MACHINE LEARNING BASED
IDS LOG ANALYSIS

by

Tianshuai Guan

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Ida **Ngambeki, Chair**

Department of Computer and Information Technology

Dr. Baijian Yang

Department of Computer and Information Technology

Dr. Jin Kocsis

Department of Computer and Information Technology

Approved by:

Dr. John Springer

Dedicated to my beloved family,

Especially my mother Fang, my father Ningqing, and my grandma

ACKNOWLEDGMENTS

This research is for master thesis from the Computer and Information Technology Program at Purdue University, with the help of Prof. Ngambeki, Prof. Yang, and Prof. Kocsis. The copies of references are from Purdue Library.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
NIDS	Network-based Intrusion Detection System
HIDS	Host-based Intrusion Detection System

ABSTRACT

With the rapid development of information technology, network traffic is also increasing dramatically. However, many cyber-attack records are buried in this large amount of network trafficking. Therefore, many Intrusion Detection Systems (IDS) that can extract those malicious activities have been developed. Zeek is one of them, and due to its powerful functions and open-source environment, Zeek has been adapted by many organizations. Information Technology at Purdue (ITaP), which uses Zeek as their IDS, captures netflow logs for all the network activities in the whole campus area but has not delved into effective use of the information. This thesis examines ways to help increase the performance of anomaly detection. As a result, this project intends to combine basic database concepts with several different machine learning algorithms and compare the result from different combinations to better find potential attack activities in log files.

Keywords: IDS, Anomaly Detection, Clustering, Log Analysis

INTRODUCTION

In this day and age, information technology is snowballing. There is a large amount of daily internet traffic, including connecting to websites to find information, connecting to servers to access and transfer information, and other Internet uses. Organizations typically also track the information that flows through their network, including material stored in various logs like IP addresses and SSH requests.

However, with the rise of cyber-attacks, the risk of using the Internet has been rising significantly. The cyber-attacks are an assault launched by cybercriminals using one or more computers against computer networks (Check Point Software, 2020).

There are common kinds of cyber-attacks.

- Malware. Malware is the collective name for a number of malicious software variants, including viruses, ransomware, and spyware (Forcepoint, 2020). It delivers a payload that can range from demanding a ransom to stealing sensitive personal data.
- Phishing. Phishing attacks are the practice of sending fraudulent communications that appear to come from a reputable source. These fraudulent communications are used to introduce malware or direct users to sites where their information can be collected.
- Denial of service attack. It is a kind of attack targeting systems, servers, or networks with traffic to exhaust resources and bandwidth.
- DNS Tunneling. DNS tunneling utilizes the DNS protocol to communicate non-DNS traffic over port 53 (Cisco, 2020). It sends HTTP and other protocol traffic over DNS.

The threat of cyber-attacks is high, and cyber-attacks have become increasingly sophisticated and result in more significant damage. Cyber-attacks result in financial and business losses. A hacker attack can lead to the interruption of business or data breach from a company or organization.

Secondly, cyber-attacks threaten personal security. For example, hackers exploit vulnerabilities to hack into medical records so that they can view patient information.

Finally, cyber-attacks are disruptive to the entire Internet environment. A botnet is a network of devices that have been infected with malicious software, and the attackers can control a botnet without the owners' knowledge to attack more and more servers.

With the increasing frequency of cyber-attacks, companies are paying greater attention to cybersecurity.

Cybersecurity is the protection of internet-connected systems such as hardware, software, and data from cyber-threats. Implementing network security provides a good security situation for computers and networks, and the data stored on these devices to protect them from malicious attackers.

To detect cybersecurity threats, more and more companies and organizations are adapting and implementing at least one Intrusion Detection System (IDS) as a defense mechanism notice the potentially malicious activities in the environments.

An intrusion detection system (IDS) is a software application or hardware appliance that monitors traffic moving on networks and through systems to search for policy violations or suspicious activity and known threats, sending up alerts to an administrator when it finds such items.

IDS can be classified into Network Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (HIDS) by where the detection takes place. They can also be classified into signature-based detection and anomaly-based detection by the detection method.

Zeek, a kind of IDS. It can be used as a network intrusion detection system with additional live analysis of network events, making it a widely used IDS in the cybersecurity industry.

Although many organizations have accepted this system, there are some gaps found in its use. This thesis is going to examine a number of these gaps to make fuller use of this system.

First, as soon as a network interaction is identified by Zeek as "abnormal" or "out of the ordinary," it will be recorded indiscriminately in a log named notice.log generated by Zeek, which means that the records appearing in notice.log are from different kinds of attacks. These may not be clear enough to be analyzed and processed by the data analysts of the company using Zeek.

Second, the existing analysis of Zeek's log files is at a rudimentary stage and does not filter for specific content within the log files, leading to a lack of features of the input data to produce a better model.

Third, based on knowledge about Zeek, connections are also be recorded in other log files (such as HTTP, SSH, and DHCP). Different attributes of the same network connection are stored in different logs. The existing log file analysis only uses one log file as input data, which is a waste of resources.

This thesis's primary goal is to find a new way to use logs from Zeek to identify anomalies in the data that have the potential to be malicious. The following is a list of measures to fill in the corresponding gaps identified above.

When it comes to the first gap, this thesis explores testing various implementations of anomaly detection and clustering algorithms on existing log files to determine the suitability of specific machine learning approaches for recognizing anomalies. Cyber-attacks can be classified into different types after using the clustering algorithm. For the second gap, this thesis tries to select specific fields based on the knowledge about network traffic and cybersecurity from all the fields in the log file as the input of the machine learning algorithm. To fill in the third gap, this thesis explores combining the columns of multiple logs together that include mentions of the same connection and gauge if it is possible to get a better anomaly detection result from a combined set of records.

In this thesis, it is assumed that the outliers found in the experiment are the request records that need to be carefully checked by the relevant personnel, rather than 100% malicious activities.

As for this thesis, the limitation would be the uncertainty caused by the data input, not only the data source, but also the popential “curse of dimentionality” problem. And as for the delimitation, the scope of this research is based on the data from MACCDC 2012 dataset, which means that the result might not be that general.

After applying these new ways to analyze the log files generated by Zeek, the company or organization's managers can clearly understand the current network risks they face and develop more targeted protection measures on their network facilities, leading to a reduction of the risks of network attacks.

LITERATURE REVIEW

Introduction

This section introduces the technologies involved in this project and how they have been combined in previous studies.

First, network traffic is discussed, which is also the raw data collected by the intrusion detection system. Then a detailed introduction of the intrusion detection system is involved, contains types and structures. Then, the intrusion detection system used in this thesis, Zeek, is described in detail. The last section is about the machine learning algorithms used in this thesis, especially the different anomaly detection algorithms.

Network Traffic

Network traffic is the data moving across a network at a given point of time (Lakhina et al., 2004). It consists of packets sent from a source port to a destination port.

Network architecture is separated by seven different layers from the physical layer on the bottom to the application layer on the top based on the OSI (Open Systems Interconnection) model (Briscoe, 2000).

For this project, I examine Zeek, an intrusion detection system. Specifically, the log files based on its analysis mainly from application layers, such as http.log, dns.log, and smtp.log (Forouzan, 2012).

With the development of the Internet, the amount of malicious network traffic is gradually increasing. Malicious network traffic is a kind of traffic with the intent of attack a computer or network. Some of the common malicious network traffics are listed below.

- **Denial of service (DOS) attacks**, which are intended as attempts to stop legitimate users from accessing a specific network resource (Zargar et al., 2013). The performance of the network would be decreased because of the overloading.
- **Botnet attacks**. The botnet is a network of computers infected by malware that is under the control of the attacker. The attacker can make every computer on its botnet simultaneously perform a criminal action to a target network or host. (Hoque et al., 2015).

- **Port scanning**, where the attacker sends packets to a network to detect which ports are open and if vulnerable services are running on the port.

Intrusion Detection System

As the proportion of malicious network traffic in network traffic increases, the number of threats has increased dramatically. Moreover, the attackers are becoming more skilled and successful (Sazzadul Hoque, 2012, p. 117). A study from Reddy (2014) shows that many companies could not withstand large-scale cyber-attacks in the beginning. Traditionally, firewalls are widely used to protect the computer or network. However, with the diversification of network attacks, the firewall cannot meet all the needs of network security, especially with the shortage of monitoring application layer of the OSI model (Kaur, Malhotra, & Singh, 2014).

Under this situation, two kinds of defense methods, IPS (intrusion prevention system) and IDS (intrusion detection system), emerged to monitor and detect the potential attack.
Intrusion detection systems are systems with the purpose of monitoring and analyzing events that may occur on a computer system or network by identifying evidence of possible events that are a violation or of a computer security policy (Aroms, 2012).

Intrusion prevention systems are systems that combine intrusion detection with trying to stop the incidents in real-time.

The main difference between intrusion detection systems and intrusion prevention systems is that intrusion detection systems are only a monitoring system (passive monitoring). In contrast, intrusion prevention systems are control systems (reactive monitoring), which means that intrusion prevention systems will proactively deny network traffic if those packets show a known threat (Trost, 2009). Moreover, Intrusion prevention systems can prevent the attacker from going deeper into the system, just as shown in Figure 1.

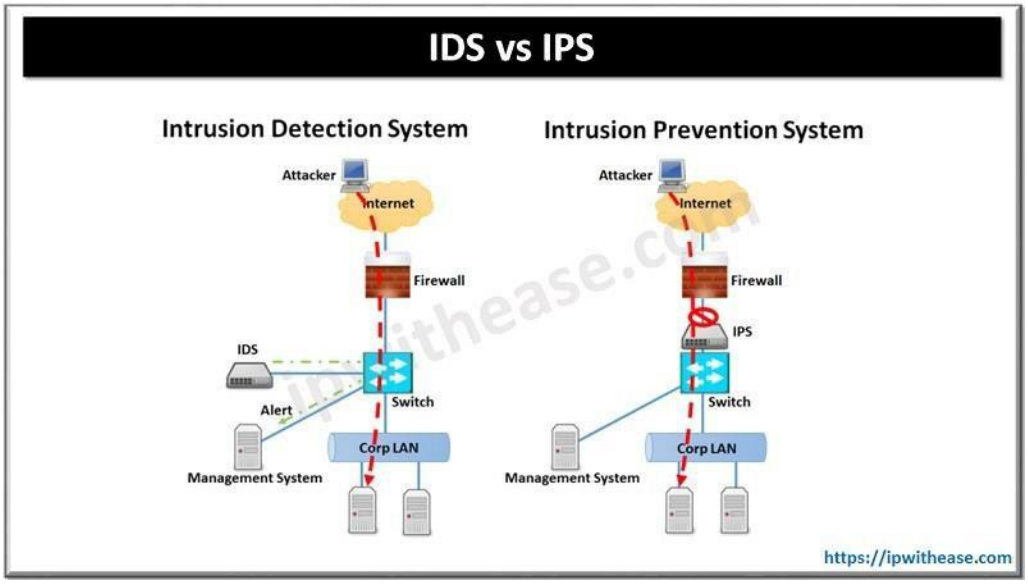


Figure 1 Comparison of IDS and IPS (Bhardwaj, 2020)

Intrusion detection systems, not intrusion prevention systems, are the focus of this thesis. Intrusion detection systems usually contain three logical components (Stallings & Brown, 2017):

- **Sensors.** Sensors are used to collect data that tends to be an intrusion, containing network packets and system call traces. And then, sensors would decode the data to the analyzer.
- **Analyzers.** Analyzers are used to detect whether an intrusion occurred based on the data sent from sensors and take actions immediately like, producing an alert to the whole system if necessary.
- **User Interface.** The user interface can help an administrator have a holistic view of the intrusion detection system and can be used to configure the intrusion detection system for better use.

There are many subclasses of intrusion detection systems, and the most common criteria are the place of deployment of the intrusion detection system. The intrusion detection system can be separated into two different subclasses, the Host-based Intrusion Detection System (HIDS) and the Network-based Intrusion Detection System (NIDS) (Stallings & Brown, 2017).

Host-based Intrusion Detection Systems (HIDS)

A host-based intrusion detection system is initiated and installed at the host-level to monitor the system from internal or external threats. A host-based intrusion detection system usually monitors the activities on a host server, such as an anti-virus program on the local computer.

More specifically, a host-based intrusion detection system can collect data from different sources of the host, like different kinds of system logs, which means when dealing with host-level threats, this system is beneficial.

However, some drawbacks of the host-based intrusion detection system also need to be discussed.

First, host-based intrusion detection systems do not support cross-platform functions or applications.

Second, when other hosts in the same network are attacked, this system cannot help them due to its host-level defense (Ying, Yan, & Yang-jia, 2010).

Third, if the attackers can control the whole host, this host-based intrusion detection system will not work (Berthier, Sanders, & Khurana, 2010). Studies suggest that the IDS should be separate from the host because of the existence of this kind of risk (Crosbie et al., 2006).

Lastly, the large-scale data that needed to be collected by the host-based intrusion detection system would cost a heavy burden to the host because this detection system was run on the host. It would make it hard for the host to keep the effectiveness, and it also needs more space to store the real-time data (Pharate et al., 2015).

Network-based Intrusion Detection System (NIDS)

Different from host-based intrusion detection systems, network-based intrusion detection systems work by monitoring and analyzing the network traffic in real-time (Brackney, 1998). As shown in Figure 2, this system is usually deployed at different levels, making it capable of detecting the transport layer, application layer, and network layer of the OSI model (Stallings & Brown, 2017).

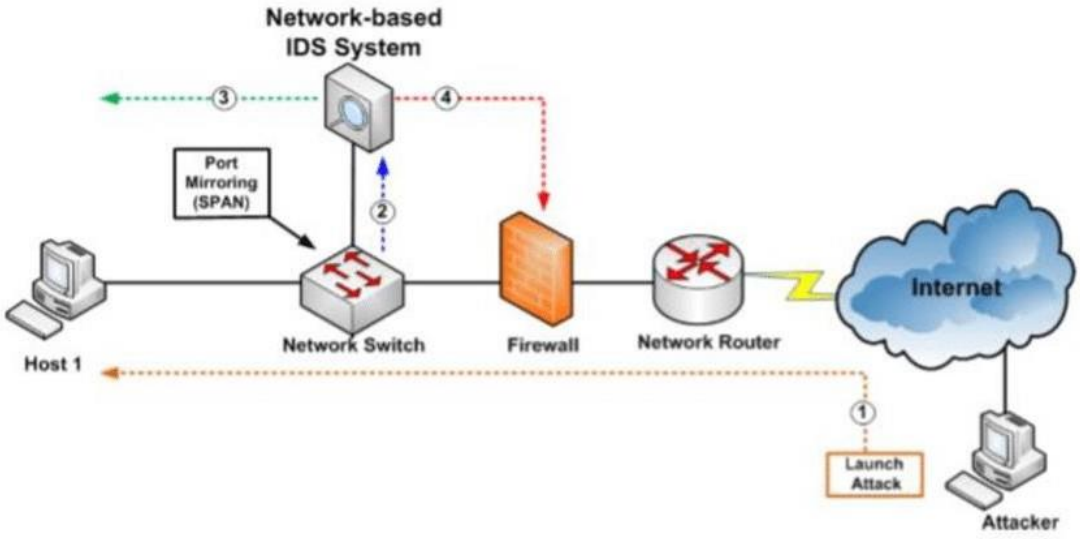


Figure 2 Network-based intrusion detection

This system monitors the network traffic in detail to decide if there was an attack before allowing the traffic to pass. For example, it will inspect the content and header information of the packets (Rights, 2004).

Nowadays, a network-based intrusion detection system is considered as the most widely used defense system in the industry (Shin et al., 2010). It has two benefits as listed below:

- **Portability.** Compared to host-based intrusion detection systems, this system can monitor the network without altering the existing infrastructure, which means that these systems can be easily integrated with the target system or host and are adaptable to a cross-platform environment.
- **Real-time detection.** Network-based intrusion detection system can monitor the network in real-time, which means that they can have a quicker response and may be able to log the evidence that attackers want to erase.

Meanwhile, the drawbacks of the network-based intrusion detection systems cannot be ignored. First, with the dramatic increase of the volume of the network traffic, how to ensure low latency while analyzing network packets is becoming a problem. Second, there are some unconformities between the detection system and the monitored system because of its "portability."

Open source intrusion detection system

Network-based intrusion detection systems are becoming an essential part of many companies' security policies due to the rapid increase in the attacks seen on the Internet.

There are lots of open source network-based intrusion detection systems. Snort is the first wildly deployed detection system in real businesses (Stallings & Brown, 2017). Snort tries to match each packet with a set of defined rules by itself, which means that if it is misconfigured, it may lead to lots of legitimate traffic being blocked or illegal traffic passing the detection system (Roesch, 1999). Another drawback of Snort is that it is not multi-threaded, which means it may not perform well facing the large scale of traffic from a distributed network system.

Zeek (formally known as Bro) gradually became a substitute for Snort for many companies. Zeek was designed and developed by Vern Paxson at Lawrence Berkeley National Lab and the International Computer Science Institute and finished in 1998. Zeek can first parse the network traffic to get the semantics and use its analyzers to compare it with those system-believed "trouble" patterns. If Zeek has found something abnormal, it will generate the log and alert the administrators in real-time.

Zeek's design was guided by the goals listed below.

- Separation of mechanism and policy. The system will become more flexible if it can separate the mechanisms needed to monitor specific policies. For example, while public research institutions can generally provide full Internet access to all internal systems, corporate networks may have to impose strict restrictions (Sommer, 2003).
- Enabled for high-speed and large-volume monitoring. Although Zeek's bandwidths routinely reach one Gbit/s, it still needs to be stable and not miss any traffic packets.
- Withstanding attacks. Network-based intrusion detection systems are usually the first target when hackers attack, which requires that Zeek should not easily be controlled or at least turned off by the attackers (Paxson, 1999).

Based on the design goals, the architecture of Zeek is shown in Figure 3 (Sommer, 2003). The general workflow of this system is as follows. The packet capture layer is responsible for collecting the packets from the network. It will feed them into the event engine layer, where

protocol analysis would be performed. The event engine layer then generates the event to the policy layer, and the policy layer would identify the events based on the existing rules and policies. The lowest layer would process an enormous amount of data. With the increase of the system layers, the amount of data processed becomes less. Moreover, that is why Zeek can handle large-scale volume network traffic.

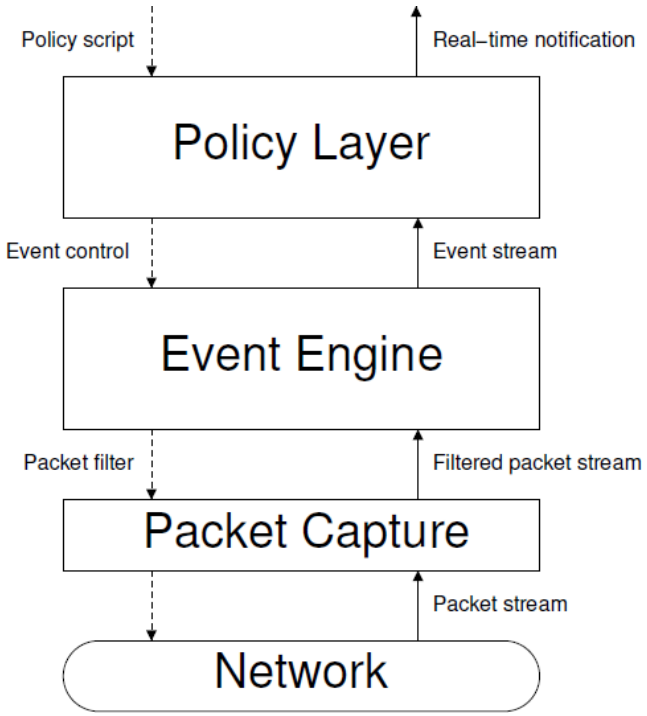


Figure 3 Design of Zeek

Here is a brief description of each layer:

- **Packet capture layer.** Zeek captures network traffic in this layer. In order to reduce the workload and be able to monitor certain traffic packets, Zeek uses library libpcap, which implements filter expressions.
- **Event engine layer.** This layer is the core of the Zeek and consists of some different analyzers based on different protocols. Protocols in the transport layer and application layer of the OSI model can find their corresponding analyzers.
- **Policy layer.** This layer contains the event handler, which accepts scripts written in Zeek's language. Users can customize their handlers to solve the events from the event engine layer.

In conclusion, Zeek is a powerful open-source network-based intrusion detection system. It contains several analyzers for the different protocols. It allows users to define the policies based on the customized scripts written in Zeek's scripting language for the current utilizing environment. Zeek can be deployed in a large-volume traffic environment and has good flexibility, making it popular and successful in business as an intrusion detection system.

Machine Learning and intrusion detection system

Network-based intrusion detection systems are divided into two major subclasses: signature-based network intrusion detection systems and anomaly detection-based network intrusion detection systems.

As for the former, whether traffic is malicious traffic or not is determined by the existing rules or policies in the system (Niyaz et al., 2017). It is a very effective method to classify the traffic, but this signature-based network intrusion detection system would not work when traffic with an unknown pattern comes in. Compared to this kind of passive defense, an anomaly detection-based network intrusion detection system performs better.

Anomaly detection-based network intrusion detection systems classify traffic as an attack when the traffic is largely different from standard patterns. Moreover, this system is more suitable for detecting unknown types of attacks. This system can be separated into two subclasses: supervised learning-based intrusion detection systems and unsupervised learning-based intrusion detection systems. To better understand those two systems, a basic knowledge of machine learning needs to be involved and discussed.

Brief introduction to machine learning algorithms

Machine learning is a system of algorithms that enables computers to learn from the data and improve themselves without being explicitly programmed, which leads to a more natural decision and better result.

Machine learning algorithms are organized into the following taxonomy:

- Supervised Learning algorithms. The algorithms generate a function that maps inputs to desired outputs. Labeled data, a data set that has already been classified, is used to predict the classification of other unlabeled data. Supervised learning

algorithms have many classic ones, such as support vector machine (SVM), decision trees, and naïve Bayes.

- Unsupervised learning algorithms. The inputs are unlabeled and uncategorized, and the system runs the algorithm without prior training. Clustering, segment data into different groups, is often used in unsupervised learning (Hastie, Tibshirani, & Friedman, 2009). The representative algorithms are K-means, t-SNE, and DBSCAN.
- Semi-supervised learning algorithms. This kind of algorithm falls in between the above two algorithms; that is, it uses a limited number of labeled data to train the model to label the unlabeled data. (LU, 2013)
- Reinforcement learning algorithms. This kind of algorithm uses observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk (Fumo, 2018).

Many of the machine learning algorithms can be applied with intrusion detection systems. For example, as for supervised learning, a paper published in 2019 by Liu and Lang from Bei hang University named "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey" is the base of this research project, as it presents a variety of approaches for applying Machine Learning in Intrusion Detection Systems, such as SVM (Support-Vector Machine), KNN (K-Nearest Neighbors), Naive Bayes, and DT (Decision Trees) (Liu, & Lang, 2019). This paper was used as a primary reference for identifying practical algorithms for this paper.

Anomaly detection algorithm

Anomaly detection identifies rare data that raise suspicions by differing significantly from the major ones (Yu et al., 2017). For example, unusual network traffic revealed by the intrusion detection systems log may be classified as an outlier.

In this paper, three different approaches of the anomaly detection algorithms will be introduced: proximity-based anomaly detection, cluster-based anomaly detection, isolation-based anomaly detection.

Proximity-based anomaly detection

Proximity-based anomaly detection identifies data as an outlier when its proximity is sparsely populated. In other words, when the proximity of the object significantly deviates from the proximity of most of the other objects in the same data set, it would be classified as an outlier.

Two major types of proximity-based outlier detection are distance-based anomaly detection and density-based anomaly detection (Aggarwal, 2013). On the one side, those two different approaches are very similar because they are all based on the notion of similarity (proximity). On the other side, those two approaches differ because of different definitions of proximity. A more detailed introduction and explanation are as follows.

Distance-based anomaly detection algorithm

Distance-based methods of anomaly detection are based on the calculation of distances between data of the database. Distance-based anomaly detection algorithms work on assumptions that the typical data have a dense neighborhood, and the outliers are far from their neighbors (Syarif, Prugel-Bennett, & Wills, 2012).

Given a set of points, a point o is called a DB (k, R) distance outlier if there are less than k points within distance R from o (Knox, & Ng, 1998). After that, many variants and updates of distance-based anomaly detection appeared. Research from Ramaswamy (2005) considered a fixed number of outliers present in the dataset. A new method for creating a probability density function over data values emerged (Bench-Capon et al., 1999).

The most popular approach in this field is kNN (k-Nearest Neighbors). K-NN classifies a data point based on how its neighbors are classified. However, k-NN is not limited to predict the result. It can also be used in detecting anomalies.

Since k-NN uses underlying patterns in the data to make predictions, any errors in these predictions are a clear sign that the data points do not match the overall trend (Coomans, & Massart, 1982). In fact, any algorithm that generates a predictive model can be used to detect anomalies with this approach. For example, in regression analysis, outliers can deviate significantly from the line of best fit.

Once anomalies are identified, they can be removed from the dataset used to train the predictive model. This will reduce the noise in the data, thereby enhancing the accuracy of the predictive model (Hautamaki, Karkkainen, & Franti, 2004).

K-NN is a kind of supervised learning algorithm, which means data with the label is needed. However, in this thesis, unsupervised learning is preferred because of the lack of labeled data.

Density-based anomaly detection algorithm

The distance-based outlier detection method calculates the number of the neighborhood data in the dataset, defined by a given radius (Ren et al., 2008). The data in the dataset is then considered an outlier if it does not have enough neighborhood. The distance-based outlier detection method usually assumes that the density around standard data is close to the density around its neighbors. In contrast, the density around an outlier is different from the density around its neighbors (Syarif, Prugel-Bennett, & Wills, 2012).

The most representative one of the density-based anomaly detection algorithms would be LOF. LOF, local outlier factor, is an anomaly detection algorithm that works by measuring the local deviation of the data with respect to its neighbors (Breunig et al., 2000). A density-based approach uses an outlier factor as a measurement of being an outlier. The algorithm calculates an outlier factor LOF for each point in the dataset and determines if it is an outlier by determining if the LOF is close to 1. If the LOF is much greater than 1, the point is considered an outlier, and if it is close to 1, it is a regular point (Gupta et al., 2014).

Cluster-based anomaly detection algorithm

Clustering is a data mining approach that groups data into different clusters with similar data instances in the same cluster. In clustering, the goal is to partition the data instance into several dense subsets (Boukerche, Zheng, & Alfandi, 2020). Models with clustering algorithms are primarily trained with unlabeled data that consists of normal and abnormal network traffic. The clustering-based anomaly detection algorithm is more efficient compared to the proximity-based ones.

The clustering algorithms can be divided into nine categories which contain 26 commonly used ones (as shown in Table 1). However, only those algorithms used in the experiment will be discussed in the thesis.

Table 1 Clustering Algorithms

Category	Typical algorithm
Clustering algorithm based on partition	K-means, K-medoids, PAM, CLARA, CLARANS
Clustering algorithm based on hierarchy	BIRCH, CURE, ROCK, Chameleon
Clustering algorithm based on fuzzy theory	FCM, FCS, MM
Clustering algorithm based on distribution	DBCLASD, GMM
Clustering algorithm based on density	DBSCAN, OPTICS, Mean-shift
Clustering algorithm based on graph theory	CLICK, MST
Clustering algorithm based on grid	STING, CLIQUE
Clustering algorithm based on fractal theory	FC
Clustering algorithm based on model	COBWEB, GMM, SOM, ART

K-means was made famous by a paper by MacQueen (1967). The core idea of K-means is to update the center of the cluster, as well as the center of the data point, by iterative computation until some criteria for convergence is met (Xu & Tian, 2015). Figure 4 (AIHUB, 2020) is shown below to help understand K-means. The procedure of K-means is as below:

1) Randomly select K cluster centers.
2) Calculate the distance between each data and cluster centers.
3) Assign the data point to the cluster with the nearest cluster center.
4) Recalculate the new cluster center.
5) Recalculate the distance between each data and new obtained cluster centers.
6) Repeat until convergence is met

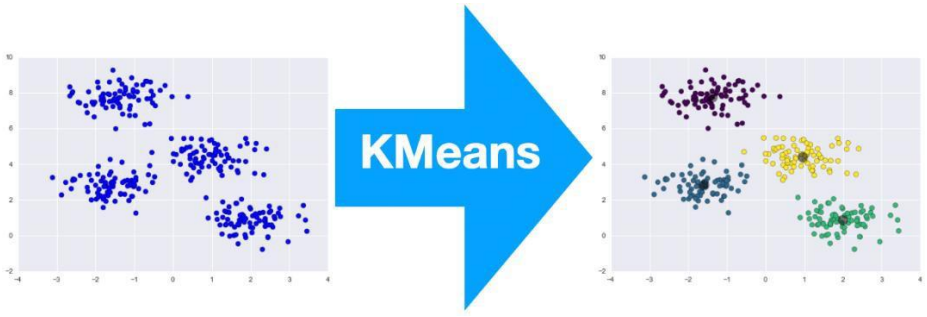


Figure 4 K-means

After K-means was invented, many different researchers tried to update the K- means algorithm. For example, the Muda and Yassin research team published a paper in 2011 where they applied two learning approaches, K-Means Clustering and a Naïve Bayes classifier called KMNB (Muda et al., 2011). K-Means was the first stage that identified similarities among groups, then passed the grouped data to the next stage, a Naïve Bayes classifier that could double-check if the data were in the correct group. By using this two-step KMNB algorithm, the accuracy, detection, and false alarm rates of a single Naïve Bayes classifier had a significant improvement, which this project adopts by similarly using a two-step process for identifying anomalies.

Another example of the K-means variant is that K-means is used to be combined with a minimum spanning tree on the center (Jiang et al., 2001). Moreover, some research about Cluster-based Local Outlier Factor is done by combing the clustering algorithm with LOF (He et al., 2003).

The density-based clustering algorithm, DBSCAN (Density-based spatial clustering of applications with noise), is another algorithm that needs to be discussed in this thesis. The idea of this algorithm is that a region with a high enough density of the data is considered to belong to the same cluster (Kriegel et al.,2011). In other words, DBSCAN groups together point that are close to each other based on a distance and a minimum number of points, and the points in low-density regions are classified as outliers.

The DBSCAN algorithm repeats the following process until all points have been assigned to a cluster or are labeled as visited (Birant & Kut, 2007):

1) Arbitrarily select a point N.
2) Retrieve all points directly density-reachable from N with respect to radius.
3) If N is a core point, a cluster is formed. Find recursively all its density connected points and assign them to the same cluster as N.
4) If N is not a core point, DBSCAN iterates through the remaining unvisited points in the dataset.

Li and Chen's research team published their result of using an enhanced DBSCAN algorithm for anomaly detection in 2011 (Chen, & Li, 2011). By comparing their enhanced DBSCAN algorithm with a supervised isolation forest, they found that the accuracy rate, when

working with DARPA data sets, could be improved significantly by using DBSCAN as an intrusion detection system, which was also noted to have high speed for processing information.

In addition to K-means and DBSCAN, there are many different clustering algorithms, such as Mean-shift (Cheng, 1995), Gaussian mixtures (Bouman et al.,1997), and Hierarchical clustering (Khan & Thuraisingham, 2007). Those algorithms also have developed a lot, but they will not be discussed in this thesis because they are not covered in the experiment.

Isolation-based anomaly detection algorithm

Isolation forest is a kind of machine learning algorithm for anomaly detection (Liu, Ting, & Zhou, 2008). It can identify the anomaly by isolating outliers in the data without relying on any distance or density measure.

The term isolation means separating an instance from the rest of the instances. It isolates instances by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that feature. The split depends on how long it takes to separate the points.

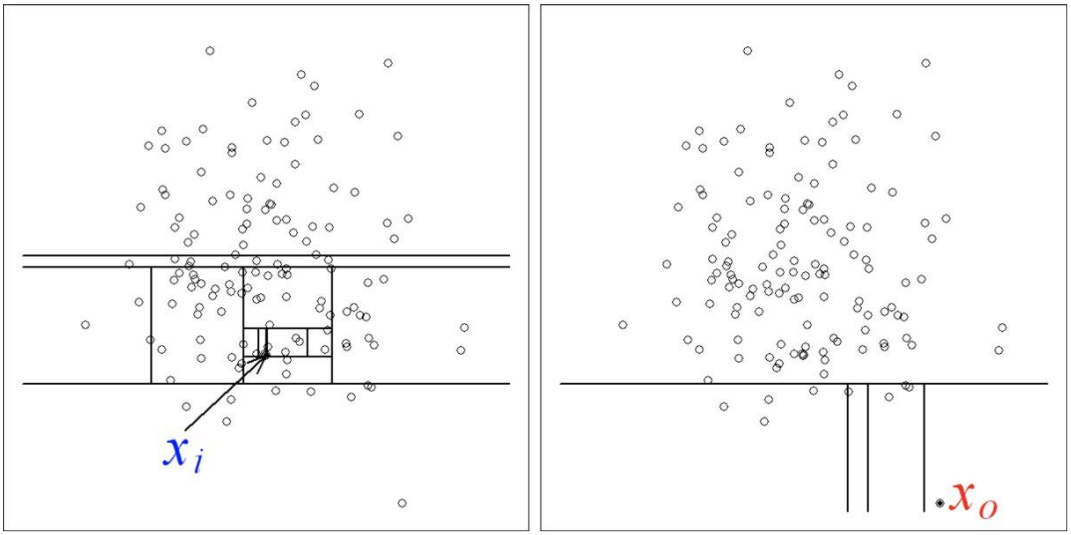


Figure 5 Isolation Forest

Figure 5 will help to illustrate this idea of isolation forest. In the right subfigure, it is evident that X0 is easy to be separated from other instances. It is possible to build a square area

with the only X0 inside within four steps (4 lines), which means that X0 has a high possibility of being an outlier because it is apparently different from other instances in the dataset.

However, as for the left subfigure, Xi is not that easy to be separated from other instances although after 11 tries, which means that Xi is more like a normal instance.

In 2017, a research team led by Marteau from Université de Bretagne-Sud introduced the Hybrid Isolation Forest algorithm, which combines supervised, semi-supervised, and unsupervised techniques (Marteau, Soheily-Khah, & Béchet, 2017). The authors claim that traditional isolation forest algorithms have a drawback that limits the scope of identifying anomalies, which gets affected by blind spots. The hybrid algorithm utilizes the unsupervised paradigm from an isolation forest algorithm as well as a supervised capability from supervised learning to get a better result.

Dimension reduction algorithm

The curse of dimensionality, an exponential increase in the size of data caused by a large number of dimensions, is becoming a big question with the rapid increase of the scale of the data.

PCA, Principal Component Analysis, is a popular way to make dimension reductions. It performs a linear mapping of the data to a lower-dimensional space. More specifically, it finds a low dimensional subspace to project the data to minimize the square of projection error and minimize the square of the distance between each point and its projection point (Walpita, 2020). It is wildly used as a dimension reduction algorithm.

Gap

Many fields are still not explored by the researchers, such as using machine learning algorithms to provide a better result to the administrator or the data analyst in the organization.

It is evident that intrusion detection systems are currently combined with many ways to identify abnormal traffic. However, there is no research to evaluate the log file generated by the intrusion detection systems. Furthermore, the Zeek system puts every weird log into a specific log file without any classification.

For a large organization, merely using an anomaly detection system to defend against every attack is not enough. The log files generated by anomaly detection systems are more like real-time information, so analyzing different types of attacks through logs, administrators, or system operations engineers can better understand where the system is vulnerable and repair it accordingly.

At the same time, the results of the log analysis can be a double-check for the anomaly detection system results to make sure that the system is running correctly.

Aims

Since no research has been found on the analysis of log files for intrusion detection systems, it is hoped that machine learning algorithms, especially anomaly detection algorithms, can be applied to log analysis in an attempt to classify anomalies by unsupervised algorithms. A comparison of the performance of two different machine learning algorithms applied to the log file will also be involved in this thesis.

Since the Zeek system generates multiple log files, this thesis also hopes to creatively combine SQL knowledge and use machine learning algorithms to analyze multiple log files as the source data to get better results than a single log file.

METHOD

In this section, the framework of the experiment is described as well as how to implement it in detail. This section will discuss each part of the framework in the order in which it is presented, which is also the order in which the experiment of the thesis is conducted.

Hypotheses

Based on the aims outlined and gaps found in the last section, for this experiment, there are two hypotheses:

H1: Combining two different log files as input will produce a better performance than using a single log file as input.

H2: Combining three different log files as input will lead to a better performance than merely using two or single log file as input.

Framework

The framework of the experiment section is shown in Figure 1 below, and each part of the framework will be discussed in detail.

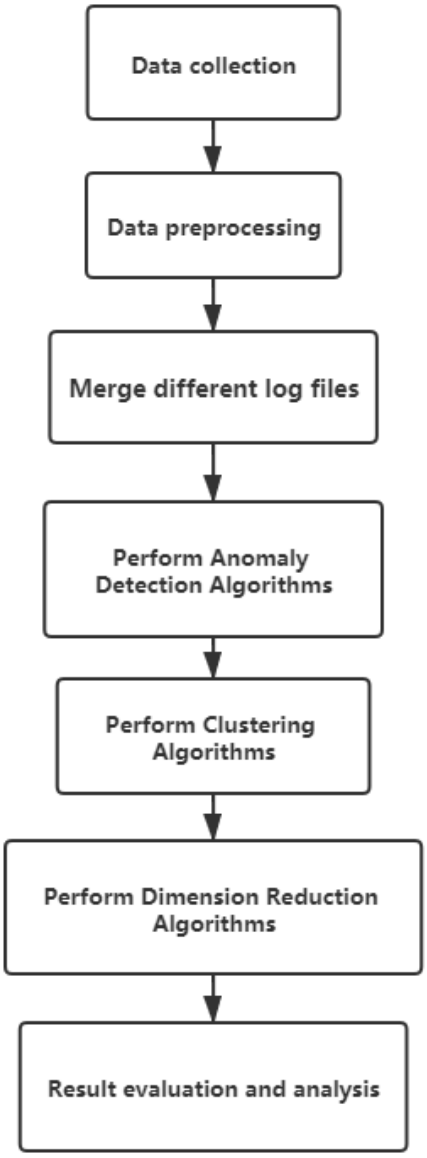


Figure 6 Framework

Data preparation

Data collection

A dataset, MACCDC 2012 (Mid-Atlantic Collegiate Cyber Defense Competition), was chosen for this experiment. MACCDC provides students with cyber defense experience in real life.

MACCDC 2012 is the network traffic data collected from a medical cybersecurity scenario, which happened when the computer system of the hospital was under cybersecurity attacks. After that, several different log files of the intrusion detection system of the hospital were generated, and six of them were relevant to the experiment. Furthermore, the raw data of this experiment will be selected and gathered from those five log files.

- Conn.log: contains connection identified by Zeek, the intrusion detection system. This log file contains complete records of network activities.
- HTTP.log: contains the result from the Zeek HTTP protocol analyzer.
- Notice.log: a file containing connections that are non-conformant with standard protocols. Some connections that cannot be analyzed by Zeek will also be recorded in this log file.
- DNS.log: contains information for connections found in DNS-related protocols.
- Weird.log: the weird.log records unusual or exceptional activity that might indicate malformed connections, traffic that does not conform to a particular protocol, malfunctioning or misconfigured hardware, or even an attacker attempting to avoid/confuse a sensor.

Three of the five log files mentioned above, i.e., Conn.log, HTTP.log, DNS.log, have too many fields, so they are first filtered to remove unimportant fields for the machine learning algorithms. The details of these three files are as below.

Conn.log

It has 24 fields, including source IP, source port, destination IP, destination port, and transfer duration. After identifying valuable and relevant information, eight fields were selected from them as features. An overview of the selection of fields in conn.log can be found in Figure 7, and the details of the fields can be found in Figure 8 below.

_node_name	ts	uid	id.orig_h
id.orig_p	id.resp_h	id.resp_p	proto
service	duration	orig_bytes	resp_bytes
conn_state	local_orig	local_resp	missed_bytes
history	orig_pkts	orig_ip_bytes	resp_pkts
resp_ip_bytes	tunnel_parents	orig_l2_addr	resp_l2_addr

Figure 7 Overview of Conn.log

uid	A unique identifier of the connection.
id.orig_h	Source IP
id.orig_p	Source port
id.resp_h	Destination IP
id.resp_p	Destination port
duration	How long the connection lasted.
orig_pkts	Number of packets that the originator sent.
resp_pkts	Number of packets that the responder sent.

Figure 8 Details of the fields in Conn.log

HTTP.log

This log provides details of each HTTP request and response by Zeek. It has 30 fields in total, and 4 of them were selected based on our knowledge about network traffic. An overview of the selection of fields of HTTP.log can be found in Figure 9, and the details of the four fields can be found in Figure 10.

_node_name	ts	uid	id.orig_h	id.orig_p
id.resp_h	id.resp_p	trans_depth	method	host
uri	referrer	version	user_agent	request_body_len
response_body_len	status_code	status_msg	info_code	info_msg
tags	username	password	proxied	orig_fuids
orig_filenames	orig_mime_types	resp_fluids	resp_filenames	resp_mime_types

Figure 9 Overview of HTTP.log

uid	A unique identifier of the connection.
method	Verb used in the HTTP request (GET, POST, HEAD, etc.).
request_body_len	Actual uncompressed content size of the data transferred from the client.
resp_mime_types	An ordered vector of mime types.

Figure 10 Details of the fields in HTTP.log

DNS.log

This log has 24 fields, and 7 of them have been selected in this thesis. The overview and the detail of this log file are as below.

ts	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p
proto	trans_id	rtt	query	qclass	qclass_name
qtype	qtype_name	rcode	rcode_name	AA	TC
RD	RA	Z	answers	TTLs	rejected

Figure 11 Overview of DNS.log

uid	A unique identifier of the connection over which DNS messages are being transferred.
proto	The transport layer protocol of the connection
AA	The Authoritative Answer bit for response messages specifies that the responding name server is an authority for the domain name in the question section.
TC	The Truncation bit specifies that the message was truncated.
RD	The Recursion Desired bit in a request message indicates that the client wants recursive service for this query.
RA	The Recursion Available bit in a response message indicates that the name server supports recursive queries.
rejected	The DNS query was rejected by the server.

Figure 12 Details of the fields in DNS.log

Data pre-processing

Handling large amounts of data requires pre-processing to reduce the time and other required resources to obtain meaningful results. Data pre-processing is a crucial step to enhance the quality of data to get meaningful insights.

There are many different procedures to handle raw data nowadays. When it comes to this experiment section, the data pre-processing contains the following steps.

Data cleaning

Data cleaning can be used to solve this problem for those data that have many irrelevant or missing fields.

As for missing data, there are usually two approaches to handle this problem.

The first solution is filling the missing value. If a field can be generated automatically or have an obvious answer, it is possible to fill in the missing value to complete the data. The second solution is deleting the corresponding fields of the data. This approach usually is used when the fields are irrelevant or unimportant from this experiment.

Another common problem is the duplicate data in the dataset. The usual way to handle it is to delete those duplicated rows.

The last point is that each of the log files contains at least ten different fields. It is essential to select the relevant fields from the raw data in order to reduce the dimension of the data. The details of the field selection will be discussed in the following experiment section.

Data transformation

The data transformation is to transform the data into appropriate forms for the later data mining process.

As for the categorical data, it is essential to encode them into numerical data because the machine learning algorithms are based on mathematical calculations, and the categorical data in the dataset will lead to unforeseen issues to the final result.

For those numerical data, scaling and normalization are often used to make the raw data better for data mining. Scaling is to transform the data into a specific range, such as 0-100.

Moreover, normalization is used to change the observations so that they can be distributed normally.

Merge Log file in a SQL way

Based on general knowledge of networking, connections should be recorded not only in the Conn.log but also in other log files (such as HTTP, SSH, and DHCP) by Zeek. Every log file has different fields.

Combining the columns of multiple logs that include mentions of the same connection and gauge if it is possible to get a better anomaly detection result from a combined set of records is a good idea. To test this theory, several columns from the conn.log and another log file from the same hour of a day were combined using an SQL left join to produce a brand- new log file named Merged.log.

This experiment will try different combinations of two log files as the input of the machine learning model. For example, the input, Merged.log, can be the combination of Conn.log and HTTP.log or Conn.log and DNS.log.

To investigate whether combining more files gives better results, the experiment will also combine three files as input.

After several tries, it is possible to compare the result from different inputs to see if some undiscovered patterns show up. Moreover, it is also accessible to compare the result from Merged.log and merely Conn.log to see if the performance is better with more log files involved.

Perform Machine Learning Algorithms

Anomaly detection algorithm

Anomaly detection is a technique used to identify unusual patterns that do not conform to expected behavior, called outliers. One of its applications is identifying strange patterns in network traffic that could signal malicious activity.

In the literature review section, this thesis discusses several anomaly detection algorithms that are widely used. The Isolation forest algorithm was selected for the experiment after comparison.

An isolation forest is a type of anomaly detection algorithm that works on the principle of isolating anomalies. Anomalies in a large-scale dataset may follow a highly complex pattern. This is why the isolation forest is very suitable for detecting anomalies. An isolation forest builds an ensemble of iTrees for the data set, and anomalies are the points that have the shortest average path lengths on the iTrees. A detailed explanation can be found in the literature review section.

After applying the isolation forest algorithm, the outliers will be discovered and collected from the whole dataset. Then it is time to separate these outliers into different clusters.

Clustering algorithm

Identifying similar characteristics between various outliers was one of the potential methods to identify if it is possible to classify various types of anomalies to detect particular attacks. In order to do this, two different clustering algorithms in this experiment were used to separate these outliers into different clusters.

The first clustering algorithm is K-Means. K-means algorithm identifies k number of centroids and then allocates every data point to the nearest cluster while keeping the centroids as small as possible.

K-Means looks straightforward but has two requirements.

- Provide the intended number of clusters (K) manually.
- Make sure all the provided data to K-Means is numerical.

For the first requirement, the elbow method is a common way to solve this problem. The Elbow method can select the optimal number of clusters by fitting the model with a range of values for K. If the line chart resembles an arm, then the "elbow" (the point of inflection on the curve) is a good indication that the underlying model fits best at that point. Figure 13 is an example of the elbow method, which indicates the number of clusters should be 3.

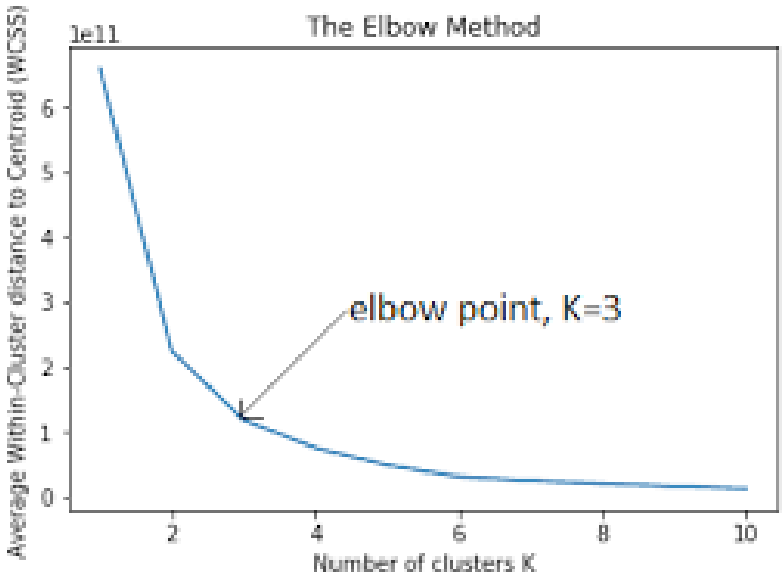


Figure 13 Elbow Method

Then data pre-processing was used to meet the second requirement. A short explanation would be dropping duplicate entries and re-order the entries, removing specific columns, and changing text data columns to numerical data. These processes can be implemented in python, and a detailed explanation can be found in the data pre-processing section above.

The second clustering algorithm used is DBSCAN. The central concept of the DBSCAN algorithm is to locate regions of high density that are separated from one another by regions of low density. The reason to use DBSCAN is that it can automatically determine the appropriate number of clusters. It is proceeded in this case to compare the results from the two algorithms to identify which one performs better.

After applying the clustering algorithm, the outliers can be separated into different clusters. Then the next step is trying to visualize them and do some analysis based on the characteristics of each cluster.

Dimension reduction algorithm and visualization

To help better understand the results of the clustering algorithms, PCA and t-SNE are used to reduce the dimensions of the feature space by feature elimination and feature extraction. It also helps the visualization of data results and gives a more intuitive feeling of the results.

Result evaluation and analysis

The first evaluation criteria are the number of outliers. The total number of outliers identified by the experiment through different approaches is recorded in a table to compare which kind of approach is better.

The number of these outliers is compared to the number of records in notice.log and weird.log, including network interactions identified by Zeek as "abnormal" or "out of the ordinary," and assumed them as ground-truth for comparison's sake. Usually, the larger the number, the better the performance of the approach.

Another evaluation criterion is based on finding which approach can generate a more explainable clustering result. Examining the specific data of the results and the visualization of the clustering are also needed to find out whether the classification criteria are interpretable. A good clustering result usually represents a relatively clear demarcation between clusters as well as the substantial different looks when checking the connections in detail.

Summary

Surveying multiple combinations of machine learning approaches to handling big data increases the potential for obtaining useful information from unlabeled data. Various implementations have particular strengths and weaknesses discovered through background research and implementation. Pre-processing big data is an important technique to identify the relevant data and remove unnecessary information to reduce the resources and time necessary for manipulating the data to a workable state. Using principal component analysis (PCA) also limits the working dimensions, further reducing calculation time.

EXPERIMENT AND RESULT

This section will discuss the experiment part of this thesis, followed by the last method section. Different combinations of the log file as input will be shown to verify whether the hypotheses are correct.

To better test whether the hypotheses are correct, this thesis will conduct four groups of experiments and draw conclusions by comparing their results.

Each group of experiments uses two clustering algorithms, K-means and DBSCAN, and then two dimensionality reduction algorithms, PCA and t-SNE, are used to visualize the results. Each group differs only in the input data. The detail of the input data can be found in the last section.

For group A, this group is the most common way that analysts currently use, i.e., the input data is only from Conn.log. This means that group A will be used as a control group.

For group B, the input data is conn-http.log, which is the new set of data generated by the SQL merge of Conn.log and HTTP.log.

For group C, this group's input data is conn-dns.log, which is a new data set generated by combining Conn.log and DNS.log through SQL.

For group D, the input data is combined with Conn.log, HTTP.log, and DNS.log through left join by SQL.

The result of each group will be introduced in this section. A more detailed analysis and comparison among groups will be discussed in the next section.

Group A

The input of this group is Conn.log. Before using K-means as the clustering algorithm, it is essential to know the number of clusters. Furthermore, according to the result of the Elbow Method, it is 4, as shown in Figure 14.

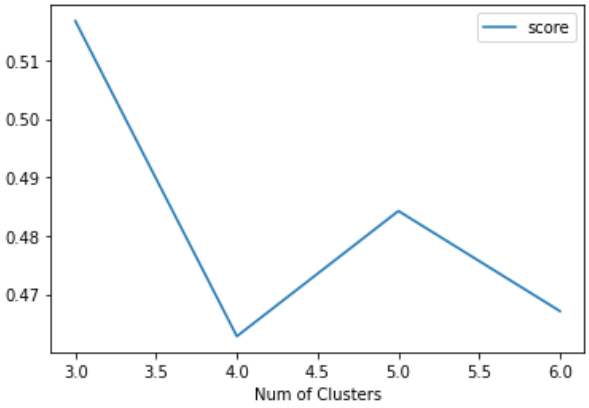


Figure 14 Num of Clusters of Group A

The distribution of the result is as below.

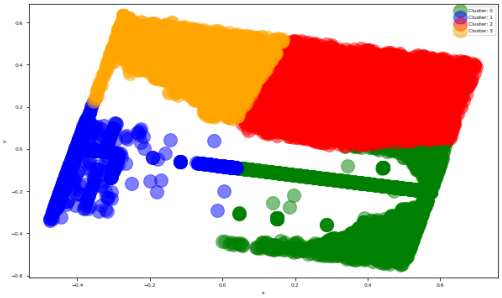
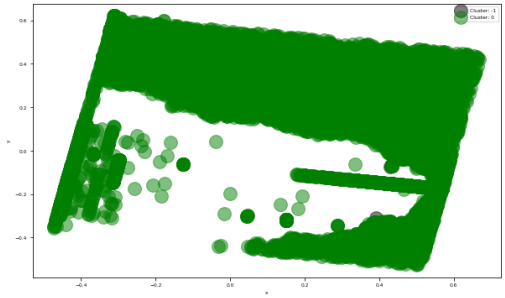
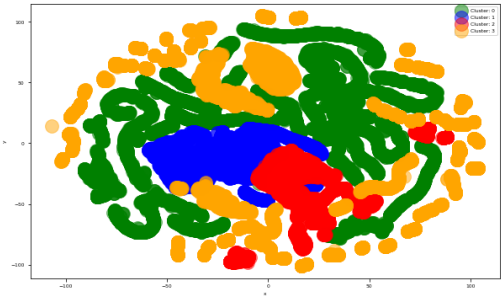
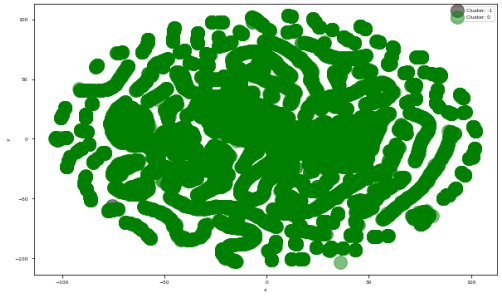
	K-means	**DBSCAN**
PCA		
t-SNE		

Figure 15 Visualization of Group A

Group A makes it reasonable to believe that none of the four sub-figures above shows a clear sign to find the outliers.

The total number of outliers identified by the experimental approach is seen in Table 2, and DBSCAN performs better than K-means as a clustering algorithm in Group A.

For DBSCAN, it shows that Group A has a total of 43 outliers. After compared these outliers to records in notice.log and weird.log, which include network interactions identified by Zeek as "abnormal" or "out of the ordinary," it can be found that the number of overlapping outliers is 25.

Table 2 Num of Outliers of Group A

	K-means	DBSCAN
Outlier detected	36	43
Overlapping outliers	20	25

Group B

As for Group B, the input is the merged log file from Conn.log and HTTP.log. Moreover, the number of clusters can be found in the figure below.

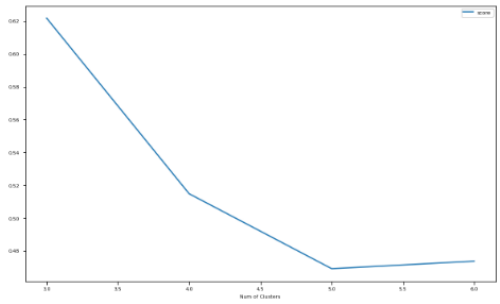


Figure 16 Num of Clusters of Group B

The visualization result is as below.

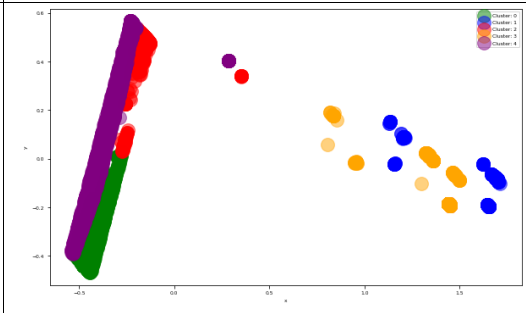
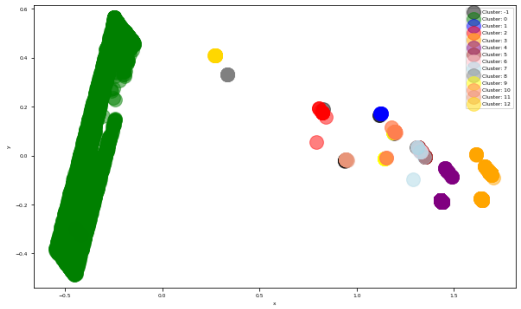
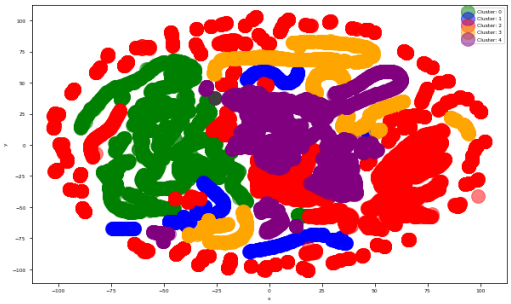
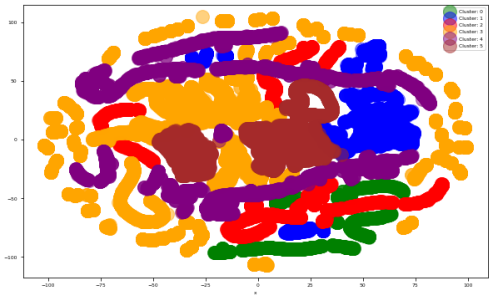
	K-means	DBSCAN
PCA		
t-SNE		

Figure 17 Visualization of Group B

After Figure 18 above, it can be obtained that PCA performs better than t-SNE, and DBSCAN performs better than K-means. Moreover, the top right sub-figure shows the best result among all the four sub-figures because there is a mass of green points in the left part while a small number of outliers in the right part in this sub-figure.

The number of outliers is shown in Table 3 below. In this group, DBSCAN still performs better than K-means in detecting 70 outliers and 43 overlapped outliers.

Table 3 Num of Outliers of Group B

	K-means	DBSCAN
Outlier detected	66	70
Overlapping outliers	37	43

Group C

As for Group C, the input is the merged log file from Conn.log and DNS.log. Figure 20 below shows that the number of clusters for K-means is 5.

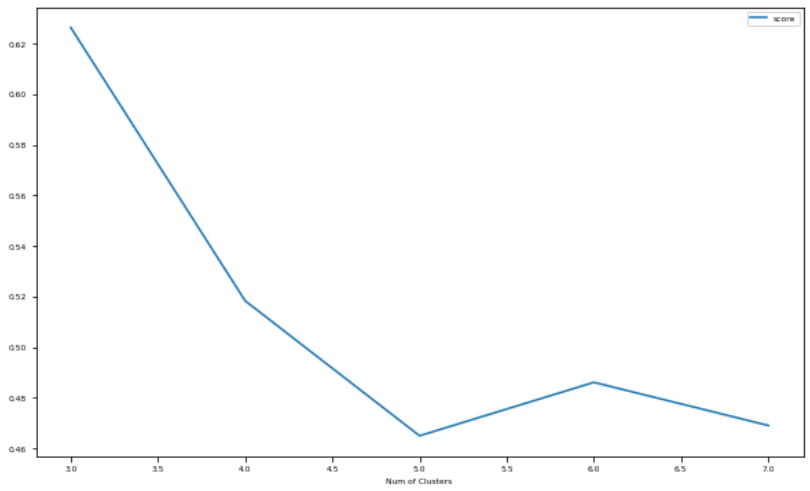


Figure 18 Num of Clusters of Group C

The visualization of the cluster is as below.

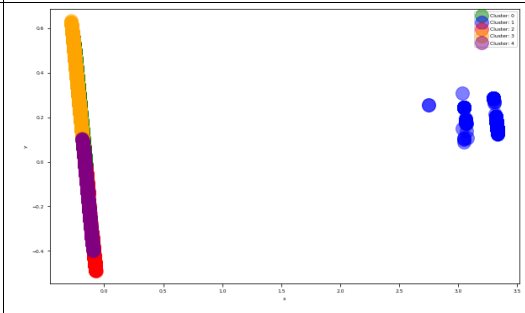
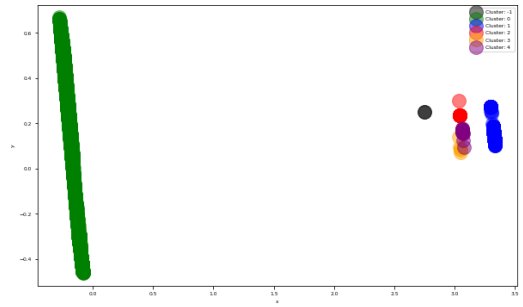
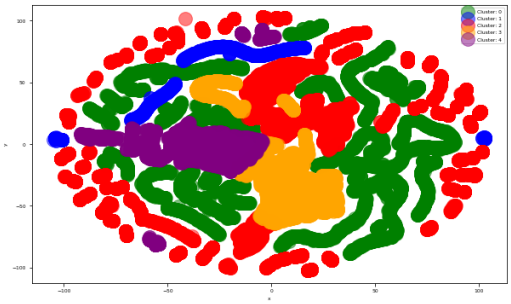
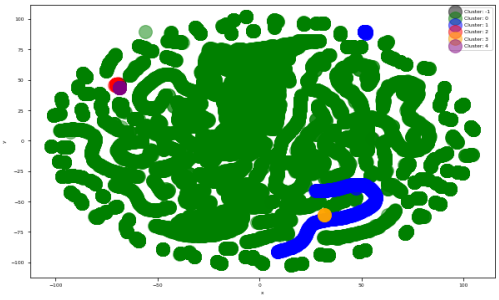
	K-means	DBSCAN
PCA		
t-SNE		

Figure 19 Visualization of Group C

After seeing the above Figure 19, the second sub-figure and the top right one show the best answer because it is obvious to identify the outliers and the "ordinary points."

The number of outliers is shown in Table 4 below. In this group, DBSCAN detected 67 outliers and 49 overlapped outliers, while K-means perform as well as DBSCAN.

Table 4 Num of Outliers of Group C

	K-means	DBSCAN
Outlier detected	66	67
Overlapping outliers	44	49

Group D

For group D, the input is a file merged from Conn.log, HTTP.log, and DNS.log. Figure 20 below shows that the number of clusters for K-means is 6.

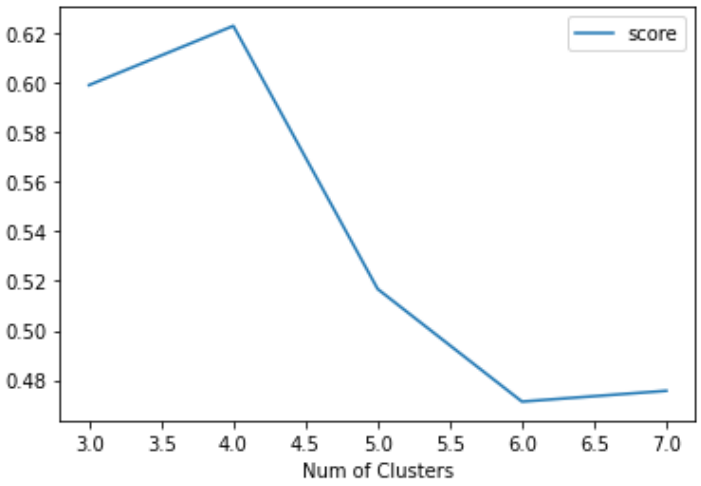


Figure 20 Num of Clusters of Group D

The visualization of the cluster is as below.

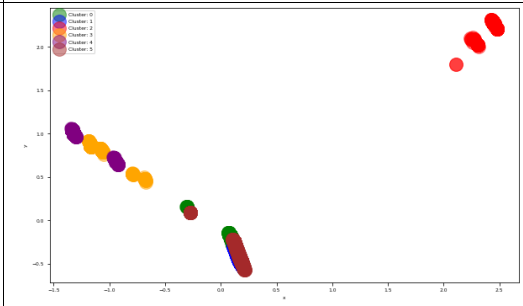
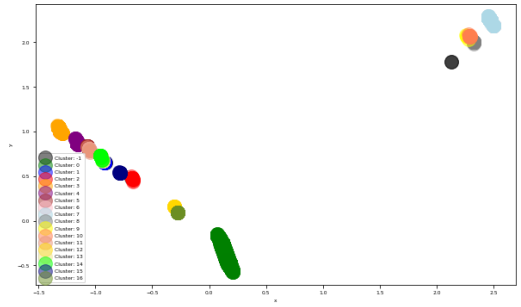
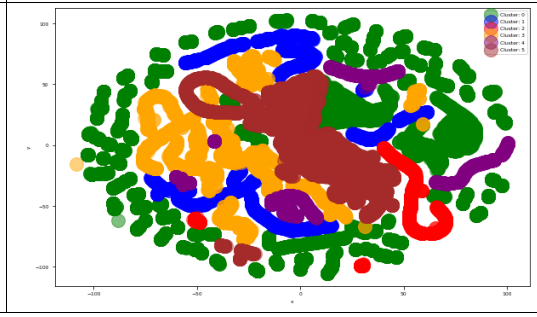
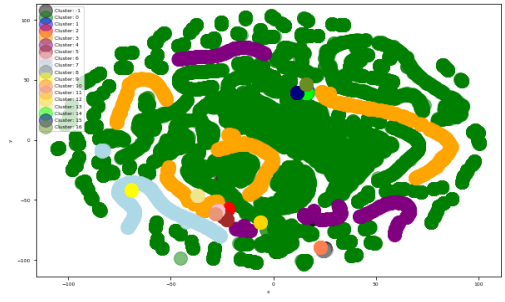
	K-means	**DBSCAN**
PCA		
t-SNE		

Figure 21 Visualization of Group D

After seeing the above Figure 21, for all four sub-figures, the normal points and outlier points are not well distinguished.

The number of outliers is shown in Table 5 below. In this group, DBSCAN detected 85 outliers and 28 overlapped outliers while K-means performs almost the same.

Table 5 Num of Outliers of Group D

	K-means	DBSCAN
Outlier detected	80	85
Overlapping outliers	25	28

Summary

The result of each group was shown and illustrated in this section. The analysis of the result will be discussed in the next section to help draw the conclusion of this thesis.

DISCUSSION AND CONCLUSION

In this section, the analysis of the result from each group will be discussed. The conclusions for the previously proposed hypotheses will also be reached through the analysis. The future work will be illustrated in the end.

Comparison of the results between the groups

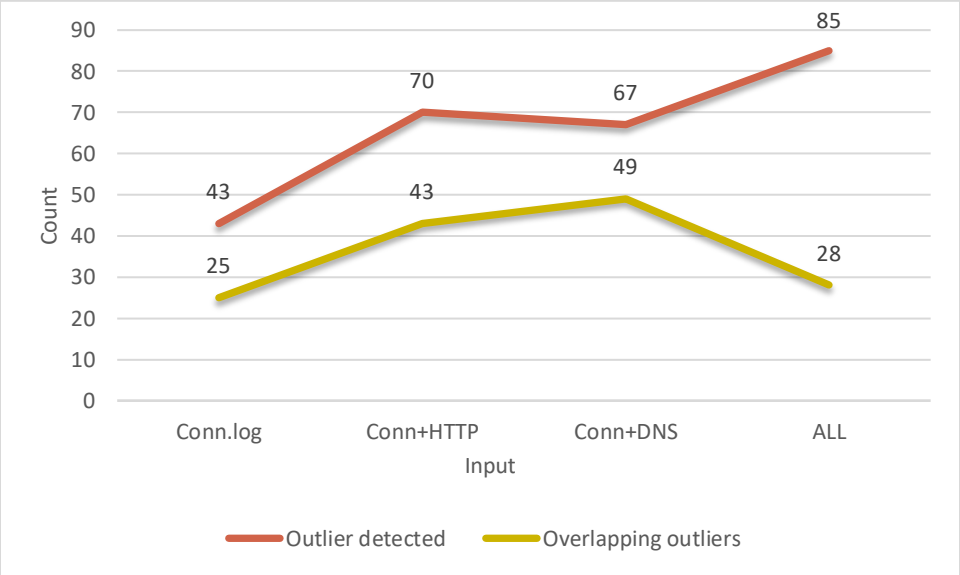
The following table shows that the number of outliers using DBSCAN as the clustering algorithm in each group is positively correlated with those using K-means. Therefore, in the later analysis, to better show the different results of different groups due to different data input, only the results of DBSCAN are used.

Table 6 Comparison of the number of outliers

		K-means	DBSCAN
Outlier detected	Conn	36	43
	Conn+HTTP	66	70
	Conn+DNS	66	67
	ALL	80	85
Overlapping outliers	Conn	20	25
	Conn+HTTP	66	67
	Conn+DNS	44	49
	ALL	25	28

When the clustering algorithm is limited to DBSCAN, the results of four groups are selected to draw a line chart as below.

Table 7 Comparison of outliers between groups



When two log files are merged and then used as data input, such as Conn + HTTP.log or Conn + Dns.log, the number of outliers can be found increased clearly. Both from Conn.log to Conn+HTTP.log and from Conn to Conn+DNS.log show this excellent momentum. As for the reason why this could happen, it is because the second log file brings more features to the original log file, which means that it might be easier for the clustering algorithm to identify those outliers based on more information.

When the above problems are solved, a new problem will naturally arise: selecting another log file for the original log file so that the result can be better.

In this thesis, HTTP.log and DNS.log were selected as the "extended log file" for Conn.log. The principal reason to select these two files is that the two files record the most network connection requests except Conn.log. This means that after merging, as many records originally belong to Conn.log will be added with new fields, there will be enough influence on the clustering algorithm to produce better results.

Another interesting finding is that if the data type of the selected field is Boolean, it can often produce better results than those fields whose contents are merely Strings. That is because clustering algorithms are easier to distinguish "True" from "False."

After understanding the above two rules, why the number of Group C's overlapping outliers is better than that of Group B can also be explained clearly. That is because the DNS.log

file in group C records more network requests, and more Boolean type fields are chosen as features.

What is more, As can be seen from the comparison in the figure below, the outlier distribution of group C is more evident than that of group B. And Group C has more overlapping outliers than Group B, so it is reasonable to believe that the clearer the boundary between clusters, the more overlapping outliers.

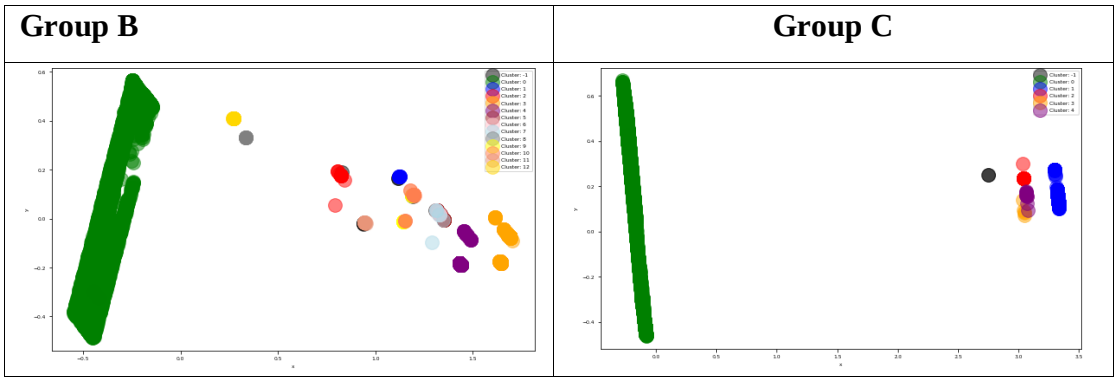


Figure 22 Comparison between Group B and C

When looking at the number of outliers from Group D, one finds that Group D detects more outliers than the other three groups. However, the number of overlapping outliers is lower than Group B and Group C, and even not much different from Group A.

As for the reason why Group D can find more outliers, that is because the input file may have involved too many fields, which causes a heavy burden to the clustering algorithm to make it impossible to identify the outliers. Although I have tried my best to do the data preprocessing, it seems like there is still a long way to go. What is more, the distribution of clusters in Figure 21 also did not show a clear boundary of the cluster.

Conclusion

The two hypotheses made before in this paper are obtained through the analysis and comparison of the above results.

For the first hypothesis: combining two different log files as input will produce a better performance. The result is true. However, how to choose an ideal "extend log file" and how to do

proper data preprocessing to the merged input file should be taken into consideration. In the thesis above, two rules are listed to make the result better.

For the second hypothesis, there is not enough evidence to draw the conclusion. Based on the result of this thesis, this hypothesis may not be right. But the possible reason may be that the data preprocessing in this experiment is not enough. Because as more and more log files are merged, the dimensionality of the data becomes higher, which introduces more uncertainty to the whole input data, especially if the data pre-processing is not done well, it can make the result poor. A further experiment is needed to get the final conclusion.

The conclusion of this thesis is that it is a good idea to introduce more dimensions, or features, to the data, but the number of dimensions introduced and the data structure of data need to be considered very carefully.

Another critical point is that researchers need to be more attentive to the new input data in terms of data cleaning and data pre-processing to ensure that the results are acceptable to the greatest extent.

It is hoped that this thesis can be a good start for those who would like to do more research in this field.

Future works

Although the two hypotheses listed above have already got their answers, there is still a long way to go before such problems are entirely understood, mainly in terms of having enough time to test different algorithms and having a solid mathematical background and good coding skills to support this.

In this thesis, only one anomaly detection algorithm and two clustering algorithms are tested. For future work, testing other anomaly detection and clustering algorithms to see the effectiveness in comparison to the isolation forest and K-Means/DBSCAN methods adopted would allow for a more comprehensive review of which combination would suit the collected log data best.

As for T-SNE, this algorithm did not perform well on all four groups. How to make it available would be another future works.

In this thesis, combing three different log files together as input did not get an ideal result. However, if correct data is available and researchers have enough mathematical background to

process the data, will it be possible to get better results? That is also a good topic on the future worklist.

REFERENCES

Aggarwal, C. C. (2013). Proximity-based outlier detection. In Outlier Analysis (pp. 101- 133). Springer, New York, NY.

aihub. Training using a k-means algorithm with TensorFlow. Aihub.Cloud.Google. Retrieved November 2, 2020, from https://aihub.cloud.google.com/p/products%2F0e0d2ed0-5563-4639 b348-53a83ac4ff4e

Angiulli, F., & Pizzuti, C. (2005). Outlier mining in large high-dimensional data sets. IEEE Transactions on Knowledge and Data Engineering, 17(2), 203–215. https://doi.org/10.1109/tkde.2005.31

Bench-Capon, T., Soda, G., & Tjoa, M. A. (1999). Database and Expert Systems Applications: 10th International Conference, DEXA'99, Florence, Italy, August 30 - September 3, 1999, Proceedings (Lecture Notes in Computer Science (1677)) (1999th ed.). Springer.

Berthier, R., Sanders, W. H., & Khurana, H. (2010, October). Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. In 2010 First IEEE International Conference on Smart Grid Communications (pp. 350- 355). IEEE.

Bhardwaj, R. (2020, September 19). IDS vs IPS in 2020 - Download Detailed Comparison Table. IP With Ease. https://ipwithease.com/difference-between-ips-and-ids-in- network-security/

Birant, D., & Kut, A. (2007). ST-DBSCAN: An algorithm for clustering spatial–temporal data. Data & knowledge engineering, 60(1), 208-221.

Boukerche, A., Zheng, L., & Alfandi, O. (2020). Outlier Detection: Methods, Models, and Classification. ACM Computing Surveys (CSUR), 53(3), 1-37.

Bouman, C. A., Shapiro, M., Cook, G. W., Atkins, C. B., & Cheng, H. (1997). Cluster: An unsupervised algorithm for modeling Gaussian mixtures.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF. ACM SIGMOD Record, 29(2), 93–104. https://doi.org/10.1145/335191.335388

Briscoe, N. (2000). Understanding the OSI 7-layer model. PC Network Advisor, 120(2), 13-15.

Check Point Software. (2020, July 24). What is a Cyber Attack.https://www.checkpoint.com/cyber-hub/cyber-security/what-is-cyber-attack/#:%7E:text=A cyber-attack is an launch point for other attacks?

Chen, Z., & Li, Y. F. (2011). Anomaly detection based on enhanced DBScan algorithm. Procedia Engineering, 15, 178-182.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. IEEE transactions on pattern analysis and machine intelligence, 17(8), 790-799.

Coomans D, Massart DL. Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules. Analytica Chimica Acta. 1982 Jan 1;136:15-27.

Forouzan, B. A. (2012). Data Communications and Networking (5th ed.). McGraw-Hill Education.

Fumo, D. (2018, June 21). Types of Machine Learning Algorithms You Should Know. Medium. https://towardsdatascience.com/types-of-machine-learning-algorithms- you-should-know-953a08248861

Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2014). Outlier Detection for Temporal Data: A Survey. IEEE Transactions on Knowledge and Data Engineering, 26(9), 2250– 2267. https://doi.org/10.1109/tkde.2013.184

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In The elements of statistical learning (pp. 485-585). Springer, New York, NY.

Hautamaki, V., Karkkainen, I., & Franti, P. (2004, August). Outlier detection using k- nearest neighbour graph. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. (Vol. 3, pp. 430-433). IEEE.

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. Pattern Recognition Letters, 24(9–10), 1641–1650. https://doi.org/10.1016/s0167-8655(03)00003-5

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2015). Botnet in DDoS Attacks: Trends and Challenges. IEEE Communications Surveys & Tutorials, 17(4), 2242–2270. https://doi.org/10.1109/comst.2015.2457491

Jiang, M. F., Tseng, S. S., & Su, C. M. (2001). Two-phase clustering process for outliers detection. Pattern Recognition Letters, 22(6–7), 691–700. https://doi.org/10.1016/s0167-8655(00)00131-8

Kaur, T., Malhotra, V., & Singh, D. (2014). Comparison of network security tools-firewall, intrusion detection system and Honeypot. Int. J. Enhanced Res. Sci. Technol. Eng, 200204.

Khan, L., Awad, M., & Thuraisingham, B. (2007). A new intrusion detection system using support vector machines and hierarchical clustering. The VLDB journal, 16(4), 507-521.

Knox, E. M., & Ng, R. T. (1998, August). Algorithms for mining distancebased outliers in large datasets. In Proceedings of the international conference on very large data bases (pp. 392-403). Citeseer.

Kriegel, H. P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(3), 231-240.

Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E. D., & Taft, N. (2004). Structural analysis of network traffic flows. ACM SIGMETRICS Performance Evaluation Review. https://doi.org/10.1145/1012888.1005697

Lin, Y., Zhang, Y., & Ou, Y. J. (2010, April). The design and implementation of host-based intrusion detection system. In 2010 third international symposium on intelligent information technology and security informatics (pp. 595-598). IEEE.

Liu, H., & Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. Applied Sciences, 9(20), 4396.

Liu, H., & Lang, B. (2019). Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. Applied Sciences, 9(20), 4396. https://doi.org/10.3390/app9204396

LU, J. (2013). Semi-supervised multi-label classification algorithm based on local learning. Journalof Computer Applications, 32(12). https://doi.org/10.3724/sp.j.1087.2012.03308

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

Marteau, P. F., Soheily-Khah, S., & Béchet, N. (2017). Hybrid Isolation Forest- Application to Intrusion Detection. arXiv preprint arXiv:1705.03800.

Muda, Z., Yassin, W., Sulaiman, M. N., & Udzir, N. I. (2011). A K-Means and Naive Bayes Learning Approach for Better Intrusion Detection. Information Technology Journal, 10(3), 648–655. https://doi.org/10.3923/itj.2011.648.655

N., & Aroms, E. (2012). NIST Special Publication 800-94 Guide to Intrusion Detection and Prevention Systems (IDPS). CreateSpace Independent Publishing Platform.

Niyaz, Q., Sun, W., & Javaid, A. Y. (2017). A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN). ICST Transactions on Security and Safety. https://doi.org/10.4108/eai.28-12-2017.153515

Paxson, V. (1999). Bro: a system for detecting network intruders in real-time. Computer networks, 31(23-24), 2435-2463.

Pharate, A., Bhat, H., Shilimkar, V., & Mhetre, N. (2015). Classification of intrusion detection system. International Journal of Computer Applications, 118(7).

Reddy, G. N., & Reddy, G. J. (2014). A study of cyber security challenges and its emerging trends on latest technologies. arXiv preprint arXiv:1402.1842.

Ren, F., Hu, L., Liang, H., Liu, X., & Ren, W. (2008, December). Using density-based incremental clustering for anomaly detection. In 2008 International Conference on Computer Science and Software Engineering (Vol. 3, pp. 986-989). IEEE.

Rights, R. F. (2004). Use offense to inform defense. Find flaws before the bad guys do. Brackney, R. (1998, October). Cyber-intrusion response. In Proceedings Seventeenth IEEE Symposium on Reliable Distributed Systems (Cat. No. 98CB36281) (pp. 413-415). IEEE.

Roesch, M. (1999, November). Snort: Lightweight intrusion detection for networks. In Lisa (Vol. 99, No. 1, pp. 229-238).

Sazzadul Hoque, M. (2012). An Implementation of Intrusion Detection System Using Genetic Algorithm. International Journal of Network Security & Its Applications. https://doi.org/10.5121/ijnsa.2012.4208

Shin, S., Kwon, T., Jo, G. Y., Park, Y., & Rhy, H. (2010). An experimental study of hierarchical intrusion detection for wireless industrial sensor networks. IEEE Transactions on Industrial Informatics, 6(4), 744-757.

Sommer, R. (2003). Bro: An open source network intrusion detection system. Security, E-learning, E-Services, 17. DFN-Arbeitstagung über Kommunikationsnetze.

Stallings, W., & Brown, L. (2017). Computer Security: Principles and Practice (4th ed.). Pearson.

Syarif, I., Prugel-Bennett, A., & Wills, G. (2012, April). Unsupervised clustering approach for network anomaly detection. In International conference on networked digital technologies (pp. 135-145). Springer, Berlin, Heidelberg.

Trost, R. (2009). Practical Intrusion Analysis: Prevention and Detection for the TwentyFirst Century: Prevention and Detection for the TwentyFirst Century (1st ed.). AddisonWesley Professional.

Walpita, P. (2020, April 9). Dimensionality Reduction in Machine Learning using PCA. Medium. https://medium.com/@priyalwalpita/dimensionality-reduction-in-machine-learning-using-pca-e714ebddc032

Xu, D., & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. Annals of Data Science, 2(2), 165–193. https://doi.org/10.1007/s40745-015-0040-1

Yu, Y., Cao, L., Rundensteiner, E. A., & Wang, Q. (2017). Outlier Detection over Massive-Scale Trajectory Streams. ACM Transactions on Database Systems, 42(2), 1–33. https://doi.org/10.1145/3013527

Zargar, S. T., Joshi, J., & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. IEEE Communications Surveys & Tutorials. https://doi.org/10.1109/surv.2013.031413.00127