	88018.0	5311.0	99348.0	919689.0
5	1173.0	509.0	1.0	0.0	459.0	0.0	3136.0	3416.0

Fig. 9. k=100 (Filtered) Malicious Data Matrix

between the other six. Nearly all of the OS Scan and SSDP data was in one cluster, with the majority of the other attacks being in the other. This led to the Jaccard similarity matrix seen in Figure 8.

These results were not spectacular, but they were a significant improvement over attempting to cluster the dataset with benign traffic included. SSDP traffic had a 95 percent similarity with its assigned cluster. ARP, Wiretap, and Fuzzing traffic had a 43, 34, and 16 percent similarity rating using only malicious traffic, while none of them reached 5 percent while looking at the full dataset.

Lastly, we looked at the malicious data using k=100 clusters and choosing the clusters with significant density (greater than 500 data points). This resulted in 6 clusters being selected. The data matrix can be seen in Figure 9.

Here the majority of the data was still grouped in a single cluster (64 percent), however there was greater dispersion among some of the others. The next two highest clusters had 34 percent and 1.6 percent of the datapoints. The Jaccard matrix can be seen in Figure 10.

This method was able to accurately cluster another attack type, the OS Scan. Both SSDP and OS Scan attack types had a Jaccard similarity rating of over 95 percent with their

Fig. 10. k=100 (Filtered) Malicious Data Jaccard Similarity

assigned cluster. Impacts on the other attack types were mostly negligible.

In general, the CURE clustering algorithm was not able to differentiate between attack types very well, however evaluating only malicious data significantly increased the success rate. This makes sense, as roughly 80 percent of the full dataset was labeled as benign, so those datapoints were effectively acting as noise. OS Scan and SSDP Flood attacks seem to have the most distinctive statistical signatures, as they were the only attacks that the algorithm was able to isolate well. Interestingly, the total amount of data represented by an attack did not seem to have an impact on the ability of algorithm to correctly cluster them. SSDP data had more data than any other attack type, while the OS Scan data second had the second fewest. Other attack types seem to have more nuanced differences. Effectively choosing cluster sizes also had a significant impact on performance, as pruning a large selection allowed an additional attack type to be isolated in comparison setting a pre-defined amount amount clusters.

VI. Project Insights

Evaluating different subsets of the data using different clustering methodologies yielded several insights, both into the nature of the data and the implementation of CURE:

- OS scanning and SSDP flooding attacks are particularly distinctive and much easier to differentiate
- The presence of additional ”noise” (for this dataset, the benign traffic that included) significantly reduces the ability to identify different attack types
- It is difficult to identify malicious from benign traffic when the majority of traffic is benign
- Selection of cluster sizes and composition has a significant effect on the ability to accurately cluster data

VII. Future Work

- Evaluating Different Algorithm: Using a different clustering algorithm and comparing the performance to CURE could lead to insights on the strengths and weaknesses of each when looking at network traffic statistics.
- Refined Cluster Selection: More nuanced ways of selecting the amount of clusters and their composition might yield better results.

References

[1] https://github.com/ymirsky/kitsune-py.
[2] https://pyclustering.github.io/docs/0.8.2/html/index.html.
[3] https://www.youtube.com/watch?v=jrojspz1cuw.
[4] Guha, S., Rastogi, R., and Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. *ACM Sigmod record*, 27(2):73–84.
[5] Mirsky, Y., Doitshman, T., Elovici, Y., and Shabtai, A. (2018). Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*.