ISSN 2278-3091
Volume 10, No.3, May - June 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse811032021.pdf
https://doi.org/10.30534/ijatcse/2021/821032021

Detection of Abnormalities in Real-Time Computer Network Traffic Empowered by Machine Learning

Javairya Nadeem[1], Arfan Ali Nagra[2], Muhammad Asif[3], Aqsa Iftikhar[4]
[1] Lahore Garrison University, Pakistan, javairyakhan@gmail.com
[2] Lahore Garrison University, Pakistan, arfan137nagra@gmail.com
[3] Lahore Garrison University, Pakistan, drmuhammadasif@lgu.edu.pk
[4] Lahore Garrison University, Pakistan, aqsaiftikhar@lgu.edu.pk

ABSTRACT

This research discloses how to utilize machine learning methods for anomaly detection in real-time on a computer network. While utilizing machine learning for this task is definitely not a novel idea, little literature is about the matter of doing it in real-time. Most machine learning research in PC network anomaly detection depends on the KDD '99 data set and means to demonstrate the proficiency of the algorithms introduced. The emphasis on this data set has caused a lack of scientific papers disclosing how to assemble network data, remove features, and train algorithms for use in real-time networks. It has been contended that utilizing the KDD '99 dataset for anomaly detection is not appropriate for real-time network systems. This research proposes how the data gathering procedure will be possible utilizing a dummy network and generating synthetic network traffic by analyzing the importance of One-class SVM. As the efficiency of k-means clustering and LTSM neural networks is lower than one-class SVM, that is why this research uses the results of existing research of LSTM and k-means clustering for the comparison with reported outcomes of a similar algorithm on the KDD '99 dataset. Precisely, without engaging KDD '99 data set by using synthetic network traffic, this research achieved the higher accuracy as compared to the previous researches.

Key words: Anomaly Detection, Computer Networks, Data Generation, Machine Learning, Network Security, Real-Time.

1. INTRODUCTION

Computer security is a consistently advancing issue as innovation turns out to be progressively powerful. The strategies for the protection of data and data systems of yesterday cannot guarantee the security of the present systems. A dynamic piece of computer security is knowing whether a system has been or is being altered. These systems are called intrusion detection systems and are made to alleviate the dangers of system failure and misuse [1]. Network intrusion detection systems can be isolated into anomaly-based systems and signature-based systems. A signature-based system works by automatically making signatures that check for known attacks or malfunctions. The issue with this sort of system is that new or unknown attacks are difficult to deal with. Ongoing advances in network technology and the public accessibility of modern hacking instruments permit users with practically zero specialized understanding to submit complex system attacks [2], [3]. These attacks have gotten difficult to detect and making signature-based detection is a bulky and costly cycle. An anomaly-based detection system could be much more successful in this situation [4]. Anomaly-based systems rather attempt to recognize what an ordinary condition of the system looks like and reports when the system is not in this state. These systems are efficient at detecting unknown system behavior which can be resulted from either an attack or a system failure. Anomaly-based systems are frequently utilized in inter-connection with signature-based security systems to protect against as many dangers as could reasonably be expected. Machine learning has been utilized for anomaly-based intrusion detection systems for a considerable length of time. These sorts of algorithms are acceptable at detecting patterns in data, which causes them to perform well as a part of an anomaly-based intrusion detection system. Although, a great deal of examination on the algorithms in this field depends on utilizing a single data set, specified as the KDD '99 data set [4], [5]. This data set is presently 20 years of age and over the most recent twenty years, computer network systems and attacks have developed massively. The KDD '99 data set depends on the 1998 DARPA intrusion detection evaluation program's gathered data [6]. The data set was gathered more than 9 weeks on a system made to copy commonplace U.S air force LAN network traffic. The data set doesn't contain data dependent on singular packets sent over the network system. Rather every packet is appointed to a connection from where the data is determined. A connection is characterized as a grouping of TCP packets beginning and ending at some well-defined time. This data set doesn't contain some other protocol than TCP. Despite the fact that the KDD '99 data set is broadly utilized it has been criticized for having a lack of quality [7], [8], [9]. The nature of the data set is basic in machine learning and it has been contended that this lessens papers dependent on the dataset of KDD '99. When a dataset is assorted as the KDD '99 dataset is admissible, there exists a gap in the academia demonstrating the cycle from gathering network traffic to executing a live system. This project means to

help fill this gap by looking at preparing said traffic, training algorithms by utilizing the traffic as data and lastly assessing the algorithms on a real-time network archiving how this can be practiced.

2. BACKGROUND

2.1 Network Intrusion Detection System

Network Intrusion Detection Systems, NIDS, are used to detect malicious traffic in a computer network. The subject can be divided into two types, signature based detection and anomaly based detection [10], [11]. Signature based NIDS follow a static set of signatures and patterns to detect already known attacks. Anomaly based NIDS are used to classify normal traffic and abnormal traffic, where the latter is considered to be an anomaly in the network. They have detected anomalies by using a set of procedures and heuristics and this project focus on these different types of NIDS. Typically, anomaly based NIDS are developed by gathering data of the system and feeding it to an algorithm with the intention that the algorithm should be able to distinguish normal and anomalous data [11].

2.2 Network anomalies

In computer networks, an anomaly would be unexpected network traffic or traffic flows [12]. Anomalies can appear in any network for various reasons and do not necessarily mean that an intruder or malicious user has tampered with the network. Examples of anomalies could be a sudden unexpected spike in network congestion due to a router failure or a denial of service attack sending many more packets to a node than normal. Below are examples of three different network anomalies.

- Node Failure
  Node failure occurs when a node in the network suddenly stops responding. This can be because of a power outage, a hardware or software failure in the node or because of any other reason. The node is simply not active in the network any longer.
- Denial of Service Attacks
  DoS attacks is a cyber-attack in which machine and network resources are unavailable to its intended user. DoS attacks can also be used to increase network congestion and cause major slowdowns in network traffic. A DoS attack causes a noticeable difference in network traffic, as there is a sudden increase in connections to a specific port that are never resolved.
- Port Scan
  Port scanning is a technique where the user is attempting to find out what services are running on a remote machine. This is usually done to find vulnerable ports that the targeted machine is listening to and either close them, add security layers on top of the services or, in the case of a malicious user, exploit these ports.

2.3 Network Traffic Generation

Network traffic fluctuates relying upon what services are running on the connected systems in the network/topology. Preferably the network communications should be made to imitate some type of predefined system. This traffic can be created by using tools to create synthetic data, replaying pre-recorded network data or by having people perform actions over the network simulating real users. Synthetic data is data that has the same features and behaves the same way as real data but does not contain any significant information. Below is a short description of three network traffic generation tools.

- **Distributed Internet Traffic Generator (D-ITG)**
  D-ITG (Distributed Internet Traffic Generator) is a platform used to generate realistic network traffic. This tool is capable of using a variety of stochastic processes to simulate real-time network performance with varying delays and packet sizes using a large variety of probability distributions [1900]. D-ITG can imitate a large number of different applications over the network, such as online games or voice communication. It uses client-server functionality to send and receive network communication and also keeps a log of overall network communication that has been sent and received by D-ITG.
- **Iperf**
  Iperf, also known as iperf3 is an open-source tool for bandwidth benchmarking in computer networks. It uses client-server functionality to set up connections and can send packets using a variety of different protocols and parameters. When connecting to a server using iperf, it will attempt to send as much data as possible to the server and report back its throughput.
- **SourcesOnOff**
  SourcesOnOff was developed to help network engineers generate network traffic for testing and evaluation of different applications on computer networks. It attempts to generate the same type of traffic a network administrator would see in a local area network and on the internet using client-server functionality [13]. It supports sending packets with different delays and sizes using various probability distributions to mimic realistic network performance.

2.4 Feature Selection

Data must be ordered and labeled clearly to be considered useful when being collected and processed. This means that all data gathered must be sorted and put into context. When talking about feature selection the intent is to extract several characteristic features that would describe the data accurately. This allows the algorithms to process the data in a way that would make it useful for machine learning. There are several ways this can be done and it depends on how the data processing algorithms work. There are however indicators that help when selecting features such as variance. For example, a feature with zero variance would not be useful since it would never change and would thus be redundant.

2.5 Pre-processing

Algorithm to work correctly it requires a numeric representation with high-quality data set i.e. a data set that the algorithms understand with minimal inaccuracies [14]. The data can contain irregularities or other skewed information that would cause the algorithm to perform poorly. The pre-processing step is usually done by filtering out faulty data points. This can be done by removing the faulty data or correcting it. There is more to pre-processing data than filtering, such as normalizing the features since they can vary between a large ranges of values.

3. METHODOLOGY

For the implementation of the proposed methodology, we create a network environment of integrated systems by using machine learning techniques to detect the attacks and zombie PC.

3.1 System Overview

Ongoing cyber-attacks are started by utilizing progressive social engineering or by sending a targeted email to assault targets [15]. In the event that a host is infected by malicious code, an endeavor to speak with a C&C worker is made and a correspondence channel is framed utilizing Internet Remote Chat (IRC) or an HTTP protocol. When a communication channel is set up, the host gets an attack instruction from the C&C server or updates the file.

For this purpose, the utilization of machine learning techniques has been designed for the detection of zombie pc to monitor real-time network traffic and abnormalities. Figure 1 shows the overall structure of this research.

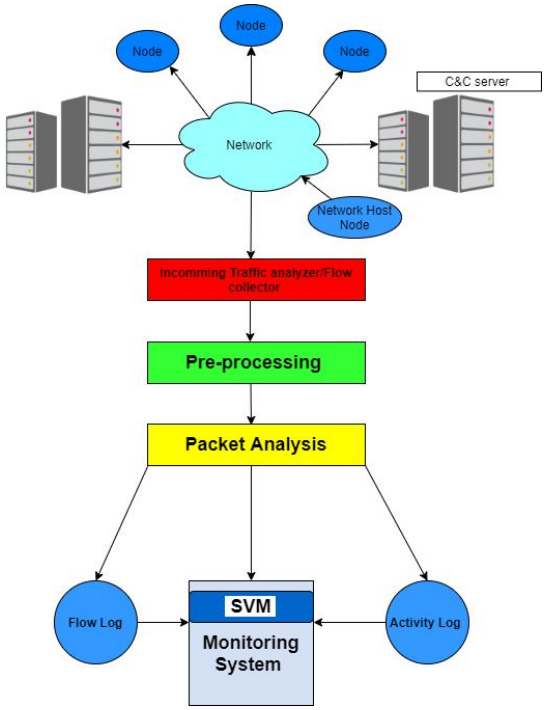

Figure 1: Proposed Methodology

3.2 The Network Structure for Generating and Gathering Network Traffic

For the detection of an anomaly, we create a setup of one or more nodes. We create a network in which one or more hosts are integrated, in this network one or more hosts performing some task over a network. In this setup, there are one or more hosts that are not visible by the host on the network. The purpose of these nodes is to collectively gather all the data in the network. These nodes are handled by the monitoring system whose task is to analyze data that come from the internet. Many surveillance nodes analyze the data. If so, they must communicate separately from the analyzed network and have some algorithm for reaching consensus of the network analyses. Figure 1, above shows the general structure of the network traffic collecting system.

3.3 Creating a Network Traffic

For the detection of an anomaly, generating traffic between nodes is the main part of this research. Realistic traffic is required to fulfill the machine learning and enough to perform machine learning data well. Many approaches were considered. Such a network system is to be made that can produce artificial network traffic. To evaluate network traffic, many tools are used that generate artificial network traffic to detect anomalies, for running programs that can generate synthetic network traffic that behaves like real network traffic. The tools that are utilized for generating synthetic network traffic are libpcap tool (which is used with a python API), D-ITG, iperf, and sourcesonoff. D-ITG was a poor tool for longer sessions of creating network traffic. Iperf is good for long sessions but it is usually built for bandwidth testing which causes network flooding with a high rate of packets per second. Sourcesonoff was able to produce a good variance of packet rates and was suitable for this research. Its performance was better than among the three tools.

Synthetic traffic proves to be beneficial for the research. For checking the accuracy of algorithms we intent to use self-made datasets. Our experimental results are based on KDD ’99 datasets. By using KDD ’99 datasets, the pre-processing techniques rely on information extracted per packet based on the network. This means that different pre-processing methods can be considered tested and evaluated. Some features were collected for pre-processing from the IP header of each packet. These features are based on the report by benferhat et al. [16]. Table 1 shows the features extracted from the IP headers of each captured network packet. The flag feature only contains SYN, ACK, and FIN flags. If the transport protocol is UDP. The flag field is left empty.

Table 1: Features extracted from the IP headers of each captured network packet

Features	Description
Protocol	Transport layer protocol.
Source address	IP-address of the source node.
Destination address	IP-address of the destination node.

Source port	The logical port on the source node for the connection.
Destination port	The logical port on the destination node for the connection.
Size	The size of the packets in bytes.
Timestamp	The network packets timestamp.
Flags	Status flag for the packet, if available.

3.4 Distinct Features by Creating Anomalies

We have arranged each anomaly in such a way that they differ from each other.

1. DoS attack causes an increase in network traffic. The DoS attack selected was an SYN flood that sends a lot of SYN packets to the targeted node. This attack was selected since it is easy to execute and causes a huge disturbance in the network flow which would make it perceptible in the dataset.
2. A node failure causes a decrease in network traffic. Node failure just needs to make a node not send or receive packets. This could be brought about by manually unplugging the node from the system, yet the arrangement ought to ideally be automated by software. The node could shut down the system interphase or essentially shut down the programs in control imparting over the network. Node failures are not complex and don't need more arrangement. The significant part is that the node failure makes the node to be inactive for sufficient time to be perceptible in the network.
3. A port scan can be designed to not cause any increase or decrease in the network traffic. Port scan was given a lot of ports to scan utilizing TCP ACK flag to separate it from SYN flood. At the point when a large number of packets are conveyed in a short time, it causes an increase in network traffic. This can cause an increase in packets sent to unknown ports just as an increase in ACK packets which should let the machine learning algorithm detect the port scan.

3.5 Preprocessing Raw Network Traffic

Data pre-processing is an essential part of this research that separates the raw network traffic. Capturing a network generates a lot of data and translates the raw network traffic into a dataset that can later be fed into a machine learning algorithm. Different tools are used to pre-process the data which analyzes this kind of data to detect malicious activities. Also, there are different techniques for the pre-processing of data, namely:

1. Packet-based numeric
2. Time series
3. connection

All these three techniques are viable for creating a dataset useful for machine learning. However, the time series pre-processing technique was the only one implemented in this research mainly due to the time constraint. Nevertheless, the drawback of the packet-based numeric technique creates a very large dataset that impacts the machine learning algorithms. While the connection technique needs a system that handles the unfinished connection and out-of-order packets.

3.6 Determining outliers and threshold for three machine learning techniques LSTM, SVM, K-means clustering

- **Long short-term memory neural network**

The LSTM network was selected because of a higher complex nature and its capacity to look at arrangements of the time series. This algorithm applies a self-supervised approach which implies that the algorithm is prepared to predict the following point in the time series. By doing so requires a sequence of previous time-series data points and utilizes the subsequent point as target output for error approximations. Since the LSTM is a sort RNN it stores information from past data points in the time series and its task can be depicted as in (1)

$$\{\vec{x}(1-h), \ldots\ldots, \vec{x}(1)\} \quad \text{predict}\, \vec{x}(1-1) \tag{1}$$

After the LSTM network is prepared on the network traffic training set, it is considered by checking the root mean square error (RMSE) for every one of the data points in the test set. Expect that $\xi(i)$ is the estimated data point, and $x(i)$ is the actual data point, at that point the root mean square is. As in (2).

$$RMSE = \sqrt{((\xi_1(i) - x_1(i)) + \cdots + (\xi_d(i) - x_d(i))} \tag{2}$$

Where *d* is the element of the dataset. Supposing that the data contains a couple of anomalous points a threshold that separates normal network traffic data and anomalous data is determined dependent on RMSE. While presenting another data point $\vec{x}$ it is delegated as anomalous or normal based on if the RMSE, of the expectation from the LSTM and the actual point, is less or more than the threshold as found in condition (3).

$$\vec{x} = \begin{cases} \text{normal} & \text{if RMSE} \leq T \\ \text{anomalous} & \text{if RMSE} > T \end{cases} \tag{3}$$

- **SVM (Support Vector Machine)**

Support vector machine is a supervised machine learning model that uses a classified algorithm for two group's classification problems. This causes a problem since the dataset in this research only contains a single class, namely, what is presumed to be normal network traffic. The one-class SVM however, handles the one-class problem by calculating a hyper-sphere around the part of the data it is fed. Normal data points are those which are inside the hyper-sphere and anomalous data points are those which are outside the hyper-sphere. Through this, the algorithm is considered to be suitable for anomaly detection.

The hyper-parameters to be turned for the one-class SVM are γ and ν, where $\gamma = 1/(2\sigma^{\wedge}2)$ is a parameter that takes the variance of the input dataset into account. A smaller γ value allows a larger variance in the input data to be mapped as similar. A larger γ value only allows a small variance in the input data and will only map data points close to each other as similar. And the ν parameter governs the upper bound on the fraction of margin errors and a lower bound on the fraction of support vector relative to the total number of training examples.

A low ν value penalizes data points that are outside the hyper-sphere while a higher ν value generalizes a better by accepting more outliers. There is no general solution to set the value of γ and ν and to find the proper value for hyper-parameters a range of different values needs to be tested and evaluated. In the live performance assessment, the one-class support vector machine outputs the best results.

- **K-means Clustering**

K-means clustering is an unsupervised technique that tries to express clusters in the data. By utilizing this algorithm to find a set of clusters such that they represent the normal network traffic. Squared Euclidean distance determines normal and abnormal network traffic from the nearest cluster center as shown in (4).

$$d(x) = \min \sum_{i=1}^{k} \|\vec{c}_i - \vec{x}\|^2 \quad (4)$$

Where *d* is the squared Euclidean distance, c_i is the position of the cluster centers and $\vec{x}$ is a data point. To be able to decide if a new point is anomalous or normal a threshold for the maximum distance squared to the nearest cluster is a set of all network traffic data points in S, the threshold, T, is selected such that a subset N⊑S will be classified as normal. Then T is chosen as in (5).

$$1 - P(d(x) < T) < \epsilon, \qquad \forall x \in X \quad (5)$$

Where € is the allowed percentage of the data point in S that might contain anomalous data.

4. COMPARATIVE RESULTS OF SVM, LSTM, AND K-MEANS CLUSTERING

The terms used for the evaluation of these three algorithms are false positive, false negative, true positive, and true negative. These values are utilized while calculating the F1-score, accuracy, and recall values.

1. Classifying normal data as anomalous is called false positive.
2. Classifying anomalous data as normal is called a false negative.
3. Classifying anomalous data as anomalous is called a true positive.
4. Classifying normal data as normal is called a true negative.

4.1 Starting Point

These results of the selected algorithms were compared with the same algorithms found in different papers.

1. The paper of L. Han [17] "Research of K-MEANS Algorithms Based on Information Entropy in Anomaly detection" was used for the comparison of k-means clustering.
2. Whereas the paper of Kim et al [18] "Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection" was considered for the comparison of the LSTM neural network.
3. The paper of Zhang et al [19] "An Anomaly Detection Model Based on One-Class SVM to Detect Network Intrusions" was studied for the comparison of One-class SVM.

4.2 K-means Clustering

The Experimental cluster size selected was 40. It took 4 hours long liv evaluation resulting in 15,540 time-series data points for the collection of results. The result of the comparison of k-means Clustering shows that the overall accuracy was 98.12%. Table 2 shows the classification of statics of k-means clustering.

Table 2: Classification of Statics of k-means Clustering

	Prediction Accuracy	**Total number**	**Percentage**
Normal	10101	10303	98.04%
SYN flood	1368	1370	99.85%
Port scan	2288	2321	98.58%
Node failure	1492	1546	96.51%

Live evaluation of k-means clustering with synthetic attacks. Table 3 shows the actual and predicted values of normal and anomalous data after live evaluation of k-means clustering with synthetic attacks and Table 4 shows the comparative Results that were illustrated from the research by L. Han [17] of k-means clustering.

Table 3: Actual and Predicted Values of Normal and Anomalous data

	Actual Values		**Predicted Values**	
	Normal data	**Anomaly data**	**Normal data**	**Anomaly data**
True positive		98.3%		98.3%
True negative	98.0%		98.0%	
False positive	2.0%			2.0%
False-negative		1.7%	1.7%	

Table 4: Comparative Results that were illustrated from the Research by L. Han [17] of k-means Clustering

	Target score	Achieved score	Difference
F1-score	93.35	97.25	3.9
Precision score	95.84	96.22	0.38
Recall score	90.98	98.30	7.32

4.2 LSTM (Long Short-Term Memory)

In the experimental section, selected 4_{LSTM} and 2_{DENSE}-model with a threshold of 0.79 to be evaluated on the live network. It took 7.71 hours providing 27,750 evaluated data points. 96.48% was the overall accuracy of the LSTM algorithm. Table 5 shows the classification statics of LSTM and Table 6 shows the comparative results that were illustrated from the research by Kim et al. [18] of LSTM

Table 5: Classification Statics of LSTM

	Prediction accuracy	Total number	Percentage
Normal	17266	18030	95.76%
SYN flood	2667	2714	98.27%
Port scan	4634	4728	98.01%
Node failure	2200	2277	96.62%

Table 6: Comparative Results that were illustrated from the Research by Kim et al. [18] of LSTM

	Target score	Achieved score	Difference
F1-score	94.11	95.11	-1
Precision score	98.66	92.56	6.1
Recall score	89.96	97.81	-7.85

Live evaluation of LSTM with synthetic attacks.

Table 7 shows the actual and predicted values of normal and anomalous data after live evaluation of LSTM with synthetic attacks.

Table 7: Shows the Actual and Predicted Values of Normal and Anomalous Data

	Actual values		Predicted values	
	Normal data	Anomaly data	Normal data	Anomaly data
True positive		97.8%		97.8%
True negative	95.8%		95.8%	
False positive	4.2%			4.2%
False negative		2.2%	2.2%	

4.4 One-Class SVM

In experimental results, the One-class SVM with a value of 0.001 and a y value of 0.04 was selected. It took 4.26 hours providing 15,349 evaluated data points. The overall accuracy of the one-class SVM algorithm was 98.6%. In Table 8 classification statics of one-class SVM is shown and figure 2 shows classification statics of SVM.

Table 8: Classification Statics of One-Class SVM

	Prediction accuracy	Total number	Percentage
Normal	10195	10302	98.96%
SYN flood	1593	1597	99.75%
Port scan	1860	1893	98.26%
Node failure	1482	1557	95.18%

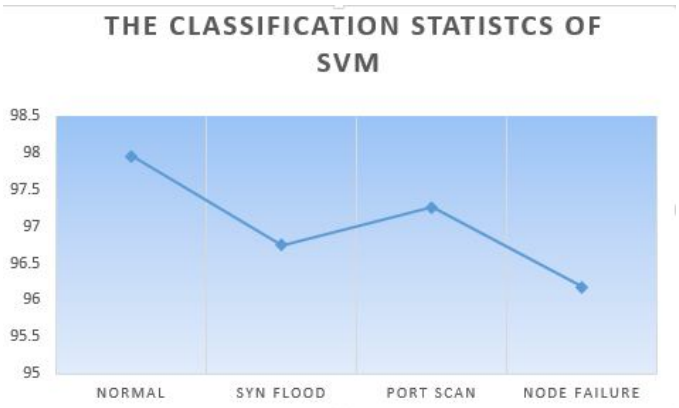

Figure 2. Classification Statics of SVM

Live evaluation of SVM with synthetic attacks.

Table 9 shows the actual and predicted values of normal and anomalous data after live evaluation of one-class SVM with synthetic attacks while Table 10 shows the comparative results.

Table 9: the actual and predicted values of normal anomaly data

	Actual values		**Predicted values**	
	Normal data	**Anomaly data**	**Normal data**	**Anomaly data**
True positive		97.8%		97.8%
True negative	99.0%		99.0%	
False positive	1.0%			1.0%
False negative		2.2%	2.2%	

Table 10: Comparative Results that were illustrated from the Research by Zhang et al. [19] of one-class SVM

	Target score	**Achieved score**	**Difference**
F1-score	98.56	98.48	0.8
Precision score	99.03	97.88	1.15
Recall score	97.17	97.02	0.15

5. COMPARISION BETWEEN DIFFERENT MACHINE LEARNING ALGORITHMS

The Table 11 shows the comparative results of different algorithms and figure 3 shows the comparison between different machine learning algorithms.

Table 11: Comparative Results between Different Algorithms

Method Name	**Evaluation Metrics Accuracy**	
Logistic Regression	Precision	0.64
	Recall	0.65
	F1- Score	0.61
	Training Accuracy	0.77
Gaussian Naïve Bayes	Precision	0.60
	Recall	0.53
	F1- Score	0.53
	Training Accuracy	0.73
KNN (k-nearest neighbor)	Precision	0.70
	Recall	0.65
	F1- Score	0.64
	Training Accuracy	0.70
Random Forest	Precision	0.59
	Recall	0.61
	F1- Score	0.59
	Training Accuracy	0.79
Proposed model SVM	Precision	0.99
	Recall	0.97
	F1- Score	0.98
	Training Accuracy	0.96
	Testing Accuracy	0.97

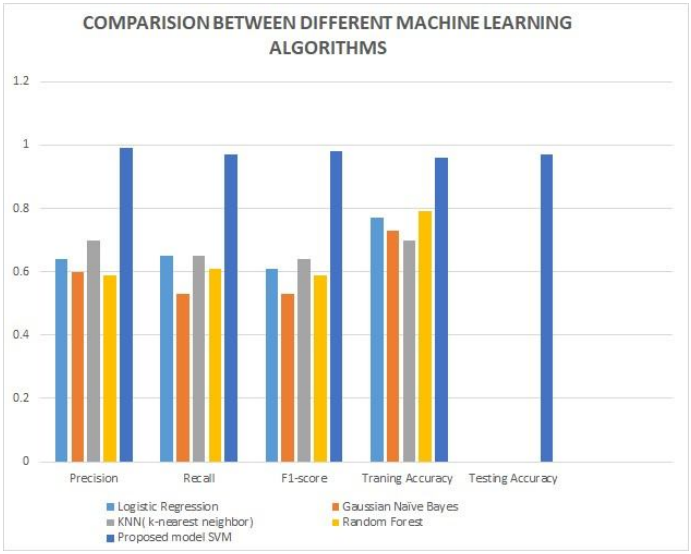

Figure 3: Comparative Results

6. CONCLUSION

In this research, the experimental results proved which algorithm performs best for the detection of an anomaly in a real-time synthetic network traffic environment. The results were similar to the ones found in reports using the KDD ’99 datasets. This determines the techniques clarified in this research can create an anomaly detection system with precision like the ones created utilizing the KDD '99 dataset without having similar disadvantages as when utilizing the KDD '99 dataset. The score presented in the result section shows that k-means clustering and the one-class SVM give almost the same results of achievement in the case of anomaly detection whereas the LSTM needs further improvements.

REFERENCES

[1] Warzyński, Arkadiusz, and Grzegorz Kołaczek. "Intrusion detection systems vulnerability on adversarial examples." *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018.

[2] Flood, Jason, Mark Denihan, Anthony Keane, and Fredrick Mtenzi. "Black hat training of white hat resources: The future of security is gaming." In *2012 International Conference for Internet Technology and Secured Transactions*, pp. 488-491. IEEE, 2012.

[3] Kao, Da-Yu, and Shou-Ching Hsiao. "The dynamic analysis of WannaCry ransomware." In *2018 20th International Conference on Advanced Communication Technology (ICACT)*, pp. 159-166. IEEE, 2018.

[4] Sommer, Robin, and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection." In *2010 IEEE symposium on security and privacy*, pp. 305-316. IEEE, 2010

[5] D. C. St. Clair. Learning programs. Volume 11, pages 19–22 Institute of Electrical and Electronics Engineers, October 1992.

[6] The association for computing machinery's special interest group on knowledge discovery and data mining. https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Tasks. Accessed: 2019-04-05.

[7] KDDCup '99 dataset (Network Intrusion) considered harmful. https://www.kdnuggets.com Accessed: 2019-04-17.

[8] Divekar, Abhishek, Meet Parekh, Vaibhav Savla, Rudra Mishra, and Mahesh Shirole. "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives." In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pp. 1-8. IEEE, 2018.

[9] Tavallaee, Mahbod, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. "A detailed analysis of the KDD CUP 99 data set." In *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1-6. IEEE, 2009.

[10] Warzyński, A., & Kołaczek, G. (2018, July). Intrusion detection systems vulnerability on adversarial examples. In *2018 Innovations in Intelligent Systems and Applications (INISTA)* (pp. 1-4). IEEE.

[11] Debar, H., Dacier, M., & Wespi, A. (2000, July). A revised taxonomy for intrusion-detection systems. In *Annales des telecommunications* (Vol. 55, No. 7-8, pp. 361-378). Springer-Verlag.

[12] Samrin, R., & Vasumathi, D. (2017, December). Review on anomaly based network intrusion detection system. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)* (pp. 141-147). IEEE.

[13] M. Alkasassbeh, G. Al-Naymat, and E. Hawari. Towards generating realistic snmp-mib dataset for network anomaly detection. *International Journal of Computer Science and Information Security ISSN 1947 5500*, Vol. 14 :(pp. 1162–1185), September 2016.

[14] Varet, A., & Larrieu, N. (2014, July). How to generate realistic network traffic. In *2014 IEEE 38th annual computer software and applications conference* (pp. 299-304). IEEE.

[15] Tankard, Colin. "Advanced persistent threats and how to monitor and deter them." *Network security* 2011, no. 8 (2011): 16-19.

[16] Benferhat, Salem, Karim Tabia, and Karima Sedki. "Preprocessing rough network data for intrusion detection purposes." 2007.

[17] Han, L. (2012, November). Research of K-MEANS algorithm based on information entropy in anomaly detection. In *2012 Fourth International Conference on Multimedia Information Networking and Security* (pp. 71-74). IEEE.

[18] Kim, J., Kim, J., Thu, H. L. T., & Kim, H. (2016, February). Long short term memory recurrent neural network classifier for intrusion detection. In *2016 International Conference on Platform Technology and Service (PlatCon)* (pp. 1-5). IEEE.

[19] Zhang, M., Xu, B., & Gong, J. (2015, December). An anomaly detection model based on one-class svm to detect network intrusions. In *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)* (pp. 102-107) IEEE.

[20] lbor, Ayei, Nitish Achar, and Purushottam Patil. "Machine Learning for Cyber Threat Detection." *International Journal* 9.1.1 (2020).