SMU Data Science Review

Volume 1 | Number 2 Article 10

2018

Data Center Application Security: Lateral Movement Detection of Malware using Behavioral Models

Harinder Pal Singh Bhasin
Southen Methodist University, Dallas, Texas, hbhasin@smu.edu

Elizabeth Ramsdell
eramsdell@mail.smu.edu

Albert Alva
aalva@mail.smu.edu

Rajiv Sreedhar
ShieldX Networks, Inc., rajiv@shieldx.com

Medha Bhadkamkar
Hewlett Packard Enterprise, Inc., medha.bhadkamkar@hpe.com

Recommended Citation
Bhasin, Harinder Pal Singh; Ramsdell, Elizabeth; Alva, Albert; Sreedhar, Rajiv; and Bhadkamkar, Medha (2018) "Data Center Application Security: Lateral Movement Detection of Malware using Behavioral Models," *SMU Data Science Review*: Vol. 1: No. 2, Article 10.
Available at: https://scholar.smu.edu/datasciencereview/vol1/iss2/10

Data Center Application Security: Lateral Movement Detection of Malware using Behavioral Models

Harinder Pal Singh Bhasin[1], Elizabeth Ramsdell[1], Albert Alva[1], Rajiv Sreedhar[2], and Medha Bhadkamkar[3]

[1] Southern Methodist University, Dallas, Texas
[2] ShieldX Networks, Inc., San Jose, California
[3] Hewlett Packard Enterprise, Inc., Palo Alto, California

Abstract. Data center security traditionally is implemented at the external network access points, i.e., the perimeter of the data center network, and focuses on preventing malicious software from entering the data center. However, these defenses do not cover all possible entry points for malicious software, and they are not 100% effective at preventing infiltration through the connection points. Therefore, security is required within the data center to detect malicious software activity including its lateral movement within the data center. In this paper, we present a machine learning-based network traffic analysis approach to detect the lateral movement of malicious software within the data center. Our approach employs an unsupervised learning approach that uses the metadata of network transactions to learn the normal application network traffic behavior and detect anomalous communications. Utilizing over two million records for the training data and four hundred thousand records for validation, our approach identified 0.61% of the communications as anomalous. The fact that any anomalies were successfully identified further confirms our theory that monitoring data center traffic for anomalous communications is an effective and necessary approach to detecting malicious software activity that remains internal to the data center.

1 Introduction

Data centers are an integral part of our daily lives, yet most people are unaware that they constantly use services housed in data centers. The widespread adoption of cloud services allows our smart phones, laptops, self-driven automobiles, and home automation devices, to name but a few connected devices, to provide a broad range of functionality and services that the devices alone are unable to provide. Applications routinely communicate from our personal devices to data centers, often relaying sensitive financial information, such as credit card numbers, and personal information, such as our current location.

Data center transactions can automatically involve multiple servers located within a single data center. Communication between servers for each of these transactions may communicate the sensitive and personal information that is transferred as part of the request. The communication of sensitive information must be secured, even when the communications are contained solely within a single data center. A data center must provide methods to securely process this information in order to protect consumer privacy and prevent theft is a complex problem. Protecting this information from malicious software, however, is a complex problem since the traffic between the servers within the data center is expected to be reliable and predictable.

It is necessary to monitor and understand the behavior of this traffic in order to detect anomalies leading to malicious software detection. Anomaly detection on network is a demanding space where quite a bit of research has already been done over the years and it continues because the demand for secure, high performance systems with increased storage capacity continues to grow. The challenge in anomaly detection is that the majority of the data consists of normally behaved patterns and a small portion is ever detected as anomalous.

Our solution uses Machine Learning techniques and algorithms to further provide mechanisms to better measure for anomaly detection. Our data set dictates Unsupervised Learning technique since the data is unlabeled. A trained set against which to classify is unavailable. The data center traffic data set is unlabeled, binary, multiclass, and yet very similar. The Jaccard similarity coefficient is applied to find dissimilarities in the traffic. Records from traffic data set are measured for their similarities and a distance is calculated. Clusters of similar distances are formed to classify as similar behavior. The training data set is used to form clusters of different traffic patterns. For validation the test data set is run against the trained model to detect anomalies.

The outcome from the training data set produced different results as we used full model, one-feature model, and a restricted model. The one-feature model was comprised of one feature and produced many dissimilarities resulting in numerous clusters. The quality of these clusters was rather weak in comparison to the restricted model. In the restricted model we used four features. The restricted model produced fewer but densely populated clusters. The quality of these clusters was tested using the distance between the clusters, measured from the centroid to centroid of resulting clusters. A test data set used for validation produced few dissimilarities attributed to anomalous data.

Our training data set with cluster quality testing and validations resulted in 0.61% of anomalous amongst the test data. Our theory to learn the application traffic behavior for detecting malicious software has shown positive results.

The rest of the paper is organized as follows: Section 2 describes related work done in application behavioral analysis field. In Section 3, we describe the data center application server and traffic patterns. In Section 4, we analyze the feature selection and the data set used for our experiments. In Section 5, we describe how we used the similarities in traffic to detect anomalies. In Subsections 5.1 and 5.2, we describe the model construction, implementation, and validation. In Section 6, we describe results from our experiments. In Section 7, we analyze our results. In Section 8, we discuss ethics surrounding the application security. The Section 9, we conclude our findings and review the future work.

2 Related Work

Some research has been done by Cao [1] to analyze malware behavior at the application level. As described in the research paper, APIs (Application Programming Interface) collect the internal call structure to monitor the application behavior. This approach requires an in depth knowledge of proprietary applications and learning the implementation to appropriately classify a "good" behavior. Our approach does not require any knowledge of proprietary interfaces and implementation for classifying the behavioral pattern.

A neural network implemented in Network-based intrusion detection using neural networks is presented by Palagiri [2]. This relatively basic method, however, does not produce preferential results as it does not flag simultaneous attacks. It is better suited for individual attacks. We leave this as an opportunity to expand on our future considerations. Our approach is to use Machine Learning techniques and algorithms to further provide mechanisms to better measure for anomaly detection. Machine Learning techniques can be divided into three categories: Supervised Learning, Reinforcement Learning, and Unsupervised Learning. In Supervised Learning the data is labeled or structured where the desired threats have already been classified. Reinforcement Learning develops an agent that improves its performance based on interactions with the environment. A classic example is a chess game where an agent continues to learn with the goal of either winning or a losing. Unsupervised Learning is used where the data is unlabeled or unstructured and a trained set against which to classify is unavailable.

Portnoy [3] discusses that in the case of Supervised Learning, the labeled data needed is either impossible to obtain or very expensive. Using stagnant, labeled data assumes that the labeler was cognizant of every possible type of attack, which is never the case. This reinforces the need to have a well-working Unsupervised model. The model in the intrusion detection with unlabeled data using clustering [3] resource takes a similar approach to what is presented here. The authors work with unlabeled data center in order to form clusters. They mark all small clusters as being anomalies. As discussed, this approach assumes that the number of normal transactions greatly outweigh the number of attacks within the training data. This assumption proves to be a fairly accurate starting point for an intrusion detection model.

Another limitation of many current anomaly detection models is the time component to this type of data. The models are based on a normal behavior that is derived from a training set. While this set may be what is normal at the time of training, transactional behavior is constantly changing, thereby creating a need for repetitive model retraining. This creates a possibility for increased error as the model will be trained on the newest available data. This data may include attacks that were not identified which will then be classified as normal behavior for all incoming interactions. Most machine learning techniques such as neural networks and SVMs are sensitive to the noise this creates in the training data. In the model presented by Hu, Liao and Vemuri [4], robust support vector machine technique is used and compared to the SVMs and K-nearest neighbor results. Robust support vector machines account for the overfitting of the initial model due to the noisy nature of the real-time training data. Another benefit of this method is a shorter testing time due to having fewer support vectors than a traditional Support Vector Machine. The authors used the three separate detection methods on the same clean data set and a deliberately noisy data set, into which simulated anomalous behavior has been injected. The results for the Robust Support Vector machine based on these data were promising in comparison to the other methods. It showed lower false positive rates and higher outlier detection rates. It also showed the lowest performance decline when used in the presence of noise.

In addition to behaviors changing over time, the timeliness of the model needs to be considered. In the event of an attack, the speediness of detection is vital in disaster management. Algorithms need to be able to detect anomalies in real-time. This can be difficult with the previously discussed dynamic environment and the large amounts of transactions happening at a given time. The need to learn continuously is addressed in Unsupervised real-time anomaly detection for streaming data [5]. Here, the authors used streaming data from Numenta Anomaly Benchmark to test their model. This is a useful resource for the testing phase as it contains real-world streams with labeled anomalies, allowing for testing to be done in real-time. To detect the anomalies, this article offers the more advanced Hierarchical Temporal Memory tool. This continuous learning model considers the spatiotemporal characteristics of its input. It does not, however, model of detect anomalies. The output from this method is then used as input for anomaly detection analysis. While this is outside the scope of this paper, we want to present this as a possible iteration technique.

3 Application Servers

As shown in Figure 1, a typical data center is considered in the south (S) side of the perimeter. Connections originating from the Internet to the data center are termed as N-S traffic. In addition to the N-S traffic there may be connections between the various application workloads in the data center. These connections are termed as east(E)-west(W) traffic. Communication to process a transaction within a data center could potentially span across any number of (application) servers shown as east (E) to/from west (W) (a.k.a. lateral).

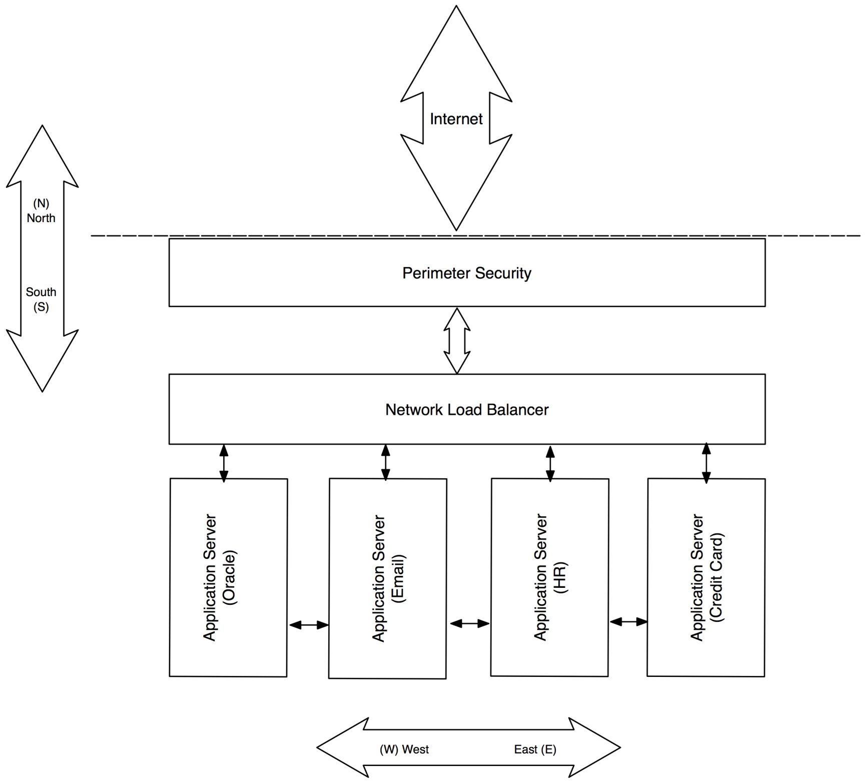

Fig. 1: A typical data center infrastructure

A malware entering into data center from Internet is usually detected at the perimeter level for intrusions. Any malware entered undetected could land up on any of the servers from east to west within the data center. Once a malware enters an application server within a data center the anomaly detection becomes complex since application servers "should" behave as expected by their respected purpose and functionality, and there is no additional security built at an application level to protect from intrusions. These applications execute are well behaved and respond to requests as normally as they would be from a well behaved requesting application. In out experiment we learn the behavior of these applications by classifying their network traffic using machine learning techniques. The network traffic carries ample information to learn about the normal behavioral pattern. As an example a communication is possible and considered as "good" if request/response falls between "HR" and "Email" application servers (between east and west). An abnormal transaction would be "HR" application server trying to communicate with the "Credit Card" application server. Or in a typical malware scenario the requesting application might not be labeled one of the "well defined" application and could potentially be trying to establish communication to other "well defined" application. A typical flow of the information could potentially require to touch multiple entities as described in Figure 1 in terms north (N), south (S), east (E), and west (W).

4 Feature Selection

The feature selection played an important role in our experiment as the data center application network traffic has similarities in addition to being enormous in size. For our experiment creating a workflow as described in Figure 2 improved performance and our ability to narrow the focus for anomaly detection. The workflow shown in Figure 2, separates out different components to modularize the implementation providing future research feasibilities.

Data set for our research comprised of fewer attributes without any user data (a.k.a. payload). As show in the Figure 2 traffic is captured at the Network Packet level, however, we chose to use metadata once the packet processing has been performed. Packet processing is not relevant for our research since the metadata which consists of information relevant to application and communicating entities.

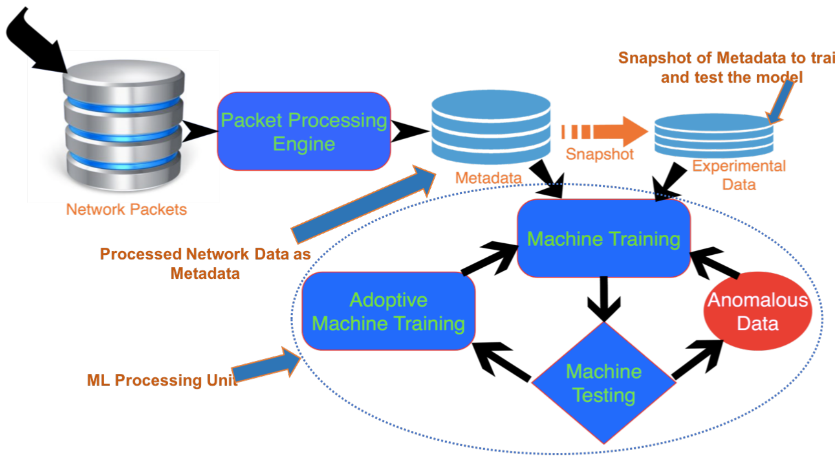

Fig. 2: Framework and the Workflow for Anomaly Detection

In our research we used the data set comprising of normal and anomalous records. We used the metadata in three different sets for our experiments: a full model as described in Table 1, a restricted model as shown in Table 2, and a one-feature model as shown in Table 3. The full model used the entire data set presented in metadata consisting of twelve features. In a restricted model we limited our experiment to four features. In each of these experiments we used 1K, 750K, and 2M size for training data sets. These records were retrieved from live traffic within a data center.

Our focus from full model was to learn the overall traffic behavior in a process to structure the data set with respect to application traffic features. The full set of features produced enormous similarities with the presence of almost redundant information such as srcIpAddress and srcMacAddress. These attributes were removed and experiment was rerun to further help narrow down the list to as described in Table 2 for the restricted model.

Features	Description
chassisId	Machine Identification
srcIpAddress	IP Address of the originator.
dstIpAddress	IP Address of destination.
clientIp	IP Address of a client.
serverIp	IP Address of the server.
srcMacAddress	MAC Address of source.
dstMacAddress	MAC Address of destination.
srcPort	Port Address of source.
dstPort	Port Address of destination.
microServiceType	Type of software service in numeric.
microServiceTypeString	Type of software service in string.
minroServiceInstanceId	Identification number of this software service.
tcpEvent	TCP event type associated with this event.
protocol	Protocol associated with this event
timeStamp	Time when this metadata was captured

Table 1: Full Model

We continued our experiment with removing additional attributes to understand the impact it might have in our results. The feature set was brought down to dstPort as shown in Table 3, with one feature only. However, after few iterations we learned the traffic behavior and the relevance of IP addresses and protocol to application servers. This led us to re-adjust the features to find the right combination.

Features	Description
srcIpAddress	IP Address of the originator.
dstIpAddress	IP Address of destination.
dstPort	Port Address of destination.
protocol	Protocol associated with this event

Table 2: Restricted Model

Features	Description
dstPort	Port Address of destination.

Table 3: One-feature Model

In the initial exploration of the data, the possible indicators for anomalies behavior are considered. The regular interactions between the IP Addresses can be used to identify when a transaction is outside the norm. Figure 3 shows each IP Address as a node, then connected to all Addresses that it converses with. If one of the right-most Addresses tried to send to one of the left-most, this should be flagged as anomalies in our model. The frequency of these communications is then considered. An outlier may follow the same normal path as shown in Figure 3, but transact on a higher frequency.

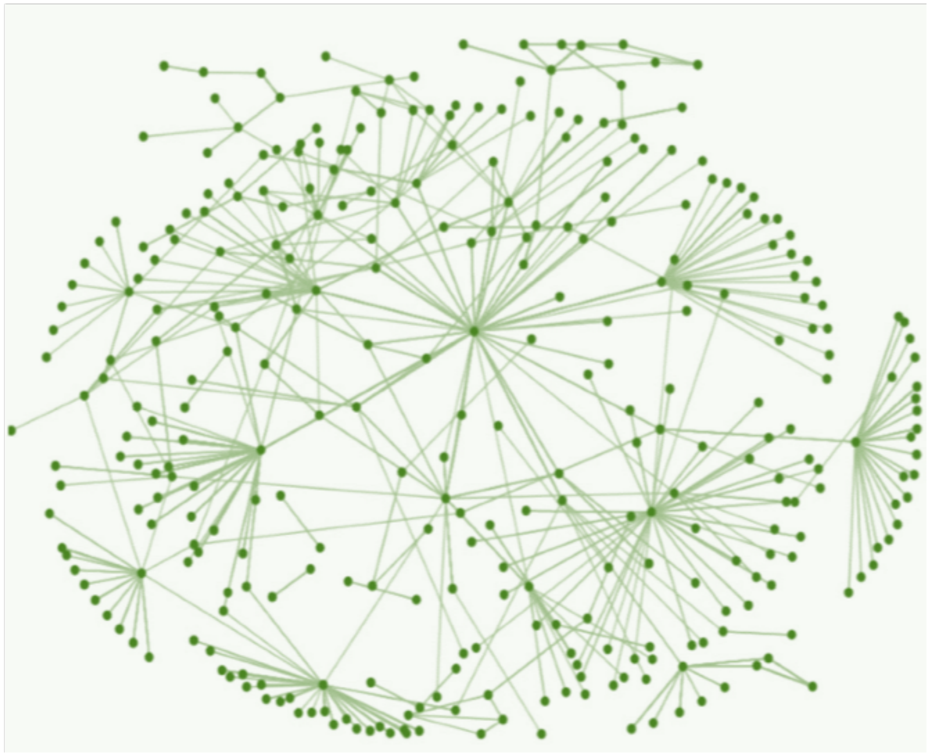

Fig. 3: A Network Diagram of Servers

Another data point that is explored as a possible indicator is the timestamp. These transactions happen all throughout the day, but certain communications should only happen during specific times of the day. If there is an increase in the amount of communications within a given time, this should be flagged as anomaly behavior. Figure 4 shows an example of the highest frequency timestamps in the data center. Using this information in real-time will give us the ability to recognize increased traffic quickly.

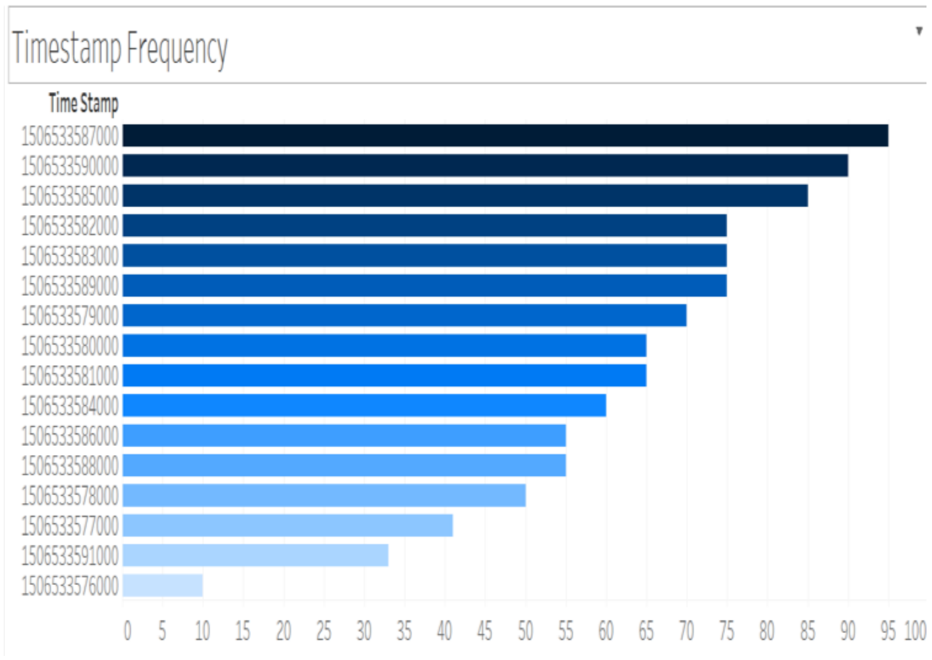

Fig. 4: Frequency of Traffic

Figure 5 shows a correlation of nodes with a dense color reflecting frequently used network paths among the application servers.

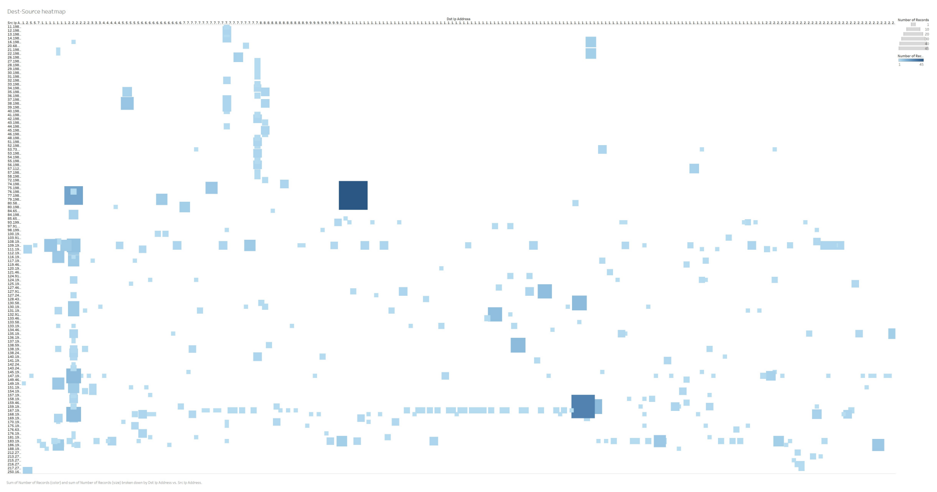

Fig. 5: Correlation of Traffic

5 Jaccard Similarity Coefficient for Anomaly Detection

We used Unsupervised Learning in our research for anomaly detection as majority of the data is well behaved with similarities (non-anomalous). The complexity is discovered as we analyze the data to recognize patterns and learn from the traffic behavior. Jaccard Similarity Coefficient to identify dissimilarities produced favorable results in our experiment. Jaccard method application is well suited for binary and multi-class feature since it operates on sets. This method takes intersection of two sets and divides by the their union resulting in a similarity score. Dissimilarity is calculated by subtracting the result from 1 (1 - Similarity Score), as shown in Equations 1 and 2 below.

$$SimilarityScore(A, B) = \frac{A \cap B}{A \cup B} \tag{1}$$

$$DissimilarityScore = 1 - SimilarityScore \tag{2}$$

The result is produced in a numeric score between 0 and 1. A higher score translates to a greater dissimilarity among the two records. In our experiment each record is treated as a set and measured against the remaining records. The experiment used both full and restricted feature sets. Our experiment further constructed clusters of the collected similar scores from Jaccard method in order to identify anomalous record.

5.1 Model construction

The model construction can be divided in two as parallel constructions is required when measuring the similarity score. The parallel construction is to form clusters upon score computation from two records. A similarity criteria score is defined as tunable to perform several iterations in order to refine the model. This score is used to identify each records as similar amongst other records in order to form clusters. The algorithm iterates over records to find Jaccard similarity index until it meets the criterion of similarity measure. Clusters are formed with similar scores as a second part of the model construction. The algorithm defines first record of a cluster as a centroid and uses it to measure against to form clusters of scores meeting the similar score criteria. The score meeting the low water mark is considered to be part of a cluster. Any score falling out of this criteria is resulted in a separate cluster or part of an existing cluster.

As shown in Algorithm 1 the iterative process ensures the existing clusters are checked before forming a new one if the criterion is not met.

Algorithm 1 Dissimilarity algorithm

```
 1: function computeScore
 2:     $distance \leftarrow$ score of $-1$
 3: $top$:
 4:     if $i > maxRecords$ then return done
 5:         $distance \leftarrow JaccardMethod(existingRecordFromCluster, thisRecord)$.
 6:         if $distance <= lowWaterMark$ then
 7:             $addToExistingCluster \leftarrow$ thisRecord
 8:         $i \leftarrow i + 1$.
 9:         goto $top$.
    return distance
10: procedure insertVector(thisRecord)
11:     $distance \leftarrow computScore(thisRecord)$
12:     if $distance == -1Or > lowWaterMark$ then
13: $top$:
14:         if $i > maxRecords$ then return
15:             if $noExistingRecord$ then
16:                 $addToNewCluster \leftarrow$ thisRecord
17:             $i \leftarrow i + 1$.
18:             goto $top$.
```

5.2 Model validation

Similar to model construction the validation is divided in two. The model is validated for the algorithm and the cluster formation. The algorithm validation is further divided in two parts. The first part is validation by constructing synthetic test data to run against the trained model. The synthetic data is constructed with good data as well as anomalous data with instrumentation. Upon refining and training the model a new data set is constructed from the existing data set from the working model by randomly selecting 20% of the records. These records are then run against the model to ensure they measure with same similarity scores as expected. The second part of the algorithm validation required us to construct another data set comprising of randomly selected records with randomly swapped features within the records and across records. This newly constructed data is then run against the trained model. Very few records were observed as anomalous. This is primarily due to the fact application traffic has quite a bit of similarity. Both of these test cases with synthetic data confirmed the algorithm is working as designed.

The second part of the model validation is performed to ensure the quality of clusters. Each cluster is constructed based on a centroid of cluster and a measured score of each record in the cluster which must fall within a set score criterion. Each record with the cluster is validated against the centroid of every other cluster to ensure the clusters are disjoint. Further the records within each cluster are counted with their corresponding score to ensure majority of the clusters are densely populated.

6 Results

Our experiments consisted of several iterations using different training and test sets with full, one-feature, and restricted feature sets. These iterations consisted of 1K, 750K, and 2M records in combination with three different feature sets. We used 20% of the training data set for validation.

The results are summarized in Tables 4, 5, and 6. The "Model" column reflects the feature sets used for each iteration. The "RecCnt" column lists the number of records used to train the model for each iteration. The "TestRecCnt" column shows the number of test records used for the validation in each iteration. The "TotClusters" column is the resulting cluster count for each iteration.

In our results we recorded multiple factors, first the quality of the clustering algorithm and then the validation of the model. The cluster quality was validated by ensuring the clusters are disjoint and densely populated. As shown in Tables 4, 5, and 6, the "SmallCluCnt" and "CluQuality" columns show our results. The "SmallCluCnt" reflects number of clusters with fewer than 100 records for 2M data set, 40 records for 750K data set, and 10 records for 1K data set.

The "CluQuality" reflects the radius of the cluster within which its members must reside. After several iterations, tuning it to 0.5 produced the best score with disjoint and densely populated clusters. This is further represented in Figure 6.

The "Anom" and "Success" columns reflect the results from the test data set. The "Anom" is a total number of anomalies discovered in the corresponding run and "Success" is the percentage of success over the total number of test records used in each iteration.

Model	RecCnt	TestRecCnt	TotClu	SmallCluCnt	CluQuality	Anom	Success
Full	1000	200	22	6	0.5	5	2.50%
1 Item	1000	200	29	13	0.5	4	100.00%
Restr	1000	200	25	5	0.5	4	2.00%

Table 4: Record 1K

Model	RecCnt	TestRecCnt	TotClu	SmallCluCnt	CluQuality	Anom	Success
1 Item	793000	156693	1386	1192	0.5	156693	100%
Restr	793000	156693	113	48	0.5	1129	0.72%

Table 5: Record 750K

Model	RecCnt	TestRecCnt	TotClu	SmallCluCnt	CluQuality	Anom	Success
1 Item	2051296	410260	4102	3527	0.5	410260	100%
Restr	2051296	410260	127	52	0.5	2523	0.61%

Table 6: Record 2M

As shown in Figure 6 our algorithm constructed disjoint and dense clusters. In the visualization, circles represent clusters with their size directly corresponding to density. The colors were picked randomly. However, the small cluster in "black" color represents the anomalous data.

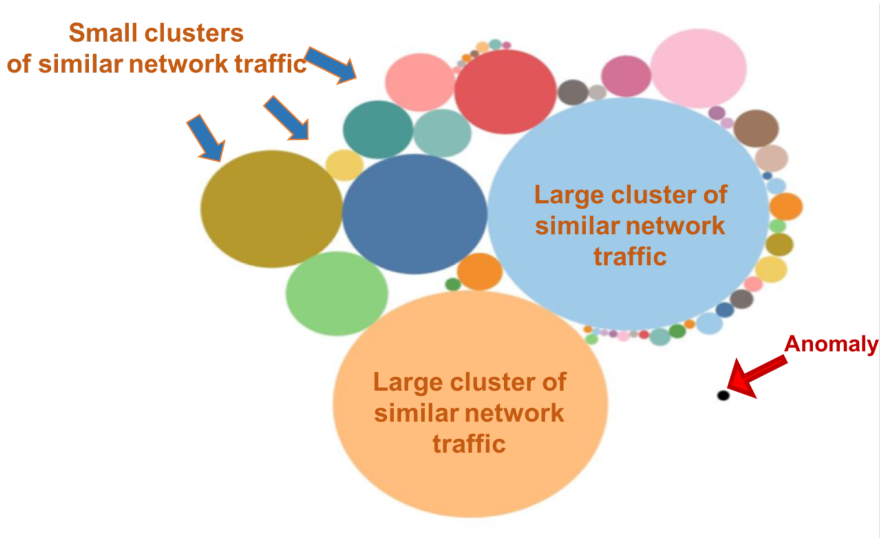

Fig. 6: Clusters of Similar Traffic

7 Analysis

The test data set for each iteration had been instrumented to run against the trained model. For the full and the restricted model the test data set had been mingled to swap certain columns in the expectation that our model will detect these dissimilarities and flag them as anomalous. As a result, varied results were observed from each model with different record sizes. Further analyses showed that one-feature model where the entire test data set consisted of only one attribute, "dstPort", once instrumented, 100% of success rate was observed. This is primarily due to the fact the instrumentation resulted in an invalid "dstPort", which one would expect to be flagged. And rightfully so the results were as expected. In our results one-feature model reflects higher number of total clusters and higher number of smaller clusters in comparison to other models. The Jaccard similarity coefficient method detected lot more dissimilarities amongst the records in the one-feature model.

For restricted models we observed the anomalous results to be consistent when larger data set was used to train the model. The smaller data set of 1K produced over 2% success rate. The Jaccard Similarity Coefficient did find similarities and dissimilarities as a result of our experiments. The results for full and restricted models were, however, under 1%. The variation in both of these results is explained by the size of the data set. The larger the data set, Jaccard Similarity Coefficient found more similarities and fewer dissimilarities amongst the records. In larger data sets with full and restricted model the quality of clusters improved in comparison to smaller data sets.

8 Ethics

The broader topic of Data center security may necessisate ethical considerations. Most of these involve following the structure of the Code of Ethics as it has been applied to Information Technology. This extends ethical parameters such as appropriate access to data and transparency of architecture to those using the service provided in this example. The Code is also used to ensure the privacy of the users and the protection of their data. Companies need to feel the same responsibility in protecting smaller clients as they do larger clients in the event of a data breach.

The provider has certain responsibilities to the users in a case that an attack is successful and the customer data is compromised. In a service-based system this responsibility is split between the provider and the company using the services. Both are responsible for some piece of the security protocol, depending on the extent of services being utilized. Once a server has been breached, they must allocate resources to minimize impact and return the customer to a sense of normalcy and privacy. This ethical responsibility depends on the type of consumer data that is being stored and used. In a case where an attack could be detrimental to either the company or the consumer, is where the presented model is most effective and important.

Data centers have ethical implications that naturally come with the aggregation of large amounts of data. Some are likely to detailed transactional data which can be used to derive personal information of vast amounts. The secure storage and privacy promise to the consumer needs to be done by the data center itself. Employing this model will aid in the protection of information by ensuring once a problem has started, it will be caught and controlled quickly.

This data center security model will be used in the identification phase of the attack. Although this is created to be a line of defense for the customer, the model itself has some ethics to consider. Due to the model monitoring the movement of transactions within the system, the highest-level access will need to be given to certain members. The current model is built using data from a security service company. The company has ethical guidelines pertaining to the use and access to the client data. Most clients likely carry data that is considered personal information for their customers. Because the company that will be running the model has access to some of this data, this adds another layer of Information Technology personnel to be trusted in appropriate data usage. Necessary levels of confidentiality of the data at hand needs to be considered when giving access to employees. Although implementation of this model does not require usage or storage of the client data, it is necessary to consider the same ethical responsibilities that they do.

Another ethical consideration in the use of our model is, in the event of a breach, how much responsibility falls on the owner of the model versus the owners of the data. This needs to be fairly and clearly documented between the parties before the first implementation. Ensuring the guidelines for this scenario are clear and precise will lead to the most effective protection to the final consumer. Since the model can be used for any type of data, the ethical considerations used within the field need to be address in a situation by situation basis. All security services come with high responsibility in the ethics of handling information.

9 Conclusions and Future Areas of Research

In this paper, we introduce our approach to detect anomalies in the data center application traffic using Jaccard Similarity Coefficient and Clustering technique. The traffic behavior is estimated to be very similar. In a data center, the application traffic is expected to be well behaved with large similarities. Our experiments show results, given a finite number of servers and applications, the traffic is expected to have more similarities than dissimilarities. The experiment results show that we can effectively detect anomalies in the data center application traffic. Our experiments resulted in the understanding that larger the data set more effective the training model, and better the anomaly detection for data center application traffic.

Many interesting aspects of our approach still remain to be explored, and comparisons with other available machine learning methods. Further refinement to our approach could be conducted using comparisons to unstructured and unlabeled data used in natural language processing algorithms.

Acknowledgements

We wish to thank Dr. Ratinder Paul Singh Ahuja, CEO of ShieldX Networks Inc. for research suggestions and making the data center application server data sets available to us. Invaluable review feedback from Dr. Daniel Engels helped us improve the work.

References

1. Ying Cao. Osiris: a malware behavior capturing system implemented at virtual machine monitor layer. 2013.
2. Chandrika Palagiri. Network-based intrusion detection using neural networks. *Department of Computer Science Rensselaer Polytechnic Institute Troy, New York*, pages 12180–3590, 2002.
3. Leonid Portnoy, Eleazar Eskin, and Sal Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. Citeseer, 2001.
4. Wenjie Hu, Yihua Liao, and V Rao Vemuri. Robust support vector machines for anomaly detection in computer security. In *ICMLA*, pages 168–174, 2003.
5. Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.