NETWORK INTRUSION DETECTION AND PREVENTION SYSTEM USING ZEEK

1rd Mr. Sakthi Agathiya P K
PG Scholar
Rathinam College of Arts and Science
Coimbatore, India
sakthiagathiyapk@gmail.com

2rd Mrs.Mohanapriya
Associate Professor
iNurture Education Solutions
Coimbatore, India
Mohanapriya.s@inurture.co.in

***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 However, in the literature, various outdated datasets as well as a plethora of different evaluation metrics are used to prove algorithm efficacy.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.**

***Index Terms*—Index Terms— IDS, IPS, Machine Learning, Zeek, Detect, Prevent, Malicious URL**

I. 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.

A. 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.

B. 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.

C. Denial of service attack:

It is a kind of attack targeting systems, servers, or networks with traffic to exhaust resources and bandwidth.

D. 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 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. 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.

II. 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:

A. Data Collection:

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

B. 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.

C. 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.

D. Machine Learning Algorithm Selection:

Selecting a suitable machine learning algorithm such as Random Forest, Support Vector Machines, or Neural Networks.

E. Training the Model:

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

F. Model Evaluation:

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

G. 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.

H. 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:

I. 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.

J. 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.

K. 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.

L. 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.

M. 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

III. 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:

A. 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.

B. 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.

C. 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.

D. 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.

E. 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.

IV. 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.

A. Denial of service (DOS) attacks:

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.

B. 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. [25] 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. [26] 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 [27], 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. [23] 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.

V. Dataset Description

An IDS is designed to only provide an alert about a potential incident, which enables a security operations center (SOC) analyst to investigate the event and determine whether it requires further action. An IPS, on the other hand, takes action itself to block the attempted intrusion or otherwise remediate the incident.

A. 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.

B. 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.

C. 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.

D. 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.

E. 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:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} \tag{1}$$

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:

$$P(x_1, x_2, \ldots, x_n|y) = P(x_1|y) \times P(x_2|y) \times \cdots \times P(x_n|y) \tag{2}$$

where $x_1, x_2, ..., x_n are the features and y is the class label.$

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

$$P(y \mid x_1, x_2, \ldots, x_n) = \frac{P(x_1 \mid y) \times P(x_2 \mid y) \times \cdots \times P(x_n \mid y) \times P(y)}{P(x_1, x_2, \ldots, x_n)} \tag{3}$$

where $P(y|x_1, x_2, ..., x_n)$ is the posterior probability of the class label y given the features $x_1, x_2, ..., x_n, P(y)$ is the prior probability of the class label y, and $P(x_1, x_2, ..., x_n)$ is a normalization 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.

F. 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.

G. 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:

$$S(x) = 1/(1 + e^{-}x) \tag{4}$$

where Y is the binary response variable, $X1, X2, ..., Xp$ are the input features, $\beta 0, \beta 1, \beta 2, ...,$ are the model parameters (also called coefficients or weights), and S() is the logistic function. The term $\beta 0$ is the intercept, and $\beta 1, \beta 2, ..., \beta p$ are the coefficients that determine the influence of each feature on the predicted probability.

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

H. 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.

VI. Zeek Implementation

Starting a new instance of Zeek

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

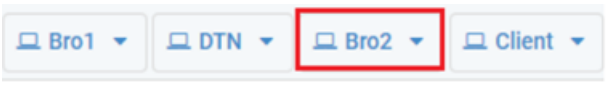

Fig. 1. Virtual environment creation

Step 2: Starting the terminal and created a new instance

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

Step 4: By TCP SYN scan traffic capture is started

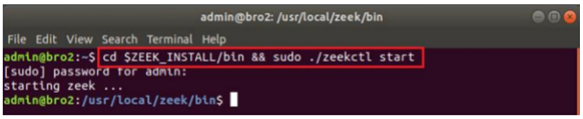

Fig. 2. Instance creation

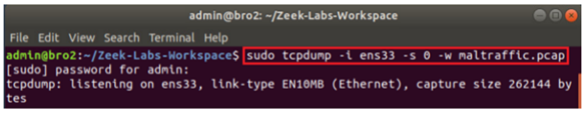

Fig. 3. Live packet capture

A. 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

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

VII. 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 logistic regression, 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 logistic regression.

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

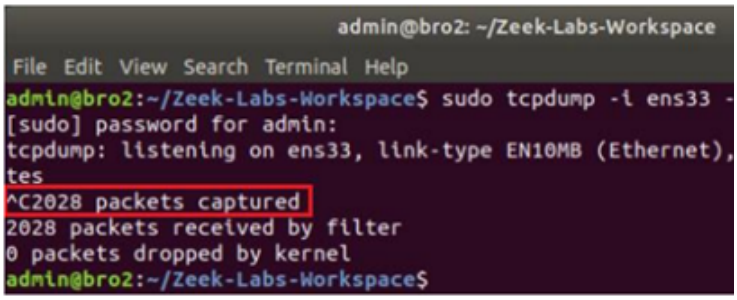

Fig. 4. TCP syn

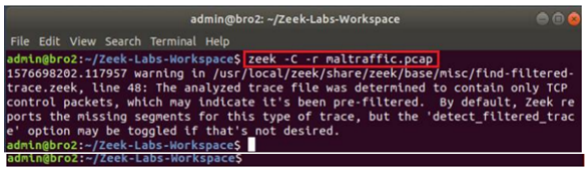

Fig. 5. Zeek log

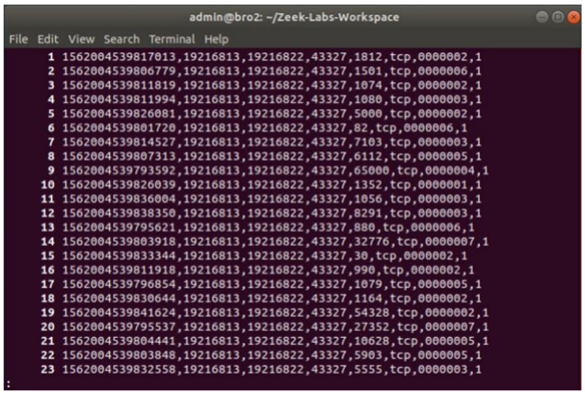

Fig. 6. Data generation - benign

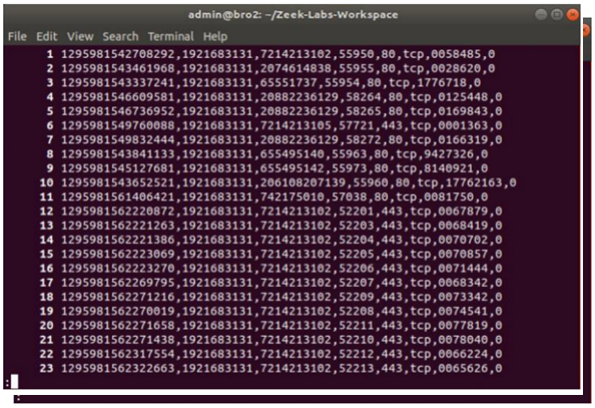

Fig. 7. Data generation - Malicious

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 logistic regression model trained on the training dataset achieved an accuracy of 98% when tested on the testing dataset.

This result indicates that Zeek, when 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 can be attributed to the effectiveness of the logistic regression algorithm. Logistic regression is a widely used algorithm for binary classification problems and has been shown to be effective in detecting fraud in network traffic. In addition, the accuracy achieved can also be attributed to the features used in the logistic regression 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 logistic regression model likely contributed to the high accuracy achieved.

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.

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.

VIII. Conclusion and Future Work

In conclusion, the use of Zeek 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 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

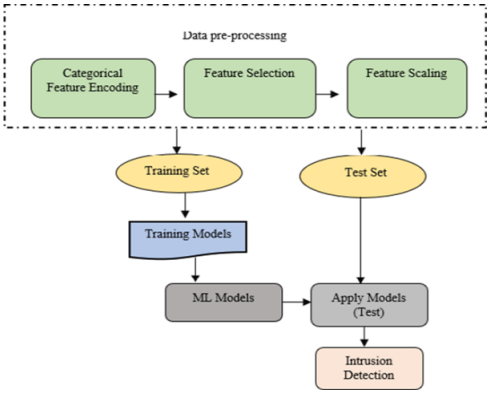

Fig. 8. Workflow

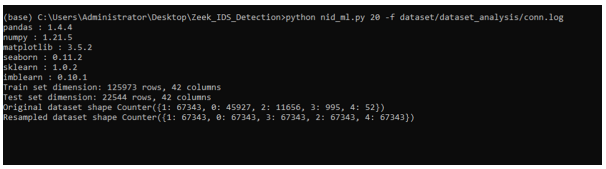

Fig. 9. Lodaing script

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

Fig. 10. Anomaly Detection Output

[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 protocolbased 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.

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

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

[21] D. E. Denning, "An intrusion-detection model," IEEE Trans. Softw. Eng., vol. 13, pp. 222–232, Feb. 1987.

[22] 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.