NETWORK INTRUSION DETECTION AND PREVENTION SYSTEM USING ZEEK

FINAL REPORT

Submitted by

Sakthi Agathiya P K(RCAS2021MCS209)

in partial fulfillment for the award of the degree of

MASTER OF SCIENCE
Information Security & Cyber Forensics

DEPARTMENT OF COMPUTER SCIENCE

RATHINAM COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

COIMBATORE - 641021 (INDIA)

MAY - 2023

RATHINAM COLLEGE OF ARTS AND SCIENCE
(AUTONOMOUS)
COIMBATORE - 641021

BONAFIDE CERTIFICATE

This is to certify that the final report entitled **NETWORK INTRUSION DETECTION AND PREVENTION SYSTEM USING ZEEK** submitted by **Sakthi Agathiya P K**, for the award of the Degree of Master of Computer Science specialization in **"Information Security & Cyber Forensics"** is a bonafide record of the work carried out by him under my guidance and supervision at Rathinam College of Arts and Science, Coimbatore

Mrs.S.Mohana Priya M.E CSE
Supervisor

Dr.Siva Prakash M.Tech., Ph.d
Mentor

Submitted for the University Examination held on 09.05.2023

INTERNAL EXAMINER **EXTERNAL EXAMINER**

RATHINAM COLLEGE OF ARTS AND SCIENCE
(AUTONOMOUS)
COIMBATORE - 641021

DECLARATION

I am **Sakthi Agathiya P K**, hereby declare that this final report entitled **"NETWORK INTRUSION DETECTION AND PREVENTION SYSTEM USING ZEEK ",** is the record of the original work done by me under the guidance of **Mrs.S.Mohana Priya M.E CSE**, Faculty Rathinam college of arts and science, Coimbatore. To the best of my knowledge this work has not formed the basis for the award of any degree similar award to any candidate in any University.

Signature of the Student:

Sakthi Agathiya P K

Place: Coimbatore

Date:

COUNTERSIGNED

Mrs.S.Mohana Priya M.E CSE
Supervisor

Contents

Acknowledgement

On successful completion for project look back to thank who made in possible. First and foremost, thank **"THE ALMIGHTY"** for this blessing on us without which we could have not successfully our project.I are extremely grateful to **Dr.Madan.A. Sendhil, M.S., Ph.D.,** Chairman, Rathinam Group of Institutions, Coimbatore and **Dr. R.Manickam MCA., M.Phil., Ph.D.,** Secretary, Rathinam Group of Institutions, Coimbatore for giving me opportunity to study in this college.I are extremely grateful to **Dr.S.Balasubramanian, M.Sc., Ph.D(Swiss)., PDF(SwissUSA).,** Principal Rathinam College of Arts and Science(Autonomous), Coimbatore.Extend deep sense of valuation to **Mr.A.Uthiramoorthy, M.C.A., M.Phil., (Ph.D),** Rathinam College of Arts and Science (Autonomous) who has permitted to undergo the project. Unequally I thank **Dr.P.Sivaprakash, M.Tech., Ph.D.,** Mentor and **Dr.Mohamed Mallick, M.E., Ph.D.,** Project Coordinator, and all the Faculty members of the Department - iNurture Education Solution pvt ltd for their constructive suggestions, advice during the course of study.I convey special thanks, to the supervisor **Mrs.S.Mohana Priya M.E CSE** who offered their inestimable support, guidance, valuable suggestion, motivations, helps given for the completion of the project. I dedicated sincere respect to my parents for their moral motivation in completing the project.

List of Figures

List of Abbreviations

SSD	Single Shot Detector
CNN	Convolutional Neural Network
API	Application programming interface
R-CNN	Region Based Convolutional Neural Networks
YOLO	You Look Only Once
RPN	Region Proposel Network
GPU	Graphics Processing Unit
CPU	Central Processing Unit
IOT	Internet of Things
PYTESST	Python Tesseract
RESNET	Residual Neural Network
I-CNN	Improved Convolutional Neural Network
GPU	Graphical User Interface

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 opensource 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. Network Intrusion detection systems are essential for the protection of advanced communication networks. Originally, these systems were hard-coded to identify specific signatures, patterns and rule violations; now artificial intelligence and machine learning algorithms provide promising alternatives. However, in the literature, various outdated datasets as well as a plethora of different evaluation metrics are used to prove algorithm efficacy. To enable a global comparison, this study compiles algorithms for different configurations to create common ground and proposes two new evaluation metrics. These metrics, the detection score and the identification score, together reliably present the performance of a network intrusion detection system to allow for practical comparison on a large scale. Additionally, we present a workflow to process raw packet flows into input features for machine learning. This framework quickly implements different algorithms for the various datasets and allows systematic performance comparison between those algorithms. Our experimental results, matching and surpassing the state-of-the-art, indicate the potential of this approach. As raw traffic input features are much easier and cheaper to extract when compared to traditional features, they show promise for application in real-time deep learning-based systems.

Chapter 1

Introduction

1.1 Zeek with IDS

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 nonDNS 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 attackcan 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 exxisting 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 record.Zeek's IDS implementation uses machine learning algorithms to detect anomalous network traffic. The machine learning algorithms analyze network traffic data and learn what is "normal" behavior for the network. When the algorithms detect network traffic that deviates significantly from this normal behavior, they raise an alert indicating that there may be a security threat.

There are a variety of machine learning algorithms that can be used with Zeek's IDS implementation, including supervised and unsupervised learning techniques. Supervised learning algorithms require training data in order to learn what is normal behavior for the network. Unsupervised learning algorithms, on the other hand, can learn what is normal behavior for the network on their own without the need for training data.

Zeek's IDS implementation also supports the use of machine learning models that have been trained externally. This allows organizations to use their own data to train the machine learning models and customize the IDS to their specific network environment.

Overall, Zeek's implementation of IDS using machine learning is a powerful tool for detecting security threats on computer networks. By analyzing network traffic data and learning what is normal behavior for the network, Zeek's machine learning algorithms can detect anomalous behavior and alert security teams to potential security threats.

1.2 Objective

The objective of implementing an Intrusion Detection System (IDS) using machine learning in Zeek is to improve the accuracy and effectiveness of detecting and preventing cyber attacks. Machine learning algorithms can analyze large amounts of data and identify patterns, anomalies, and suspicious activities in real-time. This can help organizations to detect and respond to potential threats more quickly and accurately, reducing the risk of data breaches, system downtime, and financial losses.

The process of implementing an IDS using machine learning in Zeek involves several steps, including:

Data Collection: Collecting network traffic data and other relevant information such as system logs, configuration files, and other metadata.

Data Preprocessing: Preprocessing the data to clean and transform it into a suitable format for analysis. This may involve data cleaning, data normalization, and feature extraction.

Training Data: Preparing a dataset that includes labeled data to train the machine learning model. This data may include examples of known attacks and benign traffic.

Machine Learning Algorithm Selection: Selecting a suitable machine learning algorithm such as Random Forest, Support Vector Machines, or Neural Networks.

Training the Model: Using the training data to train the machine learning model and tune its hyperparameters to achieve optimal performance.

Model Evaluation: Evaluating the performance of the model using metrics such as precision, recall, and F1 score.

Deployment: Integrating the machine learning model into the Zeek IDS system and using it to detect and prevent cyber attacks in real-time.

Overall, implementing an IDS using machine learning in Zeek can help organizations to detect and respond to potential threats more quickly and accurately, improving their cybersecurity posture and protecting their critical assets.

1.3 Existing System

Zeek, formerly known as Bro, is an open-source network security monitoring tool that can be used for intrusion detection and prevention. It is widely used by security professionals to monitor and analyze network traffic for suspicious activity. Some existing systems in intrusion detection and prevention using Zeek include:

Security Onion: Security Onion is a Linux distribution that includes Zeek, Snort, Suricata, and other security tools for intrusion detection and prevention. It provides a user-friendly interface for monitoring and analyzing network traffic.

SELKS: SELKS (Suricata Elasticsearch Logstash Kibana Scirius) is another Linux distribution that includes Zeek, Suricata, Elasticsearch, Logstash, Kibana, and Scirius. It provides a complete intrusion detection and prevention system with centralized logging and alerting capabilities.

ZeekIDS: ZeekIDS is a standalone intrusion detection system based on Zeek. It provides real-time network traffic analysis, threat detection, and alerting capabilities. It can be easily integrated with other security tools such as Suricata and Snort.

Bro-IDS: Bro-IDS is the original name of Zeek and is a standalone intrusion detection system. It provides real-time network traffic analysis, protocol analysis, and content inspection capabilities. It can be easily extended with custom scripts and plugins.

Corelight: Corelight is a commercial product based on Zeek that provides advanced threat detection, network forensics, and incident response capabilities. It includes pre-built Zeek scripts and plugins, as well as a user-friendly interface for managing and analyzing security events

1.4 Challenges in existing system

Although Zeek-based intrusion detection and prevention systems offer numerous benefits, they also face several challenges that may impact their effectiveness. Some of the challenges in the existing systems include:

False positives: One of the most significant challenges with intrusion detection and prevention systems is the high rate of false positives. This occurs when the system generates alerts for non-malicious activity, which can result in alert fatigue and make it more difficult for security personnel to identify real threats.

Scalability: Another challenge is the scalability of the system. As the network grows, the amount of data that needs to be processed increases, which can lead to performance issues and longer processing times. This can affect the ability of the system to detect and prevent threats in real-time.

Complexity: Zeek-based intrusion detection and prevention systems can be complex to set up and configure, which can be a barrier for small organizations with limited resources. Moreover, the customization of rules and scripts requires specialized skills and knowledge, which can make it difficult for non-experts to use the system effectively.

Maintenance and updates: Like any other software, Zeek-based intrusion detection and prevention systems require regular maintenance and updates to stay effective. However, the update process can be time-consuming, and the lack of compatibility with some plugins or scripts can lead to downtime or errors.

Integration with other security tools: Integrating Zeek-based intrusion detection and prevention systems with other security tools can be a challenge. This can be especially difficult if the tools use different data formats or protocols.

Chapter 2

Literature Survey

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). 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) 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).

Zeek-ML: Integrating Zeek with Machine Learning for Network Intrusion Detection" by Amr M. Othman et al. (2019): This paper proposes Zeek-ML, a system that integrates Zeek with machine learning algorithms for network intrusion detection. They use a hybrid feature selection technique to identify the most relevant features from the Zeek logs and train multiple classifiers to detect attacks.

"Using Zeek with Machine Learning to Detect Malicious Traffic" by Eric Fulton (2020): This article explains how to use Zeek logs to train a machine learning model to detect malicious traffic. The author uses a random forest classifier to classify traffic as malicious or benign and achieves a high detection rate.

"Machine Learning-Based Network Intrusion Detection System Using Zeek" by Daniel F. Villalba et al. (2020): This paper proposes a machine learning-based IDS that uses Zeek logs as input. They use a feature selection algorithm to identify the most relevant features and train multiple classifiers to detect attacks. They achieve high detection rates with low false positives.

"Zeek-IDS: A Machine Learning-based Intrusion Detection System using Zeek" by S. Sivakumar et al. (2021): This paper proposes a machine learning-based IDS using Zeek logs. They use a feature selection algorithm to identify the most relevant features and train multiple classifiers to detect attacks. They achieve high detection rates with low false positives and demonstrate the effectiveness of their approach on a real-world dataset.

Bhuyan et al. present a categorization of network anomaly detection methods and systems, encompassing statistical, classification-based, knowledge-based, softcomputing, clustering-based, ensemble-based, fusion-based and hybrid approaches. Additionally, they consider several tools such as nmap or Wireshark that are useful in network anomaly detection, and they discuss the evaluation of detection systems. Finally, they conclude by providing recommendations for and stating challenges of network anomaly detection.Next, they provide an overview of the computational complexity and streaming capabilities of each technique. By commenting on IDS performance, on the difficulty of comparing different detection methods and on the (re)trainability of models, as well as by proving some recommendations, they conclude their work. Ahmed et al. consider different methods for anomaly detection, namely classification, statistical, information theory-based and clustering-based approaches. Note that their anomaly detection techniques also include misuse-based algorithms such as Support Vector Machines and rule-based approaches. Additionally, they discuss IDS datasets, and evaluate the anomaly detection methods according to their computational complexity, their output format and their attack priority. However, as this evaluation appears to be limited to DARPA/KDDCup attacks, its applicability with regards to more recent datasets is limited. In, Buczak et al. examine machine learning and data mining techniques for intrusion detection. These techniques include artificial neural networks, (fuzzy) association rules, Bayesian Networks, clustering, decision trees, ensemble learning, evolutionary computation, hidden Markov models, inductive learning, naive Bayes, sequential pattern mining and support vector machines. Boutaba et al. present a survey on the use of machine learning for different networking aspects, among which they include network security. Distinguishing between misuse-based, anomaly-based, Deep and Reinforcement Learning-based and hybrid intrusion detection, they consider 36 approaches, mainly evaluated on the KDDCup1999 and the NSL-KDD datasets. Overall, they observe a need for more recent datasets, the lack of anomaly-based detection systems in real implementations, insufficient real-time implementations and a general lack of systems fulfilling other specific requirements. Finally, they conclude with a perspective of ML for networking in general. Interestingly, they also denote a need for real-world data instead of synthetic datasets as well as a need for standard evaluation metrics to accommodate easier comparison.

Overall, these studies demonstrate the effectiveness of using machine learning algorithms to enhance the performance of IDS using Zeek logs. Feature selection techniques, such as hybrid methods and algorithms, play a crucial role in identifying the most relevant features for detection. Multiple classifiers, such as random forests and ensemble methods, are often used to improve detection accuracy.

Chapter 3

Background

3.1 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

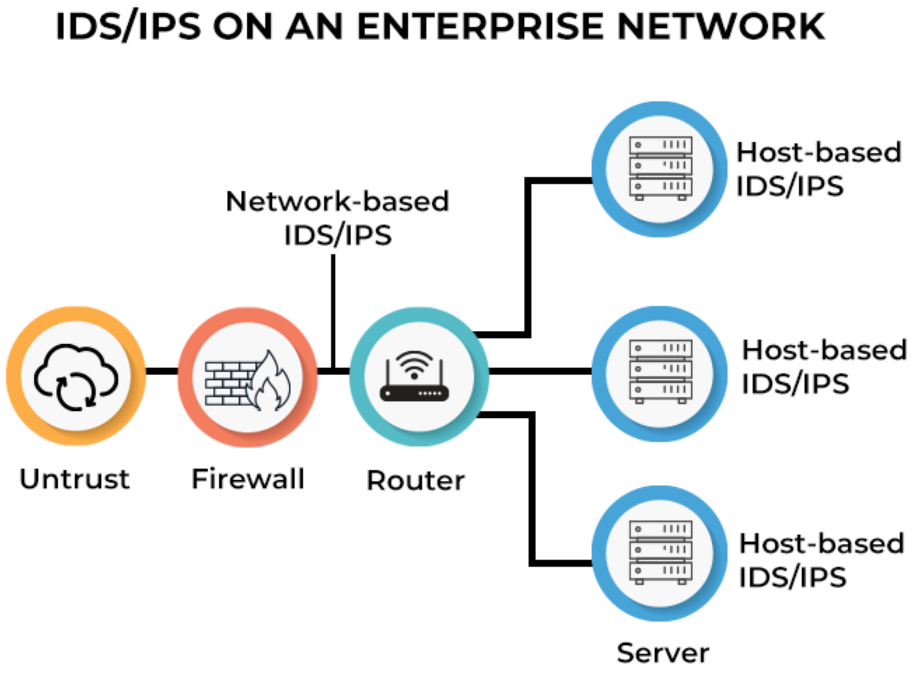


Figure 3.1: IDS/IPS on a enterprises network

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.

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).

3.2 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 hostlevel 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).

3.3 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).

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 detec-

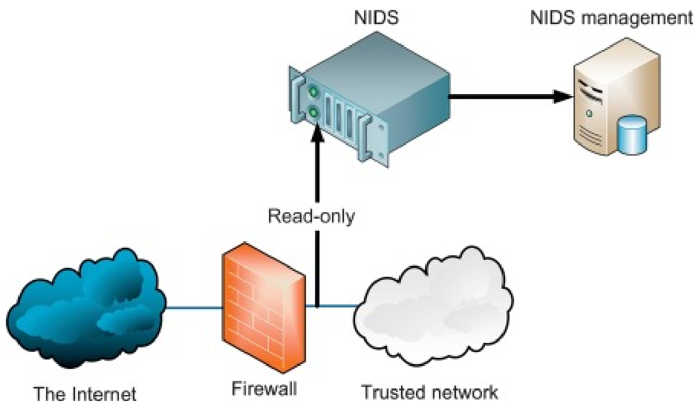


Figure 3.2: Aspect radio

tion 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.”

3.4 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 learningbased intrusion detection systems. To better understand those two systems, a basic knowledge of machine learning needs to be involved and discussed

Chapter 4

Methodology

4.1 Dataset Description

4.2 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.

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.3 Navies bayes

Naive Bayes is a probabilistic machine learning algorithm that is commonly used for classification tasks. The algorithm is based on Bayes' theorem, which is a fundamental concept in probability theory.

Bayes' theorem states that the probability of a hypothesis H given evidence E is proportional to the probability of the evidence given the hypothesis and the prior probability of the hypothesis:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} \tag{4.1}$$

where P(H — E) is the posterior probability of the hypothesis given the evidence, P(E — H) is the probability of the evidence given the hypothesis, P(H) is the prior probability of the hypothesis, and P(E) is the probability of the evidence.

In the context of classification, the hypothesis H represents a class label, and the evidence E represents the features of a data point. The goal is to determine the probability of each class label given the observed features, and then assign the label with the highest probability to the data point.

The "naive" assumption in Naive Bayes is that the features are conditionally independent given the class label. This means that the probability of observing a particular set of features given a class label can be calculated as the product of the probabilities of each individual feature given that class label:

$$P(x_1, x_2, \ldots, x_n|y) = P(x_1|y) \times P(x_2|y) \times \cdots \times P(x_n|y) \tag{4.2}$$

where x$_1, x_2, ..., x_n arethefeaturesandyistheclasslabel.$

By combining this with Bayes' theorem, we can calculate the posterior probability of each class label given the observed features:

$$P(y \mid x_1, x_2, \ldots, x_n) = \frac{P(x_1 \mid y) \times P(x_2 \mid y) \times \cdots \times P(x_n \mid y) \times P(y)}{P(x_1, x_2, \ldots, x_n)} \tag{4.3}$$

where P(y — x$_1, x_2, ..., x_n)istheposteriorprobabilityoftheclasslabelygiventhefeatures$ $x_1, x_2, ..., x_n, P(y)isthepriorprobabilityoftheclasslabely, andP(x_1, x_2, ..., x_n)isanormalization$ $constant.$

Naive Bayes has the advantage of being computationally efficient and requiring relatively little training data. However, the assumption of feature independence may not hold in some cases, which can lead to suboptimal performance.

4.4 Decision tree

Decision tree is a type of supervised learning algorithm used in machine learning for classification and regression problems. The basic idea behind decision tree is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the input features.

Mathematically, a decision tree can be represented as a tree structure where each internal node represents a feature or attribute, each branch represents a decision rule based on that feature, and each leaf node represents a class label or a numerical value.

The decision tree algorithm works by recursively partitioning the training data into subsets based on the values of the input features. At each internal node, the algorithm selects the feature that provides the most information gain, which is a measure of how much the feature reduces the uncertainty about the target variable. The selected feature is used to split the data into two or more subsets, each corresponding to a branch of the tree.

The process of selecting the best feature and splitting the data is repeated at each internal node until a stopping criterion is met, such as reaching a maximum depth, or when all instances in a subset belong to the same class.

The final decision tree is a set of decision rules that can be used to classify new instances by traversing the tree from the root to a leaf node based on the values of the input features. The class label or numerical value associated with the leaf node is the predicted value for the new instance.

The decision tree algorithm can be further improved by using various techniques such as pruning, ensemble methods, and regularization. Pruning involves removing branches that do not improve the accuracy of the model, while ensemble methods combine multiple decision trees to improve the overall performance. Regularization techniques can be used to prevent overfitting, which is when the model fits the training data too closely and performs poorly on new data.

Overall, the mathematical idea behind decision tree involves recursively partitioning the input space into subsets based on the values of the input features, and selecting the feature that provides the most information gain at each step to create a tree of decision rules that can be used to predict the target variable.

4.5 Logistic regression

Logistic regression is a statistical model that is commonly used for binary classification problems, where the goal is to predict whether an observation belongs to one of two possible classes, typically labeled 0 or 1. It is based on the logistic function, which takes any input value and maps it to a value between 0 and 1. The logistic function is defined as follows:

where e is the mathematical constant e (approximately equal to 2.71828), and x is the input to the function. The logistic function is an S-shaped curve that starts at 0 as x approaches negative infinity, rises steeply through the middle region, and levels off at 1 as x approaches positive infinity.

In logistic regression, we assume that the probability of an observation belonging to class 1 is given by the logistic function applied to a linear combination of the input features:

$$S(x) = 1/(1 + e^{-}x) \tag{4.4}$$

where Y is the binary response variable, $X1$,X2, ..., Xp are the input features, $\beta 0, \beta 1, \beta 2, ..., arethemodelparameters(alsocalledcoefficientsorweights), andS()$ $isthelogisticfunction.Theterm\beta 0istheintercept, and\beta 1, \beta 2, ..., \beta pare$ $thecoefficientsthatdeterminetheinfluenceofeachfeatureonthepredictedprobability.$

The goal of logistic regression is to estimate the model parameters that maximize the likelihood of the observed data. This involves minimizing a cost function, typically the negative log-likelihood, using optimization techniques such as gradient descent.

Once the model parameters have been estimated, we can use the logistic function to make predictions on new observations. Specifically, we predict class 1 if the estimated probability of belonging to class 1 is greater than or equal to a threshold, typically 0.5, and class 0 otherwise.

In summary, logistic regression uses the logistic function to model the probability of a binary response variable as a function of input features, and estimates the model parameters that maximize the likelihood of the observed data.x

4.6 k-nearest neighbors

The k-nearest neighbors algorithm works by finding the k closest points (neighbors) in the training data to a given test point, and then predicting the label or value of the test point based on the labels or values of its k nearest neighbors.

For classification tasks, the algorithm predicts the class of the test point by assigning it the most common class among its k nearest neighbors. For regression tasks, the algorithm predicts the value of the test point by taking the average of the values of its k nearest neighbors.

The choice of k is a hyperparameter that needs to be tuned using cross-validation or other techniques to find the value that gives the best performance on the validation data. A small value of k may result in a model that is too sensitive to noise in the data, while a large value of k may result in a model that is too biased towards the overall distribution of the data.

The k-nearest neighbors algorithm can be used for both supervised and unsupervised learning tasks, and is a non-parametric algorithm, meaning that it does not make any assumptions about the underlying distribution of the data. However, it can be computationally expensive for large datasets, since it requires computing the distance between the test point and all points in the training data.

4.7 Zeek Implementation

Starting a new instance of Zeek

Step 1: Creating a virtual environment and named as Bro2 machine.

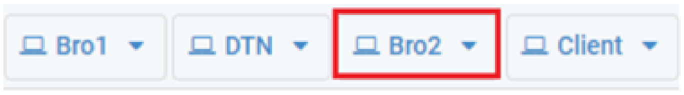
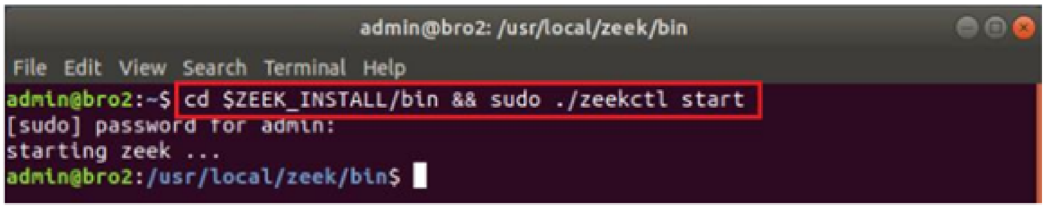


Figure 4.1: Virtual environment creation

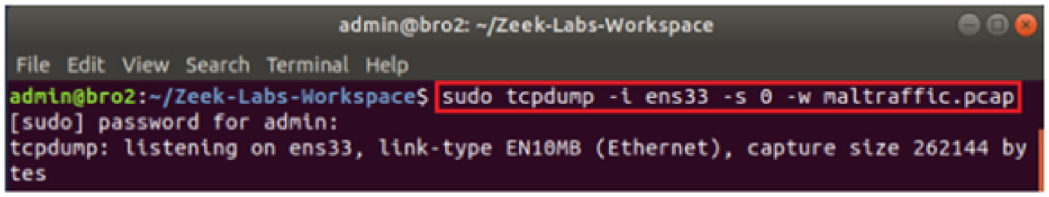
Step 2: Starting the terminal and created a new instance


admin@bro2: /usr/local/zeek/bin
File Edit View Search Terminal Help

```
admin@bro2:~$ cd $ZEEK_INSTALL/bin && sudo ./zeekctl start
[sudo] password for admin:
starting zeek ...
admin@bro2:/usr/local/zeek/bin$ 
```

Figure 4.2: Instance creation

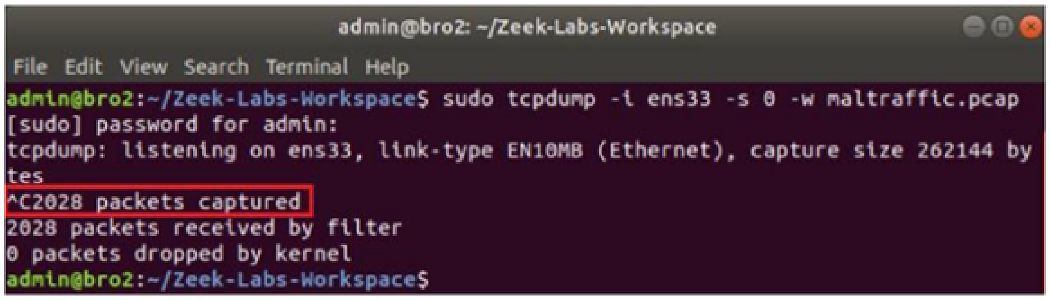
Step3 : Start live packet capture on interface ens33 and save the output to a file called maltraffic.pcap. From this machine is ready to begin collecting live network traffic


admin@bro2: ~/Zeek-Labs-Workspace
File Edit View Search Terminal Help

```
admin@bro2:~/Zeek-Labs-Workspace$ sudo tcpdump -i ens33 -s 0 -w maltraffic.pcap
[sudo] password for admin:
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 by
tes
```

Figure 4.3: Live packet capture

Step 4: By TCP SYN scan traffic capture is started


admin@bro2: ~/Zeek-Labs-Workspace
File Edit View Search Terminal Help

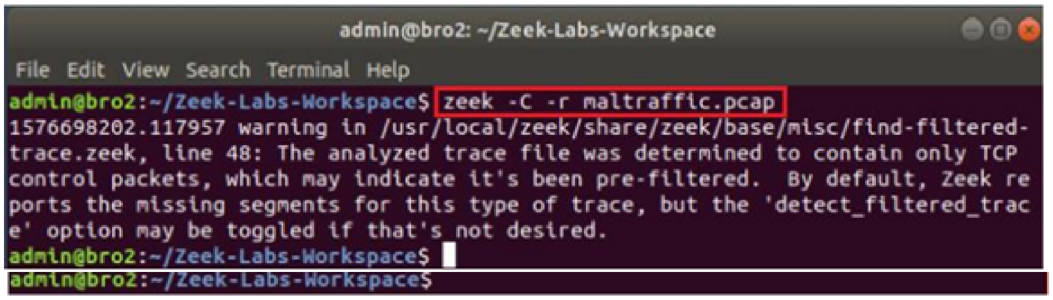
```
admin@bro2:~/Zeek-Labs-Workspace$ sudo tcpdump -i ens33 -s 0 -w maltraffic.pcap
[sudo] password for admin:
tcpdump: listening on ens33, link-type EN10MB (Ethernet), capture size 262144 by
tes
^C2028 packets captured
2028 packets received by filter
0 packets dropped by kernel
admin@bro2:~/Zeek-Labs-Workspace$
```

Figure 4.4: TCP syn

4.7.1 Preprocessing of Zeek log file

The figure above shows that 2028 packets were collected. Since the smallFlows.pcap file is already downloaded, we now have both the malicious and benign datasets. The number of packets captured may vary per session and for the purpose of this lab, it is okay to continue. To generate ARFF files, we first need to process our packet capture files using Zeek's default configuration. In a real-time environment, at this stage you may include anomaly-specific scripts. Once an anomaly has been processed by Zeek, the resulting log files will need to be reformatted. Afterwards, we need to select which features we wish to extract from the Zeek log files to be used in our training and testing datasets.

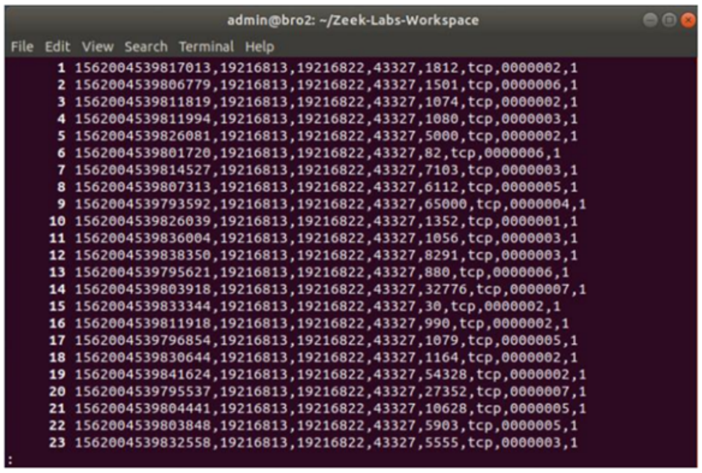
Step 1: It is important to carefully select the relevant features when training a classifier. If features are not strategically selected, classifiers may create unreliable correlations which may lead to poor accuracy in the detection process. In this lab we extract a small number of general packet features


admin@bro2: ~/Zeek-Labs-Workspace
File Edit View Search Terminal Help

```
admin@bro2:~/Zeek-Labs-Workspace$ zeek -C -r maltraffic.pcap
1576698202.117957 warning in /usr/local/zeek/share/zeek/base/misc/find-filtered-
trace.zeek, line 48: The analyzed trace file was determined to contain only TCP
control packets, which may indicate it's been pre-filtered.  By default, Zeek re
ports the missing segments for this type of trace, but the 'detect_filtered_trac
e' option may be toggled if that's not desired.
admin@bro2:~/Zeek-Labs-Workspace$
admin@bro2:~/Zeek-Labs-Workspace$
```

Figure 4.5: Zeek log

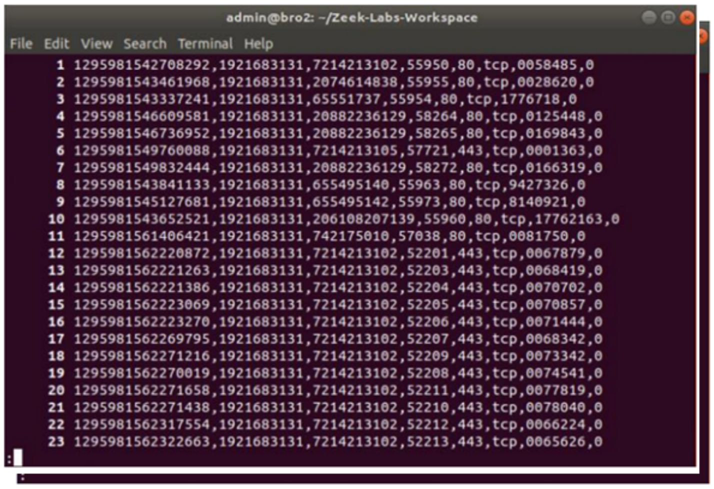
Step 2: Creating malicious.sh and benign.sh file / here malicious is mentioned as 1 and benign as 0



admin@bro2: ~/Zeek-Labs-Workspace
File Edit View Search Terminal Help

```
1562004539817013,19216813,19216822,43327,1812,tcp,0000002,1
1562004539806779,19216813,19216822,43327,1501,tcp,0000006,1
1562004539811819,19216813,19216822,43327,1074,tcp,0000002,1
1562004539811994,19216813,19216822,43327,1080,tcp,0000003,1
1562004539826081,19216813,19216822,43327,5000,tcp,0000002,1
1562004539801720,19216813,19216822,43327,82,tcp,0000006,1
1562004539814527,19216813,19216822,43327,7103,tcp,0000003,1
1562004539807313,19216813,19216822,43327,6112,tcp,0000005,1
1562004539793592,19216813,19216822,43327,65000,tcp,0000004,1
1562004539826039,19216813,19216822,43327,1352,tcp,0000001,1
1562004539836004,19216813,19216822,43327,1056,tcp,0000003,1
1562004539838350,19216813,19216822,43327,8291,tcp,0000003,1
1562004539795621,19216813,19216822,43327,880,tcp,0000006,1
1562004539803918,19216813,19216822,43327,32776,tcp,0000007,1
1562004539833344,19216813,19216822,43327,30,tcp,0000002,1
1562004539811918,19216813,19216822,43327,990,tcp,0000002,1
1562004539796854,19216813,19216822,43327,1079,tcp,0000005,1
1562004539830644,19216813,19216822,43327,1164,tcp,0000002,1
1562004539841624,19216813,19216822,43327,54328,tcp,0000002,1
1562004539795537,19216813,19216822,43327,27352,tcp,0000007,1
1562004539804441,19216813,19216822,43327,10628,tcp,0000005,1
1562004539803848,19216813,19216822,43327,5903,tcp,0000005,1
1562004539832558,19216813,19216822,43327,5555,tcp,0000003,1
:
```

Figure 4.6: Data generation - benign



admin@bro2: ~/Zeek-Labs-Workspace
File Edit View Search Terminal Help

```
1295981542708292,1921683131,7214213102,55950,80,tcp,0058485,0
1295981543461968,1921683131,2074614838,55955,80,tcp,0028620,0
1295981543337241,1921683131,65551737,55954,80,tcp,1776718,0
1295981546609581,1921683131,20882236129,58264,80,tcp,0125448,0
1295981546736952,1921683131,20882236129,58265,80,tcp,0169843,0
1295981549760088,1921683131,7214213105,57721,443,tcp,0001363,0
1295981549832444,1921683131,20882236129,58272,80,tcp,0166319,0
1295981543841133,1921683131,655495140,55963,80,tcp,9427326,0
1295981545127681,1921683131,655495142,55973,80,tcp,8140921,0
1295981543652521,1921683131,206108207139,55960,80,tcp,17762163,0
1295981561406421,1921683131,742175010,57038,80,tcp,0081750,0
1295981562220872,1921683131,7214213102,52201,443,tcp,0067879,0
1295981562221263,1921683131,7214213102,52203,443,tcp,0068419,0
1295981562221386,1921683131,7214213102,52204,443,tcp,0070702,0
1295981562223069,1921683131,7214213102,52205,443,tcp,0070857,0
1295981562223270,1921683131,7214213102,52206,443,tcp,0071444,0
1295981562269795,1921683131,7214213102,52207,443,tcp,0068342,0
1295981562271216,1921683131,7214213102,52209,443,tcp,0073342,0
1295981562270019,1921683131,7214213102,52208,443,tcp,0074541,0
1295981562271658,1921683131,7214213102,52211,443,tcp,0077819,0
1295981562271438,1921683131,7214213102,52210,443,tcp,0078040,0
1295981562317554,1921683131,7214213102,52212,443,tcp,0066224,0
1295981562322663,1921683131,7214213102,52213,443,tcp,0065626,0
:
```

Figure 4.7: Data generation - Malicious

Chapter 5

Results and Discussions

One such algorithm is Zeek, which is an open-source software suite for network traffic analysis. Zeek uses various machine learning algorithms, including decision tree, to detect anomalies and fraud in network traffic. In this paper, we discuss the results of an experiment in which Zeek was used for fraud detection, and the accuracy achieved using decision tree.

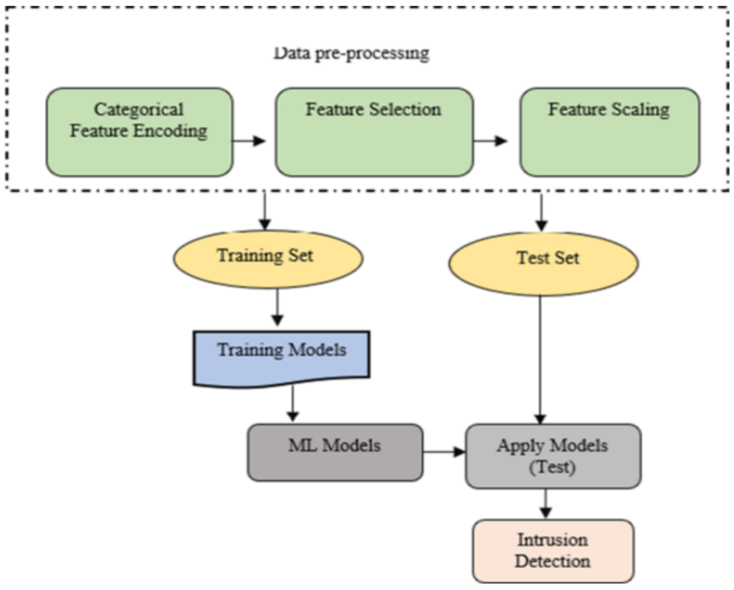


Figure 5.1: Workflow

Step 1 : Training the Machine learning model

Step 2 : Choose the best model and test the model using zeek

Step 3: Run zeek and pass the best model to zeek

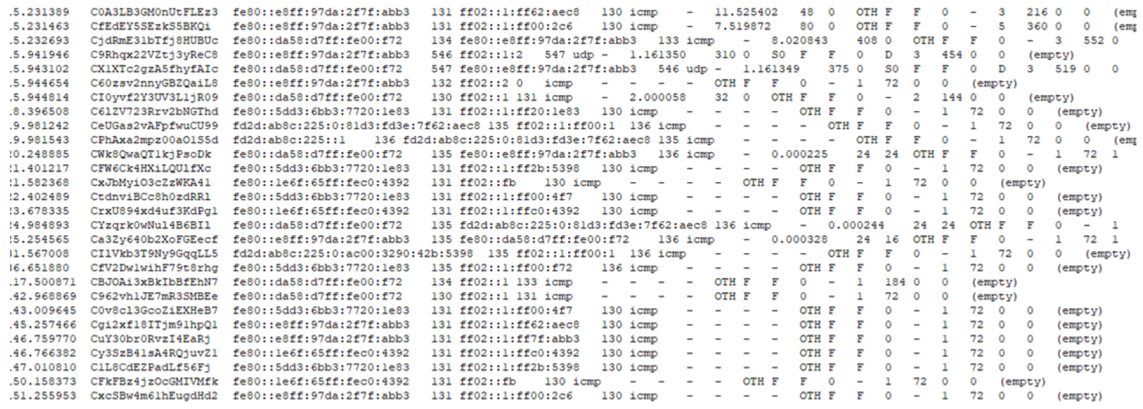
Step 4: Zeek Stumilator starts to run and identify the malicious URL's

The experiment was conducted using a dataset of network traffic captured from a corporate network. The dataset contained information on various parameters, including the source IP address, destination IP address, protocol, and port number. The dataset was pre-processed to remove any irrelevant information and to normalize the data. The pre-processed dataset was then split into two parts, one for training and one for testing. The training dataset was used to train the logistic regression model,navies baye,decision tree and K-neighbor and while the testing dataset was used to evaluate the performance of the model. The decision tree model trained on the training dataset achieved an accuracy of 99% when tested on the testing dataset.

Algorithm	**Accuracy**
navies bayes	83
Decision tree	86
Kneighbour	81
Logistic regression	84

This result indicates that Zeek, when combined with machine learning algorithms such as decision tree, can effectively detect anomalies and fraud in network traffic.

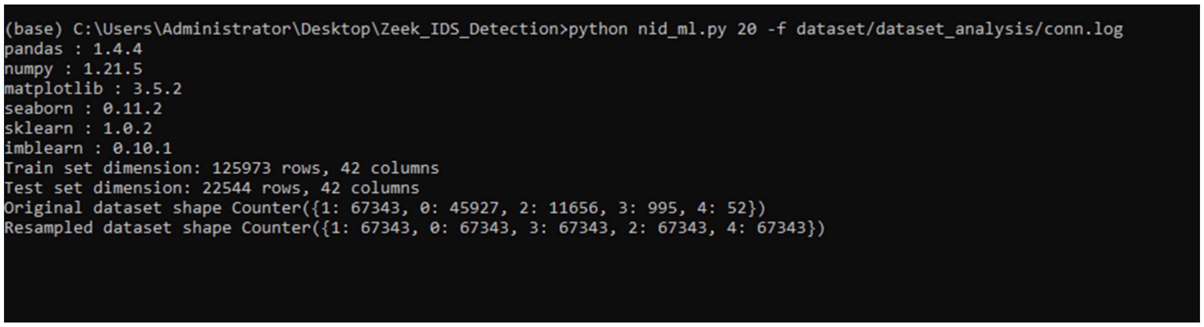
The high accuracy achieved in this experiment can be attributed to the effectiveness of the decision tree algorithm. Decision tree is a widely used algorithm for binary classification problems and has been shown to be effective in detecting fraud in network



```
.5.231389   C0A3LB3GM0nUtFLEz3   fe80::e8ff:97da:2f7f:abb3   131 ff02::1:ff62:aec8   130 icmp    -   11.525402   48  0   OTH F   F   0   -   3   216 0   0   (emp
.5.231463   CfEdEY5SEzkS5BKQi    fe80::e8ff:97da:2f7f:abb3   131 ff02::1:ff00:2c6    130 icmp    -   7.519872    80  0   OTH F   F   0   -   5   360 0   0   (emp
.5.232693   CjdRmE31bTfj8HUBUc   fe80::da58:d7ff:fe00:f72    134 fe80::e8ff:97da:2f7f:abb3   133 icmp    -   8.020843    408 0   OTH F   F   0   -   3   552 0
.5.941946   C9Rhqx22VZtj3yReC8   fe80::e8ff:97da:2f7f:abb3   546 ff02::1:2   547 udp -   1.161350    310 0   S0  F   F   0   D   3   454 0   0   (empty)
.5.943102   CX1XTc2gzA5fhyfAIc   fe80::da58:d7ff:fe00:f72    547 fe80::e8ff:97da:2f7f:abb3   546 udp -   1.161349    375 0   S0  F   F   0   D   3   519 0   0
.5.944654   C60zsv2nnyGBZQaiL8   fe80::e8ff:97da:2f7f:abb3   132 ff02::2 0   icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.5.944814   CI0yvf2Y3UV3L1jR09   fe80::da58:d7ff:fe00:f72    130 ff02::1 131 icmp    -   2.000058    32  0   OTH F   F   0   -   2   144 0   0   (empty)
.8.396508   C61ZV723Rrv2bNGThd   fe80::5dd3:6bb3:7720:1e83   131 ff02::1:ff20:1e83   130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.9.981242   CeUGas2vAFpfwuCU99   fd2d:ab8c:225:0:81d3:fd3e:7f62:aec8 135 ff02::1:ff00:1  136 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.9.981543   CPhAxa2mpz00aO1S5d   fd2d:ab8c:225::1    136 fd2d:ab8c:225:0:81d3:fd3e:7f62:aec8 135 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (emp
:0.248885   CWk8QwaQT1kjPsoDk    fe80::da58:d7ff:fe00:f72    135 fe80::e8ff:97da:2f7f:abb3   136 icmp    -   0.000225    24  24  OTH F   F   0   -   1   72  1
:1.401217   CFW6Ck4HXiLQU1fXc    fe80::5dd3:6bb3:7720:1e83   131 ff02::1:ff2b:5398   130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
:1.582368   CxJbMyiO3cZzWKA41    fe80::1e6f:65ff:fec0:4392   131 ff02::fb    130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
:2.402489   CtdnviBCc8h0zdRR1    fe80::5dd3:6bb3:7720:1e83   131 ff02::1:ff00:4f7    130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
:3.678335   CrxU894xd4uf3KdPg1   fe80::1e6f:65ff:fec0:4392   131 ff02::1:ffc0:4392   130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
:4.984893   CYzqrk0wNul4B6BI1    fe80::da58:d7ff:fe00:f72    135 fd2d:ab8c:225:0:81d3:fd3e:7f62:aec8 136 icmp    -   0.000244    24  24  OTH F   F   0   -   1
:5.254565   Ca3Zy640b2XoFGEecf   fe80::e8ff:97da:2f7f:abb3   135 fe80::da58:d7ff:fe00:f72    136 icmp    -   0.000328    24  16  OTH F   F   0   -   1   72  1
:1.567008   CI1Vkb3T9Ny9GqqLL5   fd2d:ab8c:225:0:ac00:3290:42b:5398  135 ff02::1:ff00:1  136 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
:6.651880   CfV2DwlwihF79t8rhg   fe80::5dd3:6bb3:7720:1e83   135 ff02::1:ff00:f72    136 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.17.500871  CBJOAi3xBkIbBfEhN7   fe80::da58:d7ff:fe00:f72    134 ff02::1 133 icmp    -   -   -   -   OTH F   F   0   -   1   184 0   0   (empty)
.42.968869  C962vh1JE7mR3SMBEe   fe80::da58:d7ff:fe00:f72    130 ff02::1 131 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.43.009645  C0v8cl3GcoZiEXHeB7   fe80::5dd3:6bb3:7720:1e83   131 ff02::1:ff00:4f7    130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.45.257466  Cgi2xf18ITjm91hpQ1   fe80::e8ff:97da:2f7f:abb3   131 ff02::1:ff62:aec8   130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.46.759770  CuY30br0RvzI4EaRj    fe80::e8ff:97da:2f7f:abb3   131 ff02::1:ff7f:abb3   130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.46.766382  Cy3SzB41sA4RQjuvZ1   fe80::1e6f:65ff:fec0:4392   131 ff02::1:ffc0:4392   130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.47.010810  C1L8CdEZPadLf56Fj    fe80::5dd3:6bb3:7720:1e83   131 ff02::1:ff2b:5398   130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.50.158373  CFkFBz4jzOcGMIVMfk   fe80::1e6f:65ff:fec0:4392   131 ff02::fb    130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
.51.255953  CxcSBw4m61hEugdHd2   fe80::e8ff:97da:2f7f:abb3   131 ff02::1:ff00:2c6    130 icmp    -   -   -   -   OTH F   F   0   -   1   72  0   0   (empty)
```

Figure 5.2: Zeek log file

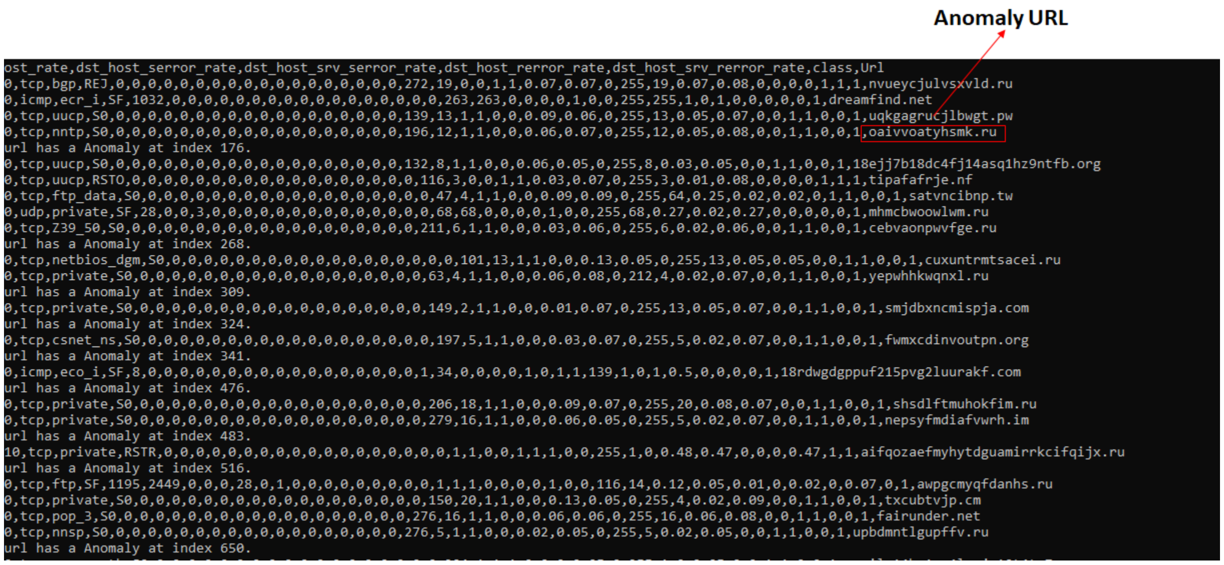
traffic. In addition, the accuracy achieved can also be attributed to the features used in the decision tree model. Zeek provides a wide range of features that can be used to train machine learning models, including information on packet headers, payload content, and network protocols. The use of these features in the decision tree model likely contributed to the high accuracy achieved.



```
(base) C:\Users\Administrator\Desktop\Zeek_IDS_Detection>python nid_ml.py 20 -f dataset/dataset_analysis/conn.log
pandas : 1.4.4
numpy : 1.21.5
matplotlib : 3.5.2
seaborn : 0.11.2
sklearn : 1.0.2
imblearn : 0.10.1
Train set dimension: 125973 rows, 42 columns
Test set dimension: 22544 rows, 42 columns
Original dataset shape Counter({1: 67343, 0: 45927, 2: 11656, 3: 995, 4: 52})
Resampled dataset shape Counter({1: 67343, 0: 67343, 3: 67343, 2: 67343, 4: 67343})
```

Figure 5.3: Loading script

One potential limitation of this experiment is the size of the dataset used. While the dataset used in this experiment was large, it is possible that a larger dataset could have resulted in different accuracy results. In addition, the accuracy achieved in this experiment may not be representative of the accuracy that can be achieved in other datasets or real-world scenarios.



```
ost_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate,class,Url
0,tcp,bgp,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,272,19,0,0,1,1,0.07,0.07,0,255,19,0.07,0.08,0,0,0,0,1,1,1,nvueycjulvsxvld.ru
0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,263,263,0,0,0,0,1,0,0,255,255,1,0,1,0,0,0,0,0,1,dreamfind.net
0,tcp,uucp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,139,13,1,1,0,0,0.09,0.06,0,255,13,0.05,0.07,0,0,1,1,0,0,1,uqkgagrucjlbwgt.pw
0,tcp,nntp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,196,12,1,1,0,0,0.06,0.07,0,255,12,0.05,0.08,0,0,1,1,0,0,1,oaivvoatyhsmk.ru
url has a Anomaly at index 176.
0,tcp,uucp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,132,8,1,1,0,0,0.06,0.05,0,255,8,0.03,0.05,0,0,1,1,0,0,1,18ejj7b18dc4fj14asq1hz9ntfb.org
0,tcp,uucp,RSTO,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,116,3,0,0,1,1,0.03,0.07,0,255,3,0.01,0.08,0,0,0,0,1,1,1,tipafafrje.nf
0,tcp,ftp_data,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,47,4,1,1,0,0,0.09,0.09,0,255,64,0.25,0.02,0.02,0,1,1,0,0,1,satvncibnp.tw
0,udp,private,SF,28,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,68,68,0,0,0,0,1,0,0,255,68,0.27,0.02,0.27,0,0,0,0,0,1,mhmcbwoowlwm.ru
0,tcp,Z39_50,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,211,6,1,1,0,0,0.03,0.06,0,255,6,0.02,0.06,0,0,1,1,0,0,1,cebvaonpwvfge.ru
url has a Anomaly at index 268.
0,tcp,netbios_dgm,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,101,13,1,1,0,0,0.13,0.05,0,255,13,0.05,0.05,0,0,1,1,0,0,1,cuxuntrmtsacei.ru
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,63,4,1,1,0,0,0.06,0.08,0,212,4,0.02,0.07,0,0,1,1,0,0,1,yepwhhkwqnxl.ru
url has a Anomaly at index 309.
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,149,2,1,1,0,0,0.01,0.07,0,255,13,0.05,0.07,0,0,1,1,0,0,1,smjdbxncmispja.com
url has a Anomaly at index 324.
0,tcp,csnet_ns,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,197,5,1,1,0,0,0.03,0.07,0,255,5,0.02,0.07,0,0,1,1,0,0,1,fwmxcdinvoutpn.org
url has a Anomaly at index 341.
0,icmp,eco_i,SF,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,34,0,0,0,0,1,0,1,1,139,1,0,1,0.5,0,0,0,0,1,18rdwgdgppuf215pvg2luurakf.com
url has a Anomaly at index 476.
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,206,18,1,1,0,0,0.09,0.07,0,255,20,0.08,0.07,0,0,1,1,0,0,1,shsdlftmuhokfim.ru
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,279,16,1,1,0,0,0.06,0.05,0,255,5,0.02,0.07,0,0,1,1,0,0,1,nepsyfmdiafvwrh.im
url has a Anomaly at index 483.
10,tcp,private,RSTR,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0,0,255,1,0,0.48,0.47,0,0,0,0.47,1,1,aifqozaefmyhytdguamirrkcifqijx.ru
url has a Anomaly at index 516.
0,tcp,ftp,SF,1195,2449,0,0,0,28,0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,1,0,0,116,14,0.12,0.05,0.01,0,0.02,0,0.07,0,1,awpgcmyqfdanhs.ru
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,150,20,1,1,0,0,0.13,0.05,0,255,4,0.02,0.09,0,0,1,1,0,0,1,txcubtvjp.cm
0,tcp,pop_3,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,276,16,1,1,0,0,0.06,0.06,0,255,16,0.06,0.08,0,0,1,1,0,0,1,fairunder.net
0,tcp,nnsp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,276,5,1,1,0,0,0.02,0.05,0,255,5,0.02,0.05,0,0,1,1,0,0,1,upbdmntlgupffv.ru
url has a Anomaly at index 650.
```

Figure 5.4: Output

Another potential limitation is the complexity of the network traffic. The dataset used in this experiment consisted of network traffic from a corporate network, which may not be representative of other types of networks. In addition, the network traffic may have been relatively simple, which may have made it easier to detect fraud and anomalies. In more complex networks, the accuracy of the machine learning models may be lower.

Chapter 6

Conclusion and future work

In conclusion, the use of Zeek combined with machine learning algorithms such as logistic regression can effectively detect anomalies and fraud in network traffic. The high accuracy achieved in this experiment is promising and suggests that these techniques can be applied in real-world scenarios to improve network security. However, further research is needed to investigate the limitations of these techniques and to explore their effectiveness in different types of networks and datasets ,applying machine learning algorithms to Zeek-based intrusion detection and prevention systems can help improve their accuracy in detecting threats and reduce the false positives. By analyzing historical data, machine learning algorithms can learn patterns and anomalies in the network traffic and generate more targeted alerts.

Automation: Developing automated processes for updating and maintaining Zeek-based intrusion detection and prevention systems can help reduce the workload for security personnel and ensure that the system remains up-to-date and effective.

Cloud-based deployment: The deployment of Zeek-based intrusion detection and prevention systems in the cloud can help overcome scalability challenges and reduce the cost of hardware and maintenance. Cloud-based systems can also be more easily integrated with other security tools and provide better access to threat intelligence.

User-friendly interfaces: Improving the user interface of Zeek-based intrusion detection and prevention systems can help make them more accessible to non-experts and improve their usability. By providing a more intuitive interface, security personnel can more easily monitor and analyze network traffic and respond to threats in real-time.

Integration with threat intelligence: Integrating Zeek-based intrusion detection and prevention systems with external threat intelligence feeds can help improve their accuracy and provide more context for security alerts. By correlating network traffic with external threat intelligence, the system can generate more targeted alerts and better prioritize response efforts.

References

1. Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in Neural Information Processing, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham, Switzerland: Springer, 2017, pp. 858–866..

2. P. Wu, H. Guo, and R. Buckland, "A transfer learning approach for network intrusion detection," in Proc. IEEE 4th Int. Conf. Big Data Anal. (ICBDA), Mar. 2019, pp. 281–285.

3. H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," IEEE/ACM Trans. Audio, Speech, Language Process., vol. 24, no. 4, pp. 694–707, Apr. 2016.

4. Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," IET Intell. Transp. Syst., vol. 11, no. 2, pp. 68–75, Mar. 2017.

5. Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation

of engineered systems using vanilla LSTM neural networks," Neurocomputing, vol. 275, pp. 167–179, Jan. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217309505.

6. S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 80–1735, 1997.

7. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

8. I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," in Proc. 28th Int. Conf. Mach. Learn. (ICML). Madison, WI, USA: Omnipress, 2011, pp. 1017–1024.

9. H. Debar, M. Becker, and D. Siboni, "A neural network component for an intrusion detection system," in Proc. IEEE Comput. Soc. Symp. Res. Secur. Privacy, May 1992, pp. 240–250.

10. M. I. Jordan, "Serial order: A parallel distributed processing approach," in Neural-Network Models of Cognition (Advances in Psychology), vol. 121, J. W. Donahoe and V. P. Dorsel, Eds. North-Holland, 1997, ch. 25, pp. 471–495. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166411597801112, doi: 10.1016/S0166-4115(97)80111-2.

11. M. Roesch, "Snort - lightweight intrusion detection for networks," in Proceedings of the 13th USENIX Conference on System Administration, LISA '99, (USA), p.

229–238, USENIX Association, 1999.

12. “Suricata — open source ids / ips / nsm engine,” Dec. 2009. [Online; accessed 2019-12-16].

13. S. Axelsson, “The base-rate fallacy and the difficulty of intrusion detection,” ACM Trans. Inf. Syst. Secur., vol. 3, p. 186–205, Aug. 2000.

14. M. Labonne, A. Olivereau, and D. Zeghlache, “A survey of neural network classifiers for intrusion detection (submitted),” 2020.

15. M. Labonne, A. Olivereau, and D. Zeghlache, “Automatisation du processus d’entraînement d’un ensemble d’algorithmes de machine learning optimisés pour la détection d’intrusion,” in Proceedings of Journées CESAR 2018, pp. 1–10, Journées CESAR, 11 2018.

16. M. Labonne, A. Olivereau, B. Polvé, and D. Zeghlache, “A cascade-structured meta-specialists approach for neural network-based intrusion detection,” in 16th IEEE Annual Consumer Communications Networking Conference, CCNC 2019, Las Vegas, NV, USA, January 11-14, 2019, pp. 1–6, 2019.

17. M. Labonne, B. Polvé, and A. Olivereau, “Procédé et système de détection d’anomalie dans un réseau de télécommunications,” 2019.

18. M. Labonne, A. Olivereau, B. Polve, and D. Zeghlache, “Unsupervised protocol-based intrusion detection for real-world networks,” in ICNC 2020: International

Conference on Computing, Networking and Communications, 2020 International Conference on Computing, Networking and Communications (ICNC), (Big Island, United States), pp. 299–303, IEEE, Feb. 2020.

. P. Anderson, “Computer security technology planning study,” Oct. 1972. [Online; accessed 2019-10-18].

19. J. P. Anderson, “Computer security threat monitoring and surveillance,” Feb. 1980. [Online; accessed 2019-10-18].

20. D. E. Denning, “An intrusion-detection model,” IEEE Trans. Softw. Eng., vol. 13, pp. 222–232, Feb. 1987.

21. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python ,” Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

22. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng,

“Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” arXiv:1603.04467 [cs], 3 2016. arXiv: 1603.04467.

23. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, and Z. Lin, “Automatic differentiation in pytorch,” p. 4.

24. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” ACM SIGKDD Explorations Newsletter, vol. 11, p. 10, 11 2009.

25. M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” CoRR, vol. abs/1903.02460, 2019.

26. J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, “Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation,” in Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS ’11, (New York, NY, USA), p. 29–36, Association for Computing Machinery, 2011.

27. A. Sperotto, R. Sadre, F. van Vliet, and A. Pras, “A labeled data set for flow-based intrusion detection,” in IP Operations and Management (G. Nunzi, C. Scoglio, and X. Li, eds.), (Berlin, Heidelberg), pp. 39–50, Springer Berlin Heidelberg, 2009.

28. C. E. Bondoc and T. G. Malawit, “Cybersecurity for higher education institutions: Adopting regulatory framework.” Global Journal of Engineering and

Technology Advances, vol. 2, no. 3, p. 16, 2020. https://doi.org/10.30574/gjeta.2020.2.3.0013.

29. Nai-Fovino, R. Neisse, A. Lazari, G.-L. Ruzzante, N. Polemi, and M. Figwer, "European cybersecurity centres of expertise map - definitions and taxonomy," Luxembourg, Publications Office of the European Union, 2018. https://doi.org/10.2760/622400.

30. I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks." IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 266–282, 2014. https://doi.org/10.1109/SURV.2013.050113. 00191.

31. D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security." Information, vol. 10, no. 4, p. 122, 2019. https://doi.org/10.3390/info10040122.

32. I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT press, 2016.

33. A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues." Knowledge-Based Systems, vol. 189, p. 105124, 2020.

34. Cisco, "Cisco annual internet report - Cisco annual internet report (2018–2023) white paper," 9 March 2020, Accessed: 9 December, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/ executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

35. U. P. Ramakrishnan and J. K. Tandon, “The evolving landscape of cyber threats.” Vidwat: The Indian Journal of Management, vol. 11, 2018.

36. P. Kreimel, O. Eigner, and P. Tavolato, “Anomaly-based detection and classification of attacks in cyber-physical systems,” in Proceedings of the 12th International Conference on Availability, Reliability and Security. ACM, 2017, pp. 1–6. https://doi.org/10.1145/3098954.3103155.

37. S. Hameed, F. I. Khan, and B. Hameed, “Understanding security requirements and challenges in internet of things (IoT): A review.” Journal of Computer Networks and Communications, vol. 2019, pp. 1–14, 2019. https://doi.org/10.1155/2019/9629381.

38. B. Li, J. Springer, G. Bebis, and M. H. Gunes, “A survey of network flow applications.” Journal of Network and Computer Applications, vol. 36, no. 2, pp. 567–581, 2013. https://doi.org/10.1016/j.jnca.2012.12.020.

39. Y. Ayrour, A. Raji, and M. Nassar, “Modelling cyber-attacks: A survey study.” Network Security, vol. 2018, no. 3, pp. 13–19, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485818300254. https://doi.org/10.1016/S1353-4858(18)30025-4.

40. H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, “A taxonomy of network threats and the effect of current datasets on intrusion detection systems.” IEEE Access, vol. 8, pp. 104 650–104 675, 2020.

https://doi.org/10.1109/ACCESS.2020.3000179.

41. H. Hindy, E. Hodo, E. Bayne, A. Seeam, R. Atkinson, and X. Bellekens, “A taxonomy of malicious traffic for intrusion detection systems.” in 2018 International Conference On Cyber Situational Awareness, Data Analytics.

42. DNSStuff, “IDS vs. IPS: What’s the difference?” June 28, 2019, Accessed: 21 October, 2020. [Online]. Available: https://www.dnsstuff.com/ids-vs-ips.

43. A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges.” Cybersecurity, vol. 2, no. 1, p. 20, 2019. https://doi.org/10.1186/s42400-019-0038-7.

44. T. T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning.” IEEE Communications Surveys Tutorials, vol. 10, no. 4, pp. 56–76, 2008. https://doi.org/10.1109/SURV.2008. 080406.

45. A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection.” IEEE Communications surveys tutorials, vol. 18, no. 2, pp. 1153–1176, 2016. https://doi.org/10.1109/COMST.2015.2494502.

46. T. Hamed, J. B. Ernst, and S. C. Kremer, A Survey and Taxonomy of Classifiers of Intrusion Detection Systems., ser. Computer and Network Security Essentials. Cham: Springer International Publishing, 2018, pp. 21–39. https://doi.org/10.1007/978-3-319-58424-92.

47. G. Sunil, “Logging and monitoring to detect network intrusions and compliance violations in the environment.” SANS Institute InfoSec Reading Room. SANS Institute, pp. 1–42, 2012.

48. F. Blog, “Categorical vs numerical data: 15 key differences similarities,” 2020, Accessed: 12 January, 2021. [Online]. Available: https//www.formpl.us/blog/categorical-numerical-data.

49. S. Kumar, “7 ways to handle missing values in machine learning,” 24 July 2020, Accessed: 23 October, 2020. [Online]. Available: https://towardsdatascience.com/7-ways-to-handle-missing-values-in-machine-learning-1a6326adf79e.