Anomaly Detection on DNS Traffic using Big Data and Machine Learning

Kelvin Soh Boon Kai
University of Glasgow
Singapore
2355381s@student.gla.ac.uk

Eugene Chong
Singtel
Singapore
eugene.chong@singtel.com

Vivek Balachandran
Singapore Institute of Technology
Singapore
vivek.b@SingaporeTech.edu.sg

***Abstract*—In this paper, we will demonstrate, devise and build an anomaly detection model for detecting general DNS anomalies in an unsupervised learning problem using multi-enterprise network traffic data collected from various organizations (NetFlow dataset) without attack labels. In our approach two types of clustering algorithms are implemented for evaluating the detection rate of the model. Clustering algorithm K-means and Gaussian Mixture Model (GMM) are investigated due to their popularity for being the state-of-the-art techniques for detecting anomalies with low false negatives [1]. In addition, another unsupervised neural network algorithm (SOM) is used for visualizing if any potential cluster can be found in the dataset. Simulation of DNS anomalies will be performed for evaluating the robustness of the final detection model, and a comparison has been made between K-means and GMM by assessing the detection rate against the simulated anomalies. The final GMM model achieved a seemingly high detection rate on the simulated anomalies.**

1. Introduction

With the advent of abundance of devices and tools in the market, collection of data has never been easier today. Corpus of network traffic data can be generated in millions or billions of records in seconds [2]. In this paper, the focus will be on these Big "Network Traffic" Data. By adopting and leveraging the current tools and off-the-shelf state-of-the-art algorithms, the objective is to to achieve cyber situational awareness thru data exploration and modelling, to devise and build a detection model for detecting network traffic anomalies with Big Data.

Cybersecurity is a major concern for companies and organizations that relies on technology to keep the business running. In any monetary intensive organization (Stocks and Banking), due to a glitch or vulnerability in the system, organizations could have lost millions or even billions. Thus, given the rise of new technologies and tools, to exploit and undermine the vulnerabilities of such systems can be achieved easily with adequate resources. There comes a point where new techniques and technologies are required to detect unusual and suspicious activities within the network, such that anomalous behavior can be promptly detected and mitigated.

With the prevalent of Big Data given the fact that it is exponentially growing every year due to the increasing technology available for generating these data e.g. IoT devices. Using Big Data, we are interested in applying Big Data Analytics on enormous network traffic dataset with the goals of generating insights and knowledge to infer and detect unusual and suspicious network behavior [3]. This paper is structured as follows. 2. Related Work. 3. Design, 4. Analysis. 5. Implementation. 6. Evaluation and 7. Conclusion.

2. Related Work

Past research has shown that traditional approach to anomaly detection uses Network "Intrusion Detection System" (IDS) to detect anomalies using known signatures. It is known that IDS is ineffective when trying to detect non-deterministic or unknown traffic, since IDS can only detect patterns or attacks from signatures, a new and unknown traffic might not yet be captured by the IDS. It is evident that detecting real time traffic has proven to be not feasible due to the non-deterministic nature of the varying traffic patterns in any enterprise network. Hence, alternative detection techniques should be implemented to handle the unknown nature of non-deterministic traffic patterns such as zero-day attacks [4].

Given the rise of Big Data, Machine Learning (ML), Deep Learning (DL) and Artificial Intelligence (AI), the strategies involving these techniques are evolving rapidly to help combat the limitations of traditional detection approaches with IDS. Given that these state-of-the-art techniques has proved to be perform better as compared to traditional IDS approach.

The following subsections A, B, C and D discusses the types of detection techniques approach to perform anomaly detection. [5].

A. Anomaly Detection

Anomaly detection is the process of detecting rare or unusual patterns that deviates from the normal behavior or norm. Unusual patterns are also known as "Outliers" or "Anomalies" [4], as shown in Fig.1. In the context of machine learning to anomaly detection, the focus will be building an anomaly detection model using ML clustering algorithms.

Fig. 1: The anomalies are the sudden spike.

Currently there exists two types of detection techniques to detect anomalies, 1. Anomaly detection, and 2. Misuse detection [4], [5], the research focus will be based on anomaly detection due to the caveat of misuse detection and the absence of attack labels.

B. Misuse Detection

Misuse detection or signature-based is an intrusion detection technique that build signatures of different types of known patterns from malicious behavior. It has been proven that misuse detection can detect malicious behavior with high detection rate due to the known patterns that are build and hard-coded as signatures by security experts. However, the downsides are they tend to perform badly due to unknown and unprecedented patterns such as zero-day attacks [4], [5]. The following Fig.2 shows the process of misuse detection using a rule-based pattern matching approach.

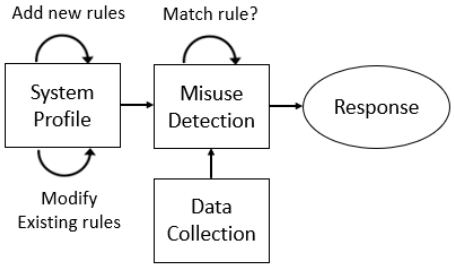


Fig. 2: Detection is only possible with existing rules and signatures.

C. Establishing a baseline of common patterns

Prior to any anomaly detection techniques described above, one of the statistical techniques for anomaly detection would be relying on the commonalities found in the data.

One being that the data should follow a normal distribution. Since anomalies don't occur on a regular basis, the assumption is that data should be normally distributed, as majority of the DNS traffic should be largely normal as compared to anomalies. By identifying the baseline of common behaviors/patterns, one can determine what are the common DNS traffic patterns flowing in the network [4], e.g. the average no. of pkts and bytes in a normal DNS flow. ***The term Flows, DNS traffic and DNS flows are used interchangeably in the subsequent sections.**

In addition, if anomalies are presented in the data, it needs to be removed, otherwise the detection model would have failed to detect future instances of anomalies as the detection would assumed it is normal during the detection [4].

Once the baseline of the common patterns has been established, detection can be deployed using the normal distribution as the baseline. Since anomalies are statistically different from normal DNS traffic, by modelling the normal distribution of the data, anomalies can be detected one, two or three standard deviations away from the mean using some threshold [4].

D. Clustering in Anomaly Detection

Clustering is a common technique used to group similar objects together for cluster analysis. Objects that are similar to each other belong to a cluster of it's own and vice versa for dissimilar objects, as shown in Fig.3. For anomaly detection in DNS traffic, similar DNS traffic patterns should belong to a cluster of its own, using some distance metrics such as the Euclidean distance. By following this rule, anomalies can be detected when new and incoming patterns stray away from any of the clusters or its distance to any of the clusters exceeds a certain threshold. It has been proven that clustering algorithms shows promises by learning distinctive and complex patterns from data without human intervention [6].

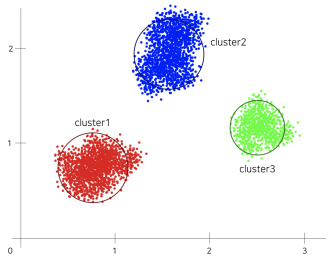


Fig. 3: Showing three blobs of clusters.

3. Design

This section gives a short overview of the tools and framework used for the proposed anomaly detection workflow.

A. Tools and Framework

The following are the computational and hardware specifications for this research which are provided by our industrial partner. Large scale data processing using Big Data Analytics are performed using the following big data cluster.

The hardware specifications for the big data cluster consists of 9 nodes with a total of 544 virtual cores and 3.5TiB of memory. The big data framework "Apache Spark" is used to

process the NetFlow datasets.

B. Proposed Anomaly Detection Workflow

The proposed anomaly detection workflow first comprises of **1) Data collection**, **2) Data analysis/preprocessing**, **3) Training using clustering algorithms**, **4) Deploying model for detection 5) Evaluation** and **6) Reiterate**.

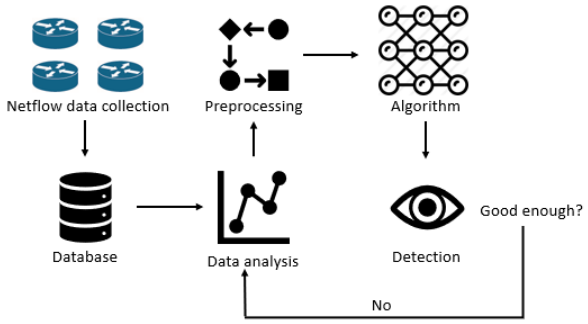


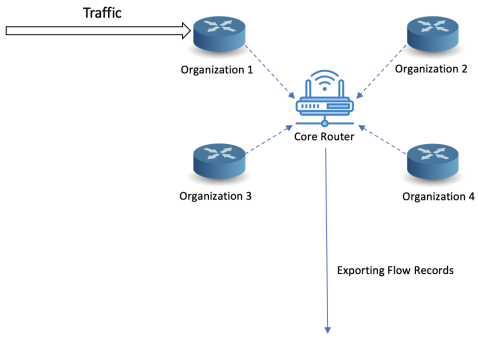
Fig. 4: Anomaly Detection Workflow.

4. Analysis

This section takes a deeper look into exploring the data. It gives an overview of the NetFlow dataset, and further analysis are explored to determine the characteristics that constitutes a normal DNS traffic by performing exploratory data analysis on the data.

A. NetFlow

NetFlow is a traffic monitoring protocol developed by Cisco for collecting network traffic flows from NetFlow-enabled router. Data collected using NetFlow can be used by network analyst to understand how network traffic is flowing in and out of the network [2].



$$Flow = (IP_{src}, IP_{dst}, Pkts, Bytes, T_{start}, T_{end}, Prot, Port_{src}, Port_{dst}, Flags)$$

Fig. 5: NetFlow data are collected from various multi-enterprise network.

The NetFlow datasets are jointly provided by our industrial partner **(Due to privacy reasons, the industrial partner would prefer to be anonymous)** which originally consists of 48 attributes with one day worth of network traffic data, but it was reduced to only 10 columns which are pre-selected as the most relevant attributes for data analysis of DNS traffic, as shown in Fig.6. We will be using one of the many NetFlow datasets stored in our database dated on "06/28/2018" which approximated around 253 million worth of records, but was reduced to 4 million records, since the objective of this research is to focus on DNS traffic to detect DNS anomalies, other services like HTTP, SSH etc. are removed. Each row or record in the dataset is known as a "Network Flow", each flow can also be referred as a transaction between the source and destination address [2].

A standard flow record F contains the following attributes.

$$F = (IP_{src}, IP_{dst}, Pkts, Bytes, T_{start}, T_{end}, Prot, Port_{src}, Port_{dst}, Flags)$$

src_ip	dst_ip	pkts	bytes	first	last	prot	sport	dport	flags
140.[redacted]	80[redacted]	1.0	79.0	1530120170	1530120170	17.0	33970.0	53.0	0
196.[redacted]	219[redacted]	1.0	1508.0	1530120170	1530120170	17.0	53.0	62437.0	0
139.[redacted]	231[redacted]	1.0	81.0	1530120170	1530120170	17.0	20586.0	53.0	0
229.[redacted]	41.[redacted]	1.0	77.0	1530120170	1530120170	17.0	58470.0	53.0	0
128.[redacted]	41.[redacted]	1.0	83.0	1530120170	1530120170	17.0	57347.0	53.0	0

only showing top 5 rows

Fig. 6: Sample of a DNS flow **(The IP address are masked for privacy reasons)**.

B. DNS flow packets count

TABLE I below shows packet feature where **pkts 1** and **pkts 2** are the most common number of packets send per DNS flow as compared to the next subsequent leading packets 3..5, hence 4.1 million counts of pkts 1 formed a commonalities of 95% of the total DNS flow.

	count (pkts)	
pkts	UDP	TCP
1	4147021	274610
2	82152	1177
3	22477	102
4	11147	95
5	6753	70

TABLE I: More than 95% of the total DNS flow are formed by only pkts 1 & pkts 2

C. Normality of Data

The above Fig.6 shows the 10 most relevant NetFlow attributes, the most useful features or attributes for determine the normality and distribution of the data are the no. of pkts per DNS flow, the **pkts** feature, as shown in TABLE I above, many of the UDP flow and TCP flow largely contains only 1 packet per DNS flow. Observations of UDP DNS flow with low packets count are much more common than larger ones.

The following two figures, Fig.7 and Fig.8 shows the typical distribution of the bytes feature commonalities in the NetFlow

dataset. There are two peaks in the distribution, one with 81 bytes another 1508 bytes Fig.7 (Red arrow), follow by a clearer zoom into a **single** well-known DNS Server e.g. "Google" that exhibits a bimodal (multi-modal) distribution with two peaks and also heavily right skewed, Fig.8. It can be concluded that most DNS flows contains only 1 packet with the average of around 81 bytes per DNS flow, Fig.7 and Fig.8 (Red arrow).

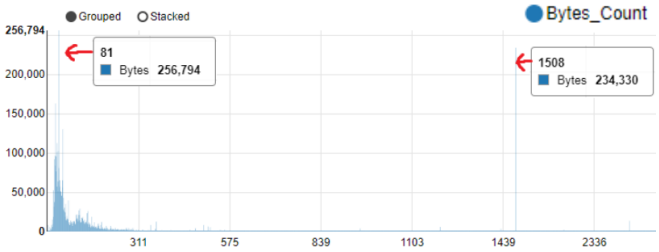


Fig. 7: Bimodal distribution of the feature bytes for all DNS servers.

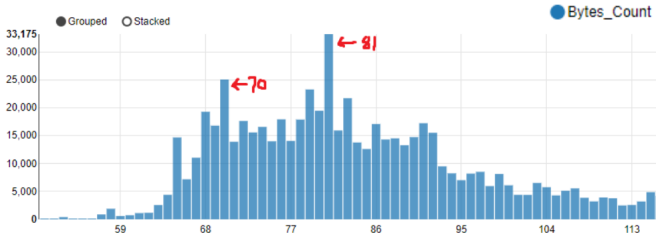


Fig. 8: Bimodal distribution of the feature bytes for one DNS server e.g. Google

Given the above TABLE I, DNS flow using UDP are much more common than DNS flow using TCP. Given the fact that UDP are short-lived, and the typical usage of DNS is to perform name-to-address translation by performing a DNS lookup using some DNS Server, thus the protocol must be fast to avoid any sort of latency, congestion and overhead in the network. The usage of using DNS via TCP is not as common as compared to UDP. Since TCP usually contains more data per flow due to the necessity of a reliable connection (three-way handshake) with additional information stored in a flow [7]. The typical usage of using DNS via TCP is Zone Transfer or sending large data over a network using DNS as a tunneling protocol where reliability is insured [7], [8]. However, the similarity of DNS flow using either UDP or TCP undoubtedly formed a commonality of only 1 packet per flow.

D. Determine important features

Relevant features of interest should be carefully hand-picked before fitting into any ML algorithms. Since irrelevant data, anomalies and noise should not exist in the data which will heavily penalize the quality of the final ML model during the actual detection [9], [10]. Given our initial analysis of the most common occurrence of pkts in TABLE I, pkts should be considered as one of the most crucial attribute for the detection of anomalies. During our initial observation of the normal baseline on DNS traffic, 95% of the flows contains only 1 packet per flow. This serves as an important information at determining whether a DNS traffic is anomalous or not.

E. Feature Selection

Selecting the relevant features is an indispensable process before data preprocessing, since the goal is to retain as much information as possible and remove any redundant information from the dataset that does not constitute towards the detection of anomalies. The feature **pkts** is one important feature, which helps in the detection of anomalies. It is evident from our initial analysis, DNS traffic with packets count between 1 and 5 made up of more than 95% of the data commonalities TABLE I; and the probability of any DNS packets per flow $P_r(2..N <= 1)$ is $<= 2\%$, where N is the packet, Fig.9.

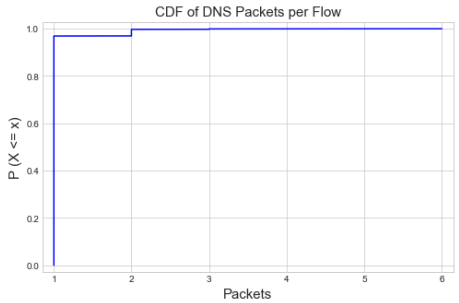


Fig. 9: Cumulative distribution function of DNS packets per flow.

The following Fig.10 shows the cumulative percentage of the average number of unique source and destination port in a typical DNS flow. Where 99% of the total DNS flow contains only at most 1 to 2 source or destination port. Hence, additional information such as using the features source port: **sport** and destination port: **dport** will be useful during the detection of anomalies.

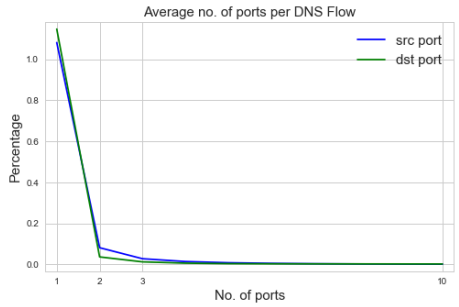


Fig. 10: Cumulative percentage of the average number of source and destination port.

F. Finalize Feature Selection

The following Fig.11 are the 10 features we originally had in the NetFlow dataset. These features will be used for data preprocessing and implementing the anomaly detection model.

src_ip	dst_ip	pkts	bytes	first	last	prot	sport	dport	flags
140.[redacted]	80.[redacted]	1.0	79.0	1530120170	1530120170	17.0	33970.0	53.0	0
196.[redacted]	219.[redacted]	1.0	1508.0	1530120170	1530120170	17.0	53.0	62437.0	0
139.[redacted]	231.[redacted]	1.0	81.0	1530120170	1530120170	17.0	20586.0	53.0	0
229.[redacted]	41.[redacted]	1.0	77.0	1530120170	1530120170	17.0	58470.0	53.0	0
128.[redacted]	41.[redacted]	1.0	83.0	1530120170	1530120170	17.0	57347.0	53.0	0

only showing top 5 rows

Fig. 11: Features selected for data preprocessing.

G. Types of Clustering Algorithm

Three clustering algorithms are investigated in this paper given their suitability towards our problem of interest. The following clustering algorithms will be further discuss in Section 5. Implementation. ***The list are by no means exhaustive, there could be a better clustering algorithm more suited for this research despite the following.**

- K-means clustering - Based on centroid models.
- Self Organizing Map (SOM) - Based on unsupervised neural network model with competitive learning.
- Gaussian Mixture Model (GMM) - Based on distribution and probabilistic models.

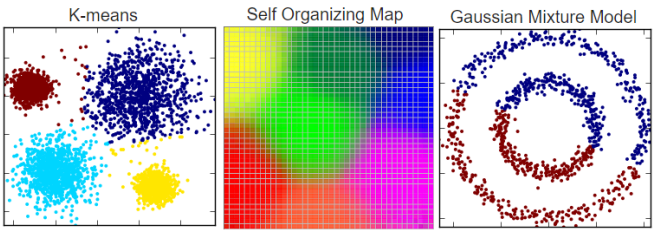


Fig. 12: Different types of clustering algorithms.

5. Implementation

This section discusses the features that will be used for implementing the anomaly detection model. It also discusses the preprocessing stage of the anomaly detection model, and finally the inherent limitation and caveat of the aforementioned three clustering algorithms. Both UDP and TCP DNS flow will be trained on clustering algorithms after data preprocessing.

A. Initial features

The following Fig.13 shows some features underlined in red, and were used to produce additional features through aggregation to derive a more sensible feature set. Since these are nominal/categorical values the ML algorithms don't understand.

src_ip	dst_ip	pkts	bytes	first	last	prot	sport	dport	flags
140.[redacted]	80.[redacted]	1.0	79.0	1530120170	1530120170	17.0	33970.0	53.0	0
196.[redacted]	219.[redacted]	1.0	1508.0	1530120170	1530120170	17.0	53.0	62437.0	0
139.[redacted]	231.[redacted]	1.0	81.0	1530120170	1530120170	17.0	20586.0	53.0	0
229.[redacted]	41.[redacted]	1.0	77.0	1530120170	1530120170	17.0	58470.0	53.0	0
128.[redacted]	41.[redacted]	1.0	83.0	1530120170	1530120170	17.0	57347.0	53.0	0

only showing top 5 rows

Fig. 13: Features selected for data preprocessing.

B. Preprocessed features (Time Window Aggregation)

The following Fig.14 shows the set of preprocessed features after time window aggregation using the features source ip: **src_ip**, datetime: **first** and **last** for aggregation in 1 minute interval. The reason for aggregating the flow in 1 minute interval is that a flow should not exceed X amount of packets when using DNS as a service [11]. By looking at the first row, the source IP "231.x.x.x" communicated with 133 unique destination IPs with 297 no. of packets with a total of 33.6k bytes, using only one unique source port to 294 unique destination ports with the protocol 17 (UDP) for transmission with a total of 295 flows in 1 minute. When inspecting the source IP of "231.x.x.x", it turns out that the source IP belong to a well-known legitimate DNS Server e.g. "Google", and is using one source port "53" to resolve DNS requests to 133 unique destination IPs in 1 minute, which is perfectly normal for a well known DNS server. After preprocessing the features, the following Fig.14 shows the aggregated flow in 1 minute interval.

src_ip	unique_dst_ip	no_pkts	no_bytes	no_sport	no_dport	prot	no_flow
231.[redacted]	133	297.0	33654.0	1	294	17.0	295
213.[redacted]	5	23.0	2938.0	20	5	17.0	23
41.[redacted]	606	1387.0	545268.0	1	1193	17.0	1326
13.5[redacted]	808	1774.0	127728.0	1604	1	17.0	1630
13.5[redacted]	583	864.0	62222.0	815	2	17.0	817

only showing top 5 rows

Fig. 14: Aggregated Flow in 1 minute interval.

C. Feature Scaling

Feature scaling consists of normalization/standardization which is to transform and bring features into the same units despite the different measurements. Normalization is to transform data into a specific range, bringing values into the range "[0,1]". Our preprocessed features are standardized and centered around mean 0 and standard deviation 1, modelling a normal distribution. Standardization can be perform using the following equation 1.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ is the mean, σ is the standard deviation and z is the z-score. The values of the rescaled features will be represented in z as continuous value.

D. Dimensionality Reduction

After feature scaling, projecting the data for visualization is difficult, since projection of data points in graph is only visually interpretable in at most two to three dimensions, anything beyond three dimensions are difficult for humans to visualize. Dimensionality reduction techniques are used such that our carefully preprocessed features can be best represented in at most two or three dimensions for easier exploration and visualization.

The preprocessed features after feature scaling has been transformed into six Principal Components (PC) using Principal Component Analysis (PCA). The following TABLE II shows that by applying PCA to our standardized preprocessed features, 95% of the total variance can be explained using only PC_1 and PC_2. Using PCA, the standardized preprocessed features are compressed into two for cluster analysis.

Component	Variance (%)	Cumulative (%)
1	78.53	78.53
2	17.05	95.58
3	2.4	97.98
4	1.1	99.08
5	0.76	99.84
6	0.16	100

TABLE II: Cumulative variance explained using six components.

E. Clustering Algorithms

The following three clustering algorithms 1) K-means, 2) Self-Organizing Maps (SOMs) and 3) Gaussian Mixture Model (GMM) are further discuss below.

1) K-means Clustering: K-means is one of the most popular unsupervised clustering algorithm used in ML. It is capable of handling large amount of data with $O(n)$ complexity, and often used for every preliminary cluster analysis priori. It is highly scalable with Big Data [11]. K-means algorithm works via an iterative approach, it takes in one important parameter called k, which is also the number of clusters you want the algorithm to return. The k is also known as the cluster centers or centroids [12]. Fig.15 shows K-means initialized with three clusters.

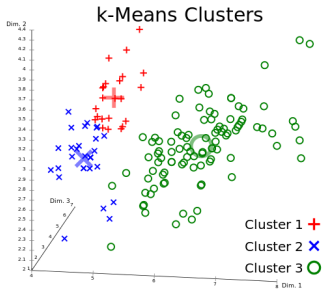


Fig. 15: K-means algorithm initialize with k=3.

The training of K-means algorithm are performed by specifying the k value to be 2, max iterations of 100, and a constant seed has been set for reproducible results. k = 2 is purely based on hypothetical assumptions and related work [11], [13]. With the absence of labels, we can only make general assumptions of the dataset, by using k = 2, we presumed 2 clusters, where cluster 0 denotes normal traffic and cluster 1 denotes anomalous traffic. Back to our original analysis of DNS flow, our CDF of DNS packets per flow contains only 1 packet 95% of the time Fig.9. The following Fig.16 shows two clusters plotted using PC_1 and PC_2 for visualization with an emerging **V** pattern, TABLE II.

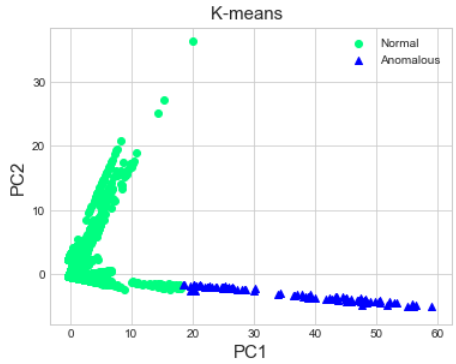


Fig. 16: K-means cluster analysis on PC_1 and $PC2$

2) Self-Organizing Maps (SOMs): Self-Organizing Map (SOM) is a type of artificial neural network that perform competitive learning known as vector quantization, unlike standard neural network architecture that uses error correction e.g. (Backpropagation via Gradient descent an optimization technique). It is also a clustering technique that maps high-dimensional features into low-dimensional for data visualization. It is similar to K-means, given that both are clustering algorithms. SOM is also a non-linear dimensional reduction technique as opposed to PCA (Linear dimensional reduction), suitable for learning complex non-linear patterns. SOM does not require a target label like many clustering algorithms. The following Fig.17 shows a map with grid X by Y, where the nodes X and Y denotes the neurons and $x_1, x_2..x_n$ denotes the input vector [14], [15].

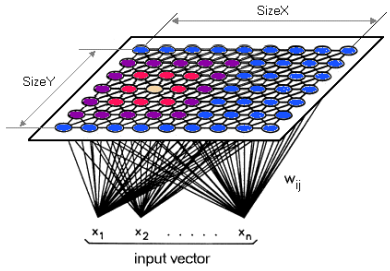


Fig. 17: Self-Organizing Maps (SOMs)

SOM can be used for interpreting high-dimensional data in at most 2D or 3D, usually accompanying with a unified

distance matrix (U-matrix) for SOM visualizations as a NxN hexagonal grid for identifying potential neighbours/clusters in large dataset without any prior k. Fig.18 shows the U-matrix of Iris dataset exhibiting around two to three clusters in a hexmap.

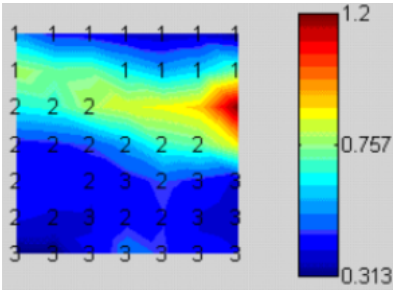


Fig. 18: Iris dataset trained on SOM with grid size 7x7, where the contrasting cyan-like color in the middle represents the color of separation between the clusters, and the blueish areas are data points/input vectors that are similar to each other. [16]

SOM is applicable to our problem as an unsupervised learning approach, since we do not know how many hidden k clusters exists in the NetFlow dataset due to the absence of labels. SOM algorithm has been fitted using the preprocessed dataset with grid size of 200x200 (***Grid size are hyperparameters which can be tuned, with larger grid size, the separation of clusters becomes more visible**). Fig.19 shows that it is hard to determine the number of clusters, due to the diversity of network traffic collected from various different organizations.

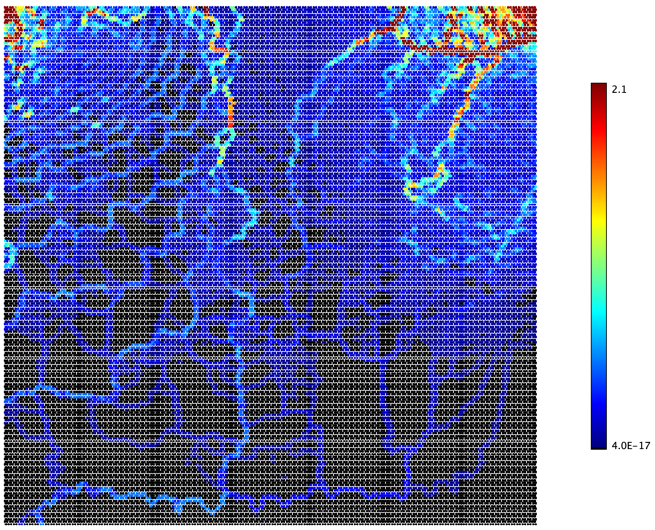


Fig. 19: U-Matrix visualized with a hexmap after SOM training. Contrasting high-value color denotes the cluster separators, while adjacent and similar colors represents the similarity of the data points.

3) Gaussian Mixture Model (GMM): Gaussian Mixture Model belongs to the distribution/probabilistic model that relies on normally distributed data. In real world scenario, many datasets tends to be Gaussian or normally distributed when enough data are collected. Using GMM, we assume there are more than one distributions (multi-modal) existed within the data that can be modelled using multiple Gaussian with unknown parameters or latent variables that needs estimation using the expectation maximization (EM) algorithm. With GMM, to approximate a multi-modal distributions, the mixture of several sub Gaussian distributions can be modelled as one big Gaussian distribution [17].

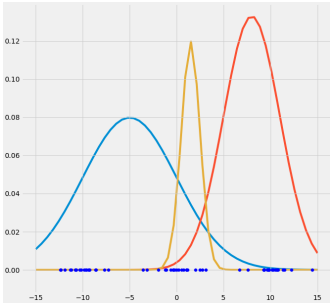


Fig. 20: Gaussian Mixture Model of three normal distributions.

During our analysis of the bimodal distribution Fig.7 and Fig.8, there seems to be two peaks for the bytes feature, thus GMM is worth considering, presumably there could be more than one distributions in the underlying NetFlow dataset. In addition, GMM is also computationally efficient when modelling large datasets, especially vital in the context of Big Data. With more than one feature, modelling the data using one Gaussian is not feasible, since there could be different latent distributions that needs to be estimated [17], [18]. Thus using GMM, one could approximate which distributions a data point belongs to, by generating data points from a mixture of Gaussians. ***(The term gaussian, components and clusters are used interchangeably in the context of GMM)**.

6. Evaluation

Evaluation is conducted on the simulated NetFlow dataset with anomalies injected. It will also discusses the caveat of the proposed clustering algorithms K-Means, SOM and GMM, such as finding the parameters k for K-means and the number of gaussians or components for GMM. Finally, the evaluation of the final proposed clustering algorithm are again used to evaluate on the detection rate of the simulated anomalies.

A. Simulated NetFlow Anomalies

The simulated NetFlow dataset are jointly provided by our industrial partner using the tool ”dns2tcp” to simulate a DNS attack known as ”Data Exfiltration”. The simulated traffic are used to assess the performance of the detection model. Where the model should detect these anomalies and cluster them into the anomalous clusters, separating them from the normal cluster. Data exfiltration (Large file transfer/Unauthorized copying over DNS) was conducted between 1130hrs and 1430hrs.

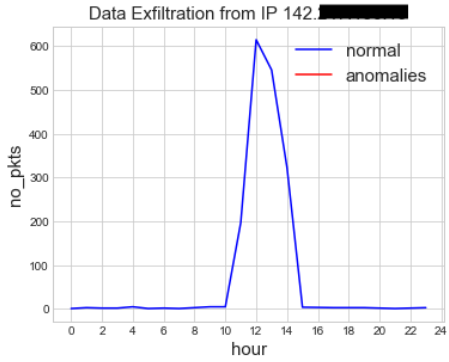


Fig. 21: Sudden spike from 1130hrs to 1430hrs.

1) NetFlow Dataset: In Section 5. Implementation, to determine what constitute the baseline of a normal DNS flow, we will be combining an additional of four more different NetFlow datasets of different date dated months and years apart. After the baseline has been ascertained from the different NetFlow datasets, such that these different NetFlow datasets exhibits similar behaviors/patterns for the typical DNS flow, we can safely combine them and form an even larger NetFlow dataset. This is to ensure that with NetFlow datasets of the future, the normal behavior of a DNS flow should not fluctuate too much, affecting normality. Finally with more data, the approximation of future unseen data can be more accurately identified.

2) Combining different NetFlow Datasets: The idea of combining the different NetFlow datasets is to validate our original assumptions of the preprocessed dataset conforms to the same normality across different NetFlow datasets of different date, e.g. days, weeks and years. The following Fig.22 are the results of combining four different NetFlow datasets, preprocessed and plotted on two principal components dated months and year apart. It is seemingly convincing that the following visualization exhibits the same unique **V** shape pattern for the DNS traffic when compared to Fig.16.

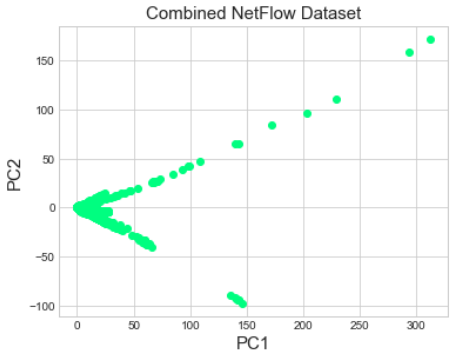


Fig. 22: Four different NetFlow datasets combined.

B. Evaluation of Clustering Algorithms

Our combined preprocessed NetFlow dataset are feed into K-means and GMM algorithm for evaluating the detection rate on the simulated data exfiltration anomalies. The following 1) and 2) discusses the detection rate on the simulated anomalies using both clustering algorithms.

1) Anomalies Detection with K-means: The following Fig.23 shows the data exfiltration anomalies plotted in two-dimensional scatterplot using the features **no_pkts** and **no_bytes**. All the blue points are anomalies, filtering has been done to show only the data exfiltration points. Given our analysis on the K-means clustering results below, K-means cluster these anomalies as normal, and fails to detect any data exfiltration anomalies. It is possible that using the hypothetical assumptions of $k = 2$ based on research and [13] may not be the optimal k choice.

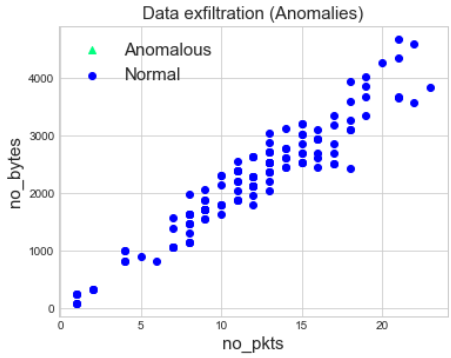


Fig. 23: K-means fails to detect even a single anomalies from the simulated dataset.

Finding the optimal k using the elbow method has also been experimented. However K-means still fails to detect even a single anomalies with the optimal k. It is evident that the model K-means is not complex enough to handle the diversities of DNS traffic.

2) Anomalies Detection with GMM: The GMM is trained on two components/clusters using the covariance type "Full" and max iterations set to 100 due to several trial and errors and hyperparameter tuning.

To determine if GMM is able to detect and cluster anomalies into the anomalous cluster, the simulated data exfiltration traffic are feed into GMM for clustering. The following Fig.24 shows that GMM is able to detect the anomalies with a detection rate of 95% given that the number of pkts and bytes for the anomalies are generally higher as compared to a normal DNS flow, thus the detection of data exfiltration anomalies works well for GMM with only 5% of false negatives, Fig. 24 blue points. In the next section, we aim to overcome these false negatives or mis-clustered anomalies that are very similar to a normal DNS flow.

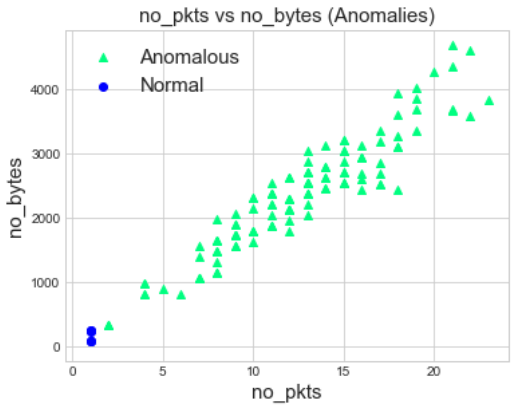


Fig. 24: GMM detected 95% of anomalies using two gaussian.

C. Finding GMM model parameters

Up till now, using the parameter $k = 2$ is a hypothetical assumptions based on existing literatures and research [13]. However, there could be a suitable k that one could choose via statistical inference, such that anomalies can be better identified with the appropriate k. The following 1) Finding no. of Components, explores two techniques for selecting the optimal k for GMM.

1) Finding no. of Components: Two techniques that will be discuss here are Bayesian Information Criterion (BIC) and Akaike's Information Criterion (AIC). Both are statistical model to perform model selection criteria to determine if one model is better than the other. [19].

The following Fig.25 shows that both BIC and AIC are identical with respect to the number of components specified, where the reduction of the maximum likelihood of BIC asymptote at around five components or more. Hence, we can safety choose the minimum number of components with the lowest BIC score where the maximum likelihood is achieved, in this case 5 (Circled in red), where the BIC line was overlapped by AIC, given that both attained the same score.

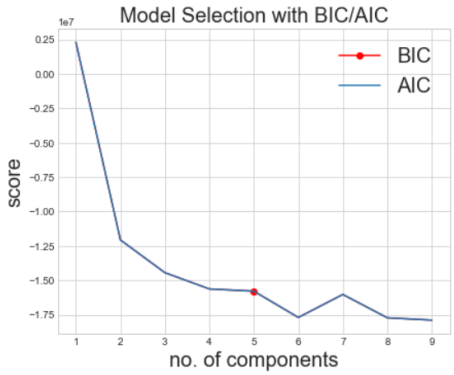


Fig. 25: BIC and AIC scores.

D. Anomalies Detection after Model Selection

After selecting the optimal number of components, in this case the lowest BIC and AIC scores tell us at around five components or more is a suitable trade-off between a better model fit with the extra computation time by adding additional gaussians or components. Thus, the GMM has been re-trained using the same data with the number of components set to five and covariance type "Full". When using this GMM model to cluster the simulated NetFlow dataset, the first cluster is the normal cluster, since it represented 81% of the total DNS flow, and any subsequent clusters from 2..N are assumed to be the anomalous or unknown clusters, where N is the number of clusters/components, in this case five clusters. The simulated data exfiltration anomalies are conducted between 1130hrs - 1430hrs, as shown in Fig.26, the GMM with five components is able to accurately detect all data exfiltration anomalies successfully under Cluster 4, with the trade-off of having more components representing the different mixture distributions of the combined NetFlow dataset Fig.22.

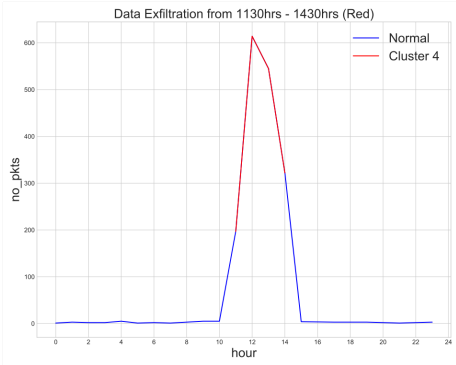


Fig. 26: Data exfiltration anomalies are detected under Cluster 4.

The following TABLE III shows the confusion matrix after GMM detection on the simulated NetFlow dataset filtered to show only the data exfiltration anomalies. The simulated NetFlow dataset have a total of 141 time window aggregated DNS flow of data exfiltration anomalies after preprocessing.

	Normal	Anomalies
Detected Normal	TP	FP
Detected Anomalies	FN	TN=141

TABLE III: Confusion Matrix: All 141 data exfiltration anomalies has been detected by GMM.

E. Kernel Density Estimation (KDE) on Gaussian Mixture

The following TABLE IV are the Gaussian mean μ and mixture weights ϕ of the simulated NetFlow dataset. Where the Normal Cluster/Gaussian represented 68% of the total distribution and the remaining clusters represents only 32%; the simulated data exfiltration anomalies are clustered under Cluster 4 by the GMM model.

Cluster k	Mean μ	Weight ϕ
Normal	86	0.68
Cluster 1	587	0.08
Cluster 2	491	0.12
Cluster 3	1762	0.08
Cluster 4	22040	0.02

TABLE IV: The mean and weight of the average number of bytes with five Gaussian.

The following Fig.27 shows the KDE of the five Gaussian after GMM clustering. We can see that there are some slight overlapping between the Normal (Blue) and Cluster 3 (Purple). By modelling the distribution of the DNS Flows using GMM, data exfiltration anomalies that are similar to normal traffic can be accurately clustered using probabilities/soft assignments.

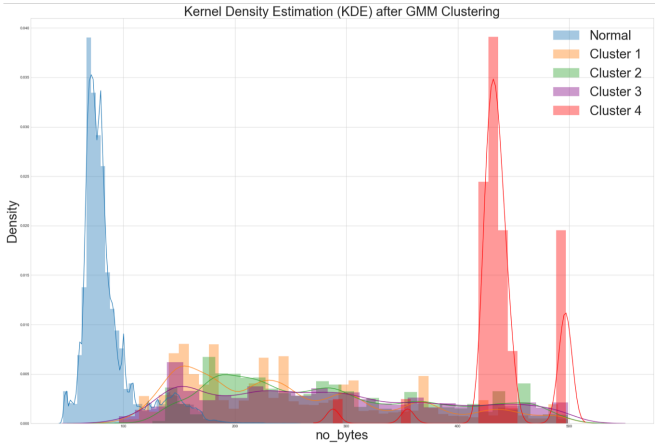


Fig. 27: KDE on a mixture of five Gaussian.

7. Conclusion

Anomaly detection is still a challenging problem that spans decades of research [20]. The purpose of this research is to detect general DNS anomalies that are statistically different from normal behavior using Big Data, given that DNS are often used as a covert channel for attackers to perform malicious activities e.g. data exfiltration. Detecting anomalies in a DNS environment is beneficial and proved to be crucial in any enterprise network. Using GMM with two clusters, the detection model is able to detect data exfiltration anomalies with a 95% detection rate. Further to that, SOM has also been used for cluster analysis to determine if there are any visible clusters in the NetFlow dataset. However, given the nature of the network traffic collected from multi-enterprise network of different organizations, no fixed number of clusters can be obtained due to the diversity of network traffic and the varying traffic behavior of how different organizations operates.

Thus, model estimation techniques using BIC/AIC has been selected to determine the optimal number of components/clusters for the GMM model. Using only five components, the final GMM model achieved a 100% detection rate on the data exfiltration anomalies. However, limited to only data exfiltration anomalies, the evaluation of the model is limited to and only that. Hence, it is probable that our model is not able to detect other kinds of DNS anomalies other than data exfiltration.

To validate the robustness of our detection model, more DNS anomalies needs to be simulated to assess the detection rate/recall of the GMM model, by understanding the patterns of the different kinds of anomalies, experimentation using statistical analysis and evaluation can be further conducted.

8. Acknowledgments

This is an industrial research project supported by the University of Glasgow and Singapore Institute of Technology (SIT) with Industrial Partner. I want to thank my professor Dr. Vivek Balachandran in the department of Singapore Institute of Technology for sharing his knowledge and guidance throughout. Finally, I would also like to thank my industrial supervisor Eugene Chong for giving me a chance for taking on this challenging research project.

References

[1] M. Facure, "Semi-Supervised Learning for Fraud Detection Part 1"," https://lamfo-unb.github.io/2017/05/09/Semi-Supervised-learning-for-fraud-detection-Part-1/, 2017.
[2] M. W. Lucas, "Network flow analysis," ch. 1, pp. 9–11.
[3] Y. Cheng, T. T. Nguyen, H. Zeng, and J. Deng., "Big data analytics in cybersecurity," ch. 2, pp. 27–30.
[4] S. Singh and G. Kaur, "Unsupervised anomaly detection in network intrusion detection using clusters," 2007.
[5] C. Zhang, G. Zhang, and S. Sun, "A mixed unsupervised clustering-based intrusion detection model," 2009.
[6] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised clustering approach for network anomaly detection," 2012.
[7] S. Northcutt and J. Novak, "Network intrusion detection third edition," ch. 6, pp. 103–115.
[8] S. Northcutt and J. Novak, "Network intrusion detection third edition," ch. 6, pp. 113–115.
[9] S. Paul, "Beginner's Guide to Feature Selection in Python," https://www.datacamp.com/community/tutorials/feature-selection-python, 2018.
[10] S. Tripathy and L. Sahoo, "A survey of different methods of clustering for anomaly detection," vol. 6, 2015.
[11] D. S. Terzi, R. Terzi, and S. Sagiroglu, "Big data analytics for network anomaly detection from netflow data," 2017.
[12] A. Trevino, "A Introduction to K-means Clustering," https://www.datascience.com/blog/k-means-clustering, 2016.
[13] G. Munz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," 2007.
[14] A. Ralhan, "Self Organizing Maps," https://towardsdatascience.com/self-organizing-maps-ff5853a118d4, 2018.
[15] D. Miljkovic, "Brief review of self-organizing maps," 2017.
[16] O. K. Pavel Stefanovic, "Visual analysis of self-organizing maps," vol. 16, p. 495, 2011.
[17] C. M. Bishop, "Pattern recognition and machine learning," ch. 9, pp. 430–435.
[18] C. M. Bishop, "Pattern recognition and machine learning," ch. 9, pp. 435–439.
[19] C. Xie, J. Chang, and Y. Liu, "Estimating the number of components in gaussian mixture models adaptively," 2013.
[20] V. Chandola and V. Kumar, "Anomaly Detection: A Survey," 2009.