by

Teja Kommineni

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

School of Computing

The University of Utah

May 2017

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Teja Kommineni**
has been approved by the following supervisory committee members:

Robert Ricci,	Chair(s)	__ Date Approved
Kobus van der Merwe,	Member	__ Date Approved
Suresh Venkatasubramanian,	Member	__ Date Approved
__,	Member	__ Date Approved
__,	Member	__ Date Approved

by **Ross Whitaker**, Chair/Dean of
the Department/College/School of **Computer Science**
and by **David Kieda**, Dean of The Graduate School.

ABSTRACT

Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah. Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

For my parents, Alice and Bob.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION AND SYMBOLS

α	fine-structure (dimensionless) constant, approximately $1/137$
α	radiation of doubly-ionized helium ions, He++
β	radiation of electrons
γ	radiation of very high frequency, beyond that of X rays
γ	Euler's constant, approximately $0.577\,215\ldots$
δ	stepsize in numerical integration
$\delta(x)$	Dirac's famous function
ϵ	a tiny number, usually in the context of a limit to zero
$\zeta(x)$	the famous Riemann zeta function
$\ldots$	$\ldots$
$\psi(x)$	logarithmic derivative of the gamma function
ω	frequency

CHAPTER 1

INTRODUCTION

CHAPTER 2

BACKGROUND

Gaining insights from the network data using Machine Learning and Data Mining is a trending research area. There is a lot of confusion within the literature about the terms ML, DM as they often employ the same methods and therefore overlap significantly. Hence, below we describe the process of ML and DM briefly and establish why we have chosen Data Mining for building our system.

Machine Learning is a method of generating rules from past data to predict the future. A Machine Learning approach usually consists of two phases: training and testing. The following steps are usually performed:

- Identifying attributes (features) and classes from training data.
- Choosing a subset of the attributes for classification.
- Choosing a ML algorithm to create a model using the training data.
- Use the trained model to classify the unknown data.

Data Mining is the process of extracting implicit, previously unknown and potentially useful information from data. Data mining is generally considered as a step in the process of Knowledge Discovery from Data (KDD). KDD usually consists of the following steps:

- Selection of raw data from which knowledge has to be extracted.
- Preprocessing the data to perform cleaning and filtering to avoid noise.
- Transforming the data so that all the attributes in the raw data are aligned towards achieving the common goal.
- Applying Data Mining algorithms to find rules or patterns.
- Interpret the observations of the above step and validate them.

As outlined above though both the techniques have similar steps of preprocessing data they differ on the end goal while ML maps the data set to known classes DM tries to find patterns out of the data set.

The decision of the approach that we want to employ in our problem solving also depends on the type of data we have in hand. If the data is completely labeled, the problem is called supervised learning and generally the task is to find a function or model that explains the data. The approaches such as curve fitting or machine-learning methods are used to model the data to the underlying problem. The label is generally the business or problem variable that experts assume has relation to the collected data. When a portion of the data is labeled during acquisition of the data or by human experts, the problem is called semi-supervised learning. The addition of the labeled data greatly helps to solve the problem. In general case the semi-supervised learning problem is converted to supervised learning problem by labeling the whole data set using the known labeled data and again the same machine-learning methods can be employed here. In unsupervised learning problems, we don't have any labels in the given data set and the main task is to find patterns, structures, or knowledge from this unlabeled data.

The dataset that we used for addressing our problem is flow data collected at routers which is explained in detail in the next section. This flow data generally comes without any labels. As, explained above if the data in hand doesn't have any labels the problem we are solving is called unsupervised learning problem. Also, in the introduction we have mentioned that the main goal of this work is to extract host behaviors from the aggregate data. This problem falls under a category of finding implicit/unseen patterns. Thus, having unlabeled data and the problem that we are trying to solve led us to use Data Mining techniques in building our system.

CHAPTER 3

DESIGN OVERVIEW

Detecting the behaviors implicit in the netflow data and using them to build systems that make network management easier is the overarching premise of this work. In this chapter we describe the design of our system, and how the different components fit into the architecture.

3.1 Components

Our design is comprised of four essential components: Netflow Collector, Feature Engineering, Pattern Detector, and applications as shown in figure 3.1.

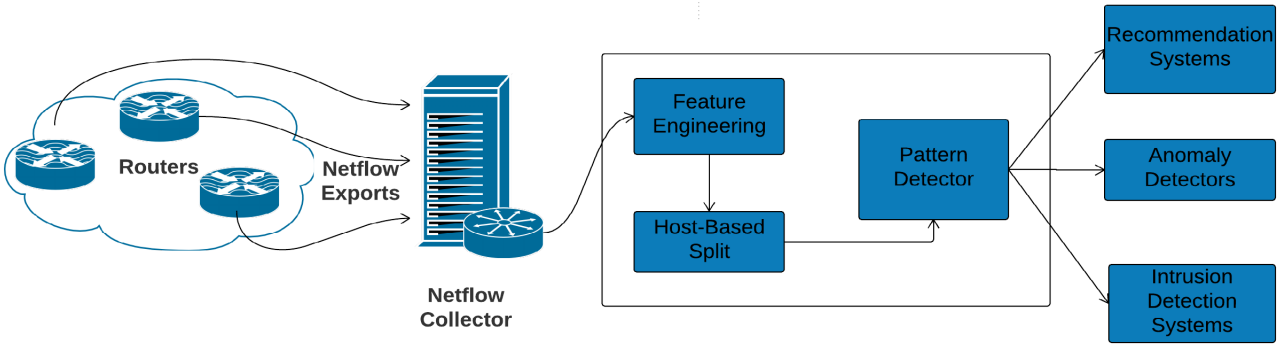

Figure 3.1. Architecture.

3.1.1 Netflow Collector

3.1.2 Feature Engineering

Feature Engineering is the process of transforming raw data into features that better represent the underlying structure to the models so that they increase the accuracy of the model when applied on unseen data. This step comes before modeling and after seeing

the data. Features extracted from the data directly influence the models generated and predictions made out of it. The better the features are the better the results will be. The results that we achieve when we are using ML/DM approaches are a factor of the data set in hand, the features generated , the prediction model and the way the problem is framed. Most of the times even simpler models perform better if the features describe the structure inherent to the data. Below we describe different steps involved in the Feature Engineering with sample set of data.

Source IP	Destination IP	Source Port	Destination Port	Protocol	Duration	Flags	Packets	Bytes
59.2.154.56	155.98.47.116	11045	7547	TCP	318.0618	.....	10	64516.125
223.157.2.51	155.98.44.63	7828	80	TCP			22	19800
190.72.57.37	155.98.47.29	44539	2323	UDP	318.0618	.....	1	
223.157.2.51	155.98.44.63	24342	23		50.023	.....	50	45838.708
113.23.73.77	155.98.36.146	6176	80	TCP	318.0618	..s...	17	
223.157.2.51	155.98.46.35	52336	23	TCP	12.6705	.....	50	388.098
184.105.247.2	155.98.34.233	44969	3389	UDP		.....	50	19800
184.105.247.2	155.98.34.233	1318	21	TCP	40.230		121	19800
208.100.26.22	155.98.35.94	50861	53	ICMP	40.230		21	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 3.1. Netflow Raw Data.

3.1.2.1 Missing Data

Table 3.1 on this page is a sample of network data collected using netflow. In total there are 40 columns for each record with each column capturing different flow information. From the table we can see that there are few missing values. Missing data is a common issue that every data science experiment has to deal with and there could be different reasons why netflow records could be missing some values such as, few filters could be turned on the router that doesn't let the netflow collector collect all the information, overloading of netflow collector and others. We handled Missing data using the following techniques

1) Replace missing values with the mean. For the features such as duration, total packets and bytes exchanged we assume that missing values are distributed similarly to the values that are present for each source IP. In this case, substituting values that represent the existing distribution, such as the mean, is a reasonable approach.

2) Replace missing values with the median/ mode. This is another justifiable way to handle missing-at-random data, although note that it gives a different answer. For categorical data, it's also common to use the mode, the most commonly occurring value. Missing protocols were handled using the mode approach, filling the missing values with the most occurring value for each source IP.

3) Delete columns that are missing too many values to be useful. If a feature is missing too many values, or if there is not enough information available to make reasonable assumptions about how to replace the missing values, you can delete the column entirely. If the feature does not provide useful information, including it can slow down the model's runtime. For many algorithms, having many noisy or uninformative columns can actually degrade their performance. In those cases, deleting these columns during data preparation is the best policy. In our case there were handful of columns that fell under this category and are discarded. flags shown in the **Table 3.1** on the previous page is one among them. After handling missing values our data looks as in **Table 3.2** on this page.

Source IP	Destination IP	Source Port	Destination Port	Protocol	Duration	Packets	Bytes
59.2.154.56	155.98.47.116	11045	7547	TCP	318.0618	10	64516.125
223.157.2.51	155.98.44.63	7828	80	TCP	**62.6935**	22	19800
190.72.57.37	155.98.47.29	44539	2323	UDP	318.0618	1	19800
223.157.2.51	155.98.44.63	24342	23	**TCP**	50.023	50	**20188.098**
113.23.73.77	155.98.36.146	6176	80	TCP	318.0618	17	388.098
223.157.2.51	155.98.46.35	52336	23	TCP	12.6705	50	388.098
184.105.247.2	155.98.34.233	44969	3389	UDP	**40.230**	50	**19800**
184.105.247.2	155.98.34.233	1318	21	TCP	40.230	121	19800
208.100.26.22	155.98.35.94	50861	53	ICMP	40.230	21	45838.708

Table 3.2. After Handling Missing Data

3.1.2.2 Converting categorical to Numerical Variables

Features could be either numerical or categorical variables. When dealing with algorithms that work on numerical data we have to make sure that the categorical variables are handled. For example, in our case feature protocol, has values such as TCP, UDP and others. One approach is to encode them as 0, 1, and 2, respectively converting them to numerical variables. This could pose a problem when using distance measuring algorithms as the distance between the encoded attributes, like 1 (UDP) minus 0 (TCP) doesn't mean anything. Another approach would be to create a feature for each value of the categorical variable and mark it as 0 or 1 based on if the value is present or not. There are both pros and cons to this method. The pros are if we choose to aggregate the values on all features this conversion gives a numerical value to work with. The cons are scaling and standardization could be affected. **Table 3.3** on the current page represents data set after this step.

Source IP	Destination IP	Source Port	Destination Port	TCP	UDP	ICMP	Duration	Packets	Bytes
59.2.154.56	155.98.47.116	11045	7547	**1**	**0**	**0**	318.0618	10	64516.125
223.157.2.51	155.98.44.63	7828	80	**1**	**0**	**0**	62.6935	22	19800
190.72.57.37	155.98.47.29	44539	2323	**0**	**1**	**0**	318.0618	1	19800
223.157.2.51	155.98.44.63	24342	23	**1**	**0**	**0**	50.023	50	20188.098
113.23.73.77	155.98.36.146	6176	80	**1**	**0**	**0**	318.0618	17	388.098
223.157.2.51	155.98.46.35	52336	23	**1**	**0**	**0**	12.6705	50	388.098
184.105.247.2	155.98.34.233	44969	3389	**0**	**1**	**0**	40.230	50	19800
184.105.247.2	155.98.34.233	1318	21	**1**	**0**	**0**	40.230	121	19800
208.100.26.22	155.98.35.94	50861	53	**0**	**0**	**1**	40.230	21	45838.708

Table 3.3. After Converting Categorical data to Numerical

Above two steps fall under a family of techniques called data cleaning. After the data cleaning and removing the irrelevant columns of data using domain knowledge we aggregated netflow data based on source IP as our work focuses learning about hosts from their aggregate data. The **Table 3.4** on the following page shows a sample of aggregated data with ten features. The first record in the **Table 3.4** on the next page conveys the

information that in the given data set the host 59.2.154.56 (source IP) had appeared 450 times. Out of those 450 instances it had contacted 116 unique hosts. A total of 110 different source ports were used in the 450 flows and all the packets were sent to single destination port. The columns TCP, UDP, ICMP indicate how many of these flows fall under each category. So of the 450 flows there are 406 flows in which TCP protocol was used , 4 had UDP protocol used and the rest 40 had used ICMP protocol. And the columns Total Duration, Total Bytes, Total Packets indicate the respective information exchanged by this sourceIP on a whole. This aggregated data is what we use for learning host behaviors. But, before that we have to apply few other feature engineering techniques as described below.

Source IP	Total Flows	Unique Destinations	Unique Source Ports	Unique Destination Ports	#TCP	#UDP	#ICMP	Total Duration	Total Packets	Total Bytes
59.2.154.56	450	116	110	1	406	4	40	318.0618	10	64516.125
223.157.2.51	3	2	78	80	3	0	0	62.6935	22	19800
190.72.57.37	84	10	10	2	70	14	0	318.0618	1	19800
113.23.73.77	18	18	1	23	18	0	0	50.023	50	20188.098

Table 3.4. Aggregated data by Source IP

3.1.2.3 Feature Scaling

Feature Scaling/Feature Normalization, a method used to standardize the range of independent features of data. Failure to standardize variables might result in algorithms placing undue significance to variables that are on a higher scale. For example from the **Table 3.4** on this page it is evident that the total packets and total bytes have higher magnitude compared to total flows and destinations. This range difference shouldn't bias our results. There are different ways to do feature scaling namely, Min-Max scaling, Variance scaling, L2 normalization etc.. The choice depends on the data and different statistics of it such as Mean, Variance, L2 norm. Scaling is not advisable in all instances we might loose valuable information if applied without proper thought. In our case we have chosen simple Min-Max scaling as our goal was merely to change the range of the data and not the distribution and Min-Max scaling helps us in achieving this at a lower cost.

Table ?? on page ?? represents the normalized aggregated data set after applying scaling.

3.1.2.4 Feature Transformation

Transforming Non-normal distribution to Normal, many ML/DM tools perform well on normalized data and having skewed data will give inaccurate results. Log transformations are generally used to normalize the skewed data which is the case with most network data. After transforming the data if it doesn't capture the essence of original data we could look at other options such as square root, cube root. BoxCox is also another technique that changes the distribution of variables from non-normal to normal or near normal. **Figure 3.2** on this page shows the distribution of a feature Total Flows from our aggregated data, as seen from the graph it is heavily skewed towards left and this can be directly attributed to the data set we have in hand. The graph indicates that in most instances number of flows in which a source IP appears is less than 100 and hence we employ transformation techniques to decrease the amount of skewness in the data. **Figure 3.3** on the next page shows the same data after applying log transformation.

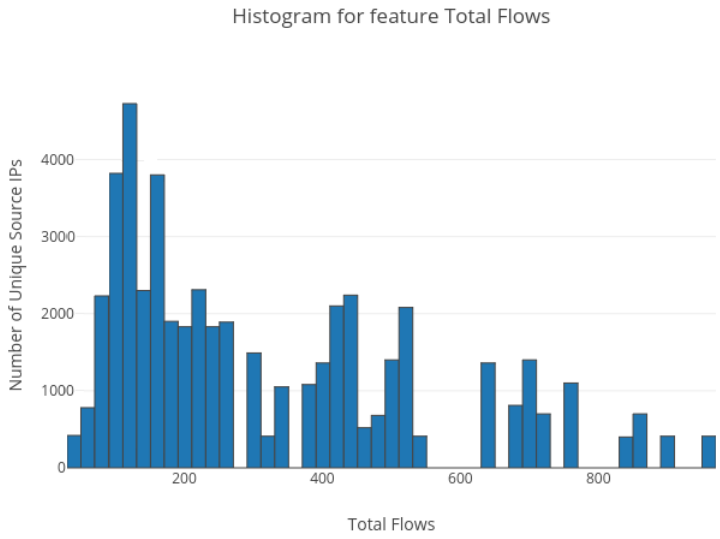

Figure 3.2. Skewed data before transformation

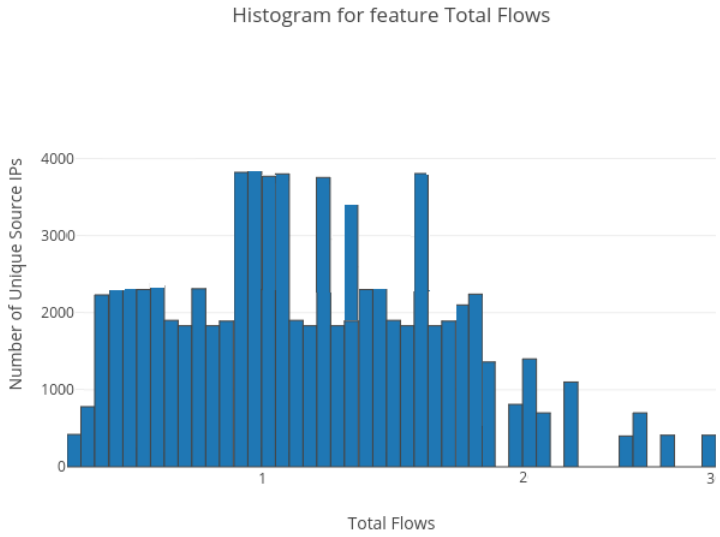

Figure 3.3. Normalized data

3.1.2.5 Feature Selection

Feature Selection refers to the process of selecting a subset of relevant features to model the data. It helps in shorter time periods of execution, overcoming the overfitting problem and making the solution more generic and avoid curse of dimensionality. The central premise of this process is to remove redundant or irrelevant features that don't make much sense to the problem we ought to solve. Feature selection and dimensionality reduction are two confusing terms. Though, both methods seek to reduce the number of attributes in the dataset, but dimensionality reduction does it by creating new combinations of attributes, where as feature selection methods include and exclude attributes present in the data without changing them. Principal Component Analysis, Singular Value Decomposition are examples of dimensionality reduction methods. The techniques used to do feature selection depend on text, numerical variables and they are three general classes of them Filter, Wrapper, Embedded methods. Selecting features in unsupervised learning scenarios is a much harder problem, due to the absence of class labels that would guide the search for relevant information. Problems of this kind have been rarely studied in the literature [4] [6].

The common strategy of most approaches is the use of filter methods Filter model methods do not utilize any clustering algorithm to test the quality of the features. They

evaluate the score of each feature according to certain criteria[6]. it selects the features with the highest score. It is called the filter since it filters out the irrelevant features using given criteria. In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset. Some common examples of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination, Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model. In backward elimination, we start with all the features and remove the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features. Recursive feature elimination, It is a greedy optimization algorithm which aims to find the best performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

We have chosen wrapper methods backward elimination technique for feature selection as it measures the usefulness of a subset of features by actually training a model on it. Though, it is computationally expensive compared to Filter based approaches in our context it was reasonable as we had only 10 features to look into. The result of feature selection is we have detected two features that have very less information gain namely the ICMP ports and duration and hence we removed these two from our feature list making our final feature count to 8. Feature Selection has helped us in enabling the machine learning algorithm to train faster. It reduced the complexity of the model and made it easier to interpret. We were also able to address the overfitting problem.

3.2 Host-Based Split

3.3 Pattern Detector

This is the main

3.4 Applications

CHAPTER 4

IMPLEMENTATION

CHAPTER 5

RELATED WORK

In this chapter, we review works that relate to usage of Machine Learning/Data Mining techniques in the area of networking and discuss in detail how researches are using these methods for different purposes. We also put in to the context how our problem statement and solution differ from the existing work and advances the state of the art.

First, we review the different ML/DM approaches that are being used for traffic classification and intrusion detection in the area of cyber security.

Second, we survey the vast body of work that does packet based inspection and also analyze the shift of research focus towards inspecting at higher granularities specifically the flow based inspection techniques.

Finally, we discuss how unsupervised approaches are being used to look at network data at higher granularities and compare our solution with the prior work.

5.1 Data Mining and Machine Learning Techniques for Intrusion Detection

There are many papers published in the area of cyber security describing the different ML/DM techniques used. Buczak et al [5] provided a survey of the popular techniques used and thoroughly describe ML/DM methods which provides a good starting point to people who intend to do research in the area of ML/DM for intrusion detection.

The survey by Buczak et al [5] concentration has been mainly on explaining the ML and DM methods where as Bhuyan et al. [3] in his paper described different techniques used for network anomaly detection in detail. Garcia's work [17] also explained different intrusion techniques in detail. He extended his work to use Machine Learning techniques for anomaly detection with focus on signature based intrusion-detection. Narayan et al. [22] proposed a hybrid classification method utilizing both Naive Bayes and decision trees for intrusion detection. While their findings had higher accuracy applying these

supervised techniques to our dataset was not feasible as we had very less labeled data.

Wired networks generally have multiple layers of security before the intruder enters the network. But, the wireless networks are more vulnerable to malicious attacks. They add a whole new semantics to the network security such as dynamically changing topology, different authentication techniques and ad-hoc formation of networks. The process followed in this paper works in the context of both wired and wireless networks. Zhang et al. in his work [28] provides a perspective focused only on wireless network protection.

5.2 Machine Learning for Traffic Classification

Apart form using ML/DM techniques for intrusion detection one other field where these have been extensively used is to classify the network data. The survey [21] lists the prominent ML/DM techniques used to classify the data and describes the need for traffic classification. In order to provide the promised Quality of service and be liable to the government laws network administrators should be able to distinguish the traffic on their network.

Traditional well-known port number based classification is not sufficient on it's own to distinguish different applications as different services are using http protocol to send their data and obfuscate from the traffic classifiers. Also, papers such as choicenet [27] point out that to develop an economy plane for the internet similar to the implicit content based billing architecture in mobile architecture there is need for network administrators to classify the network data.

Naive Bayes form is the simplest technique used for this type of classification and has been explained in greater detail in the work of Andrew and Denis [20]. This work was extended by applying Bayesian neural network approach [1] for increasing the accuracy. Though these classification techniques proved to be efficient in differentiating network data they have a drawback that they can only distinguish the network data into known classes which is in contrast to our goal of identifying implicit behavior which is not known in advance.

Renata[2] looked at unsupervised techniques for traffic classification. In contrast to the existing approaches he followed a principle of early detection. Accordingly, he looked at the first few packets of tcp flow and classified them into different applications. The

underlying logic behind this method is that during handshake process each application behaves in a specific way exchanging a particular sequence of messages. He used K-Means algorithm to form clusters. Initially, training data collected over a period is used to generate a model which gives a set of clusters. When new data arrives simple Euclidean distance is used to map this flow to a cluster. The flow belongs to the cluster which it is closest to. Though, this approach had 80 accuracy it has few drawbacks, If we cannot capture initial few packets of a service it's effectiveness is compromised. If a flow doesn't fall under any cluster the behavior is undefined. Renata's [2] approach was similar to us as we also explored unsupervised techniques in our system but they differ in the point that they examine the packet data and map their clusters to only known classes of traffic.

Jeffrey and Mahanti et al. [14] explored hybrid techniques for traffic classification. The intuition behind this exploration is that not all data available will be labeled and when data from new services get appended to the existing dataset the supervised learning techniques are falling short in recognizing them and they map the new data set to one of the existing classes. To overcome this shortcoming they approached the problem in two steps. In the first step the dataset containing labeled and unlabeled data is passed to a clustering algorithm. In the second step the labeled data is used to classify clusters and this is done as follows: Within each cluster all the labeled data is considered and the label with majority forms the label of this cluster. Clusters without any labeled data are classified as 'Unknown'. When new data arrives it will be assigned based on the Euclidean distance to the closest cluster similar to [2] . This has combined advantages of supervised and unsupervised learning. It also decreases the training time because of few labeled data. It's uniqueness comes from being able to map the data from new applications and services to Unknown cluster or to their respective clusters if they are simple variation of existing application's characteristics. A slight variation of this approach has been used by us during our experiments and the following issues have been noticed. First, there were cases in which we have seen a cluster mapping to multiple labels as both turned out to be major labels in the cluster differing by a small value. In this case labeling a cluster just based on majority doesn't suggest the clusters behavior. Second, the amount of labeled data could be negligible compared to the size of the cluster. Third, two clusters could turn out to have similar labels. Lastly, the time consumed for this two step approach is high as we had to

iterate through the data twice. Noticing that this technique isn't inline with our goals we have embraced a totally unsupervised approach for our problem.

5.3 Usage of flow records for IDS/Classification

Traditionally Network data inspection is done by inspecting the contents of every packet. But, the granularity at which this is happening is changing based on the requirements. Earlier packet inspection is used to be a norm but inspecting the contents of every packet is prohibitive in this data-centric age. In this section we look at how flows (aggregated packet data) are being used as input for ML/DM algorithms in place of packets.

In our opinion, flow-based detection should not be seen as a replacement but should be treated as a complement to packet based inspection. We envision that a network administrator should be able to understand the behavior of his network through an aggregated view of network data (in our case flows) and should dive into packet data only when suspicious about a activity. This two step approach will ease the way the network administrators manage their network.

Work of Li et al. [15] and Gao et al. [16] are two examples of Dos detection using flows. In their approach a process tracks the presence of a flow in a specified time frame and an other process runs an anomaly-based engine that triggers if there is a sharp variation from the expected mean. A similar approach was proposed by Zhao et al.[29] to identify the IP addresses of the Scanning hosts and Dos attackers using the flow data. Our system though uses flows captures a broader view apart from identifying these specific attacks. Network Flows are also used in building Worm Detectors [8] [7], Botnet Detectors [24] [19] [18]. Specifically, the botnet detector built by Livadas et al. [19] is of interest as they build a model using the aggregated flows similar to ours.

REFERENCES

[1] T. Auld, A. W. Moore, and S. F. Gull, *Bayesian neural networks for internet traffic classification*, IEEE Transactions on neural networks, 18 (2007), pp. 223–239.

[2] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, *Traffic classification on the fly*, ACM SIGCOMM Computer Communication Review, 36 (2006), pp. 23–26.

[3] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, *Network anomaly detection: methods, systems and tools*, Ieee communications surveys & tutorials, 16 (2014), pp. 303–336.

[4] C. Boutsidis, P. Drineas, and M. W. Mahoney, *Unsupervised feature selection for the k-means clustering problem*, in Advances in Neural Information Processing Systems, 2009, pp. 153–161.

[5] A. L. Buczak and E. Guven, *A survey of data mining and machine learning methods for cyber security intrusion detection*, IEEE Communications Surveys & Tutorials, 18 (2016), pp. 1153–1176.

[6] M. Dash, K. Choi, P. Scheuermann, and H. Liu, *Feature selection for clustering-a filter solution*, in Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, IEEE, 2002, pp. 115–122.

[7] T. Diibendorfer and B. Plattner, *Host behaviour based early detection of worm outbreaks in internet backbones*, in Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on, IEEE, 2005, pp. 166–171.

[8] T. Dubendorfer, A. Wagner, and B. Plattner, *A framework for real-time worm attack detection and backbone monitoring*, in Critical Infrastructure Protection, First IEEE International Workshop on, IEEE, 2005, pp. 10–pp.

[9] A. Einstein, *Eine Neue Bestimmung der Moleküldimensionen. (German) [A new determination of molecular dimensions]*, inaugural dissertation, Bern Wyss., Bern, Switzerland, 1905. Published in [10].

[10] ——, *Eine neue Bestimmung der Moleküldimensionen. (German) [A new determination of molecular dimensions]*, Annalen der Physik (1900) (series 4), 324 (1906), pp. 289–306. See corrections [11, 12]. This is a slightly revised version of Einstein's doctoral dissertation [9].

[11] A. Einstein, *Bemerkung zu meiner Arbeit: Eine Beziehung zwischen dem elastischen Verhalten. (German) [Remark on my paper: "A relationship between the elastic behavior ..."]*, Annalen der Physik (1900) (series 4), 339 (1911), pp. 590–590. See [13].

[12] A. Einstein, *Berichtigung zu meiner Arbeit: Eine neue Bestimmung der Moleküldimensionen. (German) [Corrections to my work: a new determination of molecular dimensions]*, Annalen der Physik (1900) (series 4), 339 (1911), pp. 591–592. See [10].

[13] ——, *Eine Beziehung zwischen dem elastischen Verhalten und der spezifischen Wärme bei festen Körpen mit einatomigem Molekül. (German) [A relationship between the elastic behavior and the specific heat of solid bodies with monatomic molecules]*, Annalen der Physik (1900) (series 4), 339 (1911), pp. 170–174, 590. See remarks [11, 12].

[14] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, *Semi-supervised network traffic classification*, in ACM SIGMETRICS Performance Evaluation Review, vol. 35, ACM, 2007, pp. 369–370.

[15] Y. Gao, Z. Li, and Y. Chen, *Towards a high-speed routerbased anomaly/intrusion detection system*, Technical Report, (2005).

[16] ——, *A dos resilient flow-level intrusion detection approach for high-speed networks*, in Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on, IEEE, 2006, pp. 39–39.

[17] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, *Anomaly-based network intrusion detection: Techniques, systems and challenges*, computers & security, 28 (2009), pp. 18–28.

[18] A. Karasaridis, B. Rexroad, D. A. Hoeflin, et al., *Wide-scale botnet detection and characterization.*, HotBots, 7 (2007), pp. 7–7.

[19] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, *Usilng machine learning technliques to identify botnet traffic*, in Local Computer Networks, Proceedings 2006 31st IEEE Conference on, IEEE, 2006, pp. 967–974.

[20] A. W. Moore and D. Zuev, *Internet traffic classification using bayesian analysis techniques*, in ACM SIGMETRICS Performance Evaluation Review, vol. 33, ACM, 2005, pp. 50–60.

[21] T. T. Nguyen and G. Armitage, *A survey of techniques for internet traffic classification using machine learning*, IEEE Communications Surveys & Tutorials, 10 (2008), pp. 56–76.

[22] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, *Modeling intrusion detection system using hybrid intelligent systems*, Journal of network and computer applications, 30 (2007), pp. 114–132.

[23] S. Singh, *Fermat's Enigma: The Epic Quest to Solve the World's Greatest Mathematical Problem*, Walker and Company, 435 Hudson Street, New York, NY 10014, USA, 1997.

[24] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, *Botnet detection based on network behavior*, in Botnet detection, Springer, 2008, pp. 1–24.

[25] R. Taylor and A. Wiles, *Ring-theoretic properties of certain Hecke algebras*, Annals of Mathematics, 142 (1995), pp. 553–572. This paper is a companion to [26], providing the remedy for the flaw in Wiles' 1993 proof of Fermat's Last Theorem. See also [23].

[26] A. Wiles, *Modular elliptic curves and Fermat's Last Theorem*, Annals of Mathematics, 142 (1995), pp. 443–551. This paper contains the bulk of the author's proof of the Taniyama–Shimura conjecture and Fermat's Last Theorem, carried out at Princeton University. The companion paper [25] contains the solution to the flaw discovered in the proof that Wiles announced on June 23, 1993, in Cambridge, England. See also [23]. In March 2014, now Royal Society Research Professor Sir Andrew John Wiles of Oxford University was awarded the prestigious Abel Prize in Mathematics for this proof — an award that also carries a cash prize of six million Norwegian crowns, or about US$722,000.

[27] T. Wolf, J. Griffioen, K. L. Calvert, R. Dutta, G. N. Rouskas, I. Baldin, and A. Nagurney, *Choicenet: toward an economy plane for the internet*, ACM SIGCOMM Computer Communication Review, 44 (2014), pp. 58–65.

[28] Y. Zhang, W. Lee, and Y.-A. Huang, *Intrusion detection techniques for mobile wireless networks*, Wireless Networks, 9 (2003), pp. 545–556.

[29] Q. Zhao, J. Xu, and A. Kumar, *Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation*, IEEE Journal on Selected Areas in Communications, 24 (2006), pp. 1840–1852.