IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 5, May 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005228 |

Real Time Intrusion Detection System Using Machine Learning

Mr. K Azarudeen[1], Hariram Sridharan[2] , Hitesh Adhitya SL[3], Sudarsan N[4]

Assistant Professor, Dept. of CSE, Velammal College of Engineering And Technology, Madurai,Tamilnadu[1]

UG Student, Dept. of CSE, Velammal College of Engineering And Technology, Madurai, Tamilnadu[2,3,4]

ABSTRACT: The lack of a comprehensive network-based data set that can reflect modern network traffic situations, a wide variety of low-footprint intrusions, and deep structured information about network traffic is one of the primary research difficulties in this subject. However, multiple recent studies have shown that traditional data sets do not accurately reflect network traffic and modern low-footprint attacks in the current network security environment. This research investigates the establishment of a UNSW-NB15 data set to address the challenges of network benchmark data set unavailability. Existing and novel methods are utilised to generate the features of the UNSW-NB15 data set. This data set is available for research purposes. Using UNSW-NB15 and several Algorithms like SVM, RF and MLP, a model was trained, and performance of each algorithm was studied under various optimizers to create a reliable and efficient Intrusion Detection System.

KEYWORDS: UNSW-NB15 data set; Intrusion Detection System;NIDS; low footprint attacks; pcap files; testbed

I.INTRODUCTION

The fourth industrial revolution has altered our perceptions of modern living. Apart from personal computers, the vast network known as the Internet now includes mobile phones, tablets, IoT devices, and many more people who have network access and expertise that was once considered archaic.

The potential and prospects are unlimited; regrettably, the hazards and chances of malicious invasions are also limitless. Cybersecurity is the practise of defending systems, networks, and programmes against digital attacks, and Intrusion Detection is the subprocess of detecting these intrusions. NIDS (network intrusion detection systems) are intrusion detection systems that monitor network traffic and abnormalities. They frequently use a method called anomaly-based detection, which detects deviations from typical traffic patterns using Machine Learning.

II.RELATED WORK

It is not a novel concept to apply Machine Learning techniques in the development of intrusion detection systems.Knowing this, we decided to look for publications and conduct study on the subject.

We concentrated on the findings of two articles, both of which were based on research undertaken at the University of New Mexico.

The first, by Mahdi Zamani and Mahnush Movahedi [2], digs into the usage of Machine Learning techniques in the field of Intrusion Detection, using a more general approach that familiarised us with the problem we sought to address.Mukkamala, Janoski, and Sung[1] take a closer look at the problem, attempting to construct solutions using Support Vector Machines and Neural Networks on a DARPA dataset and comparing the results of the two methods.

We opted to incorporate some of these strategies into our project based on the work of these esteemed educators.We reviewed the article of Nour Moustafa and Abdelhameed Moustafa, the inventors of the UNSW dataset we utilised for training and testing, for the project's extension, the live data streaming phase.

This paper provided us with valuable insight into the data gathering and processing that led to the construction of the training and testing samples.

III. DATASET

A. General Description:

The dataset we used to train and test our Machine Learning models came from the Australian Centre for Cyber Security's (ACCS)[5] Cyber Range Lab.The dataset's creators created a hybrid of real modern regular activities and synthetic contemporary attack behaviours using the IXIA PerfectStorm tool.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005228 |

The dataset was chosen primarily because it makes use of the fields found in PCAP files as features.This means that the system may be used in real-world scenarios, with packets intercepted in the network with Tcpdump or Wireshark as input, converted to PCAP files, and then to csv format using a simple parsing script.

B. Specifics of the Data:

We already had a training set of 175341 instances and a testing set of 82332 instances in the dataset we acquired from the UNSW website.We removed several small discrepancies in the dataset, such as missing features, during the dataset cleaning process.The collection contains 43 PCAP file features, such as IP sources and destinations, ports, service kinds, and so on.

• Fuzzers • Analysis • Backdoors • DoS • Exploits • Generic • Reconnaissance • Shellcode • Worms are the attack categories provided as labels by the dataset (instead of normal packets).

C. Preprocessing:

Preprocessing and data preparation are two important aspects of Machine Learning.We opted to discover highly correlated features and remove them from the dataset first, in order to assure the optimum performance of the algorithms we will explain in the next part.We were able to eliminate features with at least 85 percent correlation using Will Koehrsen's FeatureSelector package [4], making our approach more cost and time effective.

Next, we utilise the Scikit Learn library's LabelEncoder class to convert the nominal values to numeric values so that the algorithm may execute arithmetic operations on them.

Finally, for easier convergence, we scaled our data in the range of (0,1) using Scikit Learn's MinMaxScaler class. Because SVMs are scale invariant, this process was not implemented for the SVM methodology.

IV. DATA STREAMING PIPELINE

A real-time data flow is required for the model we shall provide below.Data is created, moved, and removed in a fraction of a second, and if we want to achieve "real" intrusion detection, we must handle and evaluate data as soon as it passes through our network.For us, the best option was to build a pipeline with four phases/stages:

• Phase 1: Data gathering
• Phase 2: Feature extraction
• Phase 3: Processing and Dataframe extraction
• Phase 4: Classification

The pipeline's four phases can work on distinct threads at the same time, allowing us to manage up to four data batches each cycle.The pipeline is seen in the diagram below:

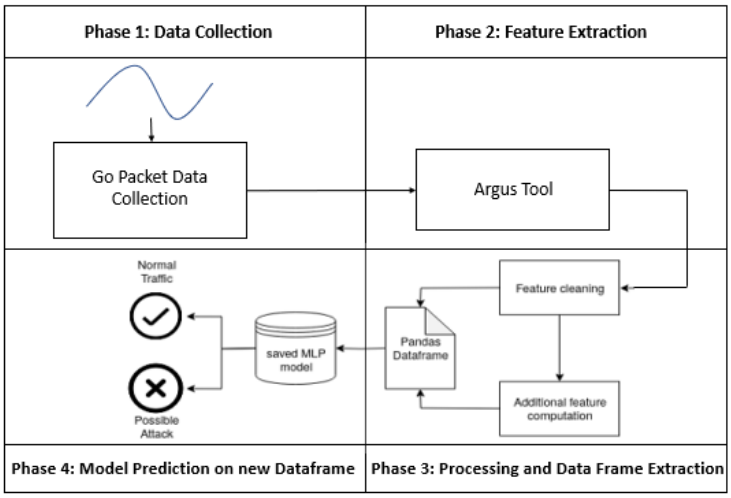

Fig 1. Pipeline overview

V. REAL TIME DATA COLLECTION

Using the tcpdump utility to sniff the network and extract information in the form of a.pcap file is a simple way to collect data travelling via the network.Anyone might later analyse the packets and extract useful information using Wireshark.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005228 |

However, the dataset we used for training, and thus the dataset we'll need to generate from the pipeline, is far more complex than processing apcap file and using its fields as features.
In order to gather packets and transfer them as bytes to the next phase for inspection and processing, we used gopacket, a Go lang package for decoding network packets.

A. Feature Extraction with Argus

The real-time data must create the fields that will be used as features in our machine learning models.After removing the strongly associated features, we are left with 28 features.

dur	proto	service	state
spkts	dpkts	rate	sttl
dttl	sload	dload	sinpkt
dinpkt	sjit	djit	swin
stcpb	dtcpb	tcprtt	smeansz
dmeansz	tans_depth	response_body_len	ct_srv_sr
ct_state_ttl	ct_dst_ltm	is_ftp_login	ct_flw_http_mthd

Table 1. Features used in training and testing

The light-gray coloured features are features that we can extract with Argus, a highly useful tool.
Argus reports on the transactions it discovers as periodic network flow data that is suited for historical and near real-time processing for forensics, trending, and alarm/alerting, according to the tool's wiki description.

As a result, we used argus' reports to do flow analysis on the data generated during the first pipeline phase and extract the majority of the attributes required for predicting an attack using the ra command (read argus),.
After argus has analysed the input, it will report in the standard output, where ra will analyse the output and generate the results - features - that we require.
This phase generates the majority of the characteristics we require for our model, however some cannot be generated by argus.

B. Additional feature computation

Table 1 shows that five of the traits are slightly darker in colour than the others.These are characteristics that largely contain flow data and require some additional data processing to produce.Apart from the ftp login function, which is self-explanatory, the other four features contain data that requires more than one packet to compute.
As the packets flow through the pipeline and are analysed, they also pass through a programming logical block in the third stage, which saves state on consecutive packets that satisfy a specific scenario.
By systematically comparing the appropriate fields of passing packets, ct srv src, for example, keeps track of consecutive packets relating to the same source IP and service. We were able to compute the last features needed to generate a Pandas Data frame that will be transmitted to the final and most significant step using basic Python routines.

We also eliminate packets (rows in the dataframe) containing NaN fields, convert nominal fields to numeric, and scale some values during this phase.

Before we get into how our model predicts on the data we've prepared in real time, it's important to read the sections below about the machine learning and neural network models we looked into, the experiments we ran and the results we got, and the model we chose as the best answer.

VI. ALGORITHMS AND BODY OF WORK

A. General Approach and Algorithms used

As targets, the dataset includes two columns: one with normal-not-normal values and the other with attack type values.We discovered that categorising by attack type and aggregating the data into normal and abnormal traffic produced better results than categorising by normal/abnormal traffic.
As a result, we've classified our task as a supervised multiclass classification problem.

We choose to focus on the issue using three algorithms based on the studies we mentioned:
• Multilayer Perceptron
• Random Forest
• Vector Support Machine
We'll look at the work that was done in each one and provide a score based on how well it performed.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005228 |

B. Support Vector Machine

The above-mentioned articles presented SVMs as the best performing Machine Learning technique for this type of problem.After evaluating numerous permutations using a trial-and-error technique, the best parameters we selected were 100 for the C value and 0.00001 for the gamma value.In addition, we used an RBF kernel, which is the most successful in general cases.

Though our dataset was huge, the time coefficient was a factor that made it unsuitable in our opinion when compared to the other two techniques.

The training phase took roughly 100 seconds using only about 12% of the training set, and the time would grow exponentially with more data as SVM tends to cross-calculate each point with the rest of the data.
In comparison to the other two methods, the time was significantly longer.

C. Random Forest

The random forest technique was not included in the research publications, but it was a technique we opted to investigate because of its speed in processing huge datasets.

After experimenting with numerous different options, we settled on 1000 estimators and 5 as the tree depth. The algorithm delivered a high-accuracy result in a short amount of time, with an accuracy of 81.336 percent on the binary classification problem of traffic normalcy and 68.087 percent on the attack type classification problem.In general, we thought the algorithm supplied us with quite accurate results, especially considering how simple it was to create and execute.

D. Multilayer Perceptron

The results of the Multilayer Perceptron method were the best of the three algorithms.

• A dense input layer of 300 neurons and a rectifier linear unit activation function were used in the neural network's architecture.
• A dense hidden layer of 150 neurons with a rectifier linear unit activation function.
• A dense output layer of 10 output classes with a softmax activation function.

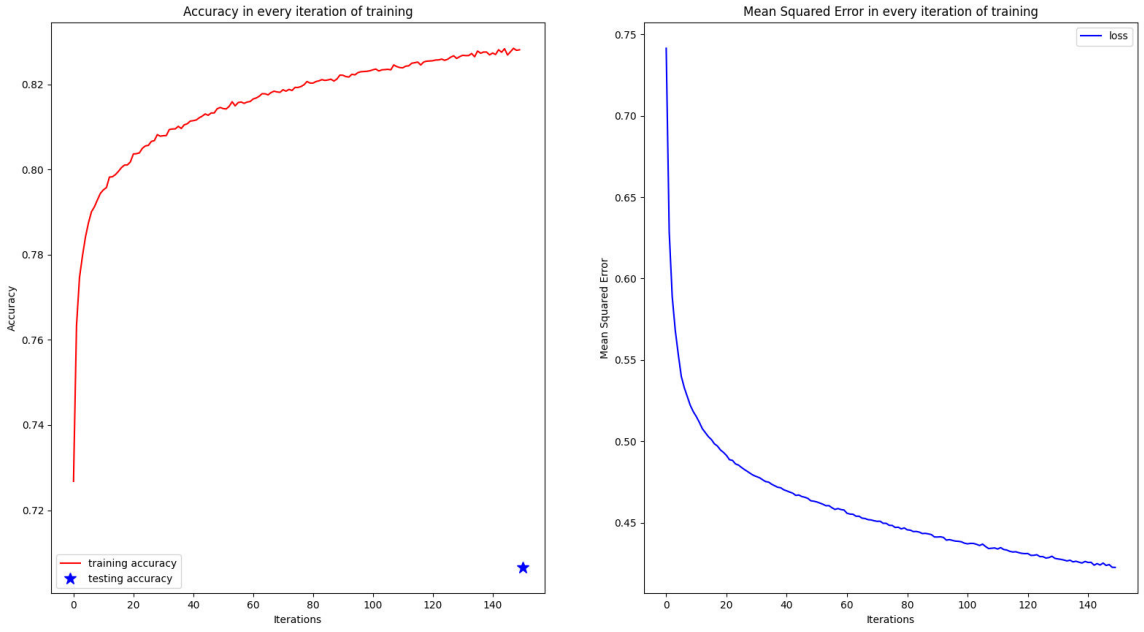

Fig 2. Plots of the Accuracy of atk type and the Mean Square Error by Iteration

Among the optimizers in Keras, Adam produced the best results and thus the model was built using Adam and a categorical cross-entropy loss function, and it was trained in batches of 150 cases across 150 iterations.The normalcy classification had a problem-high accuracy of 83.96 percent, while the assault type classification had an accuracy of 82.81 percent.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005228 |

	T	F
T	24517	12483
F	1184	44148

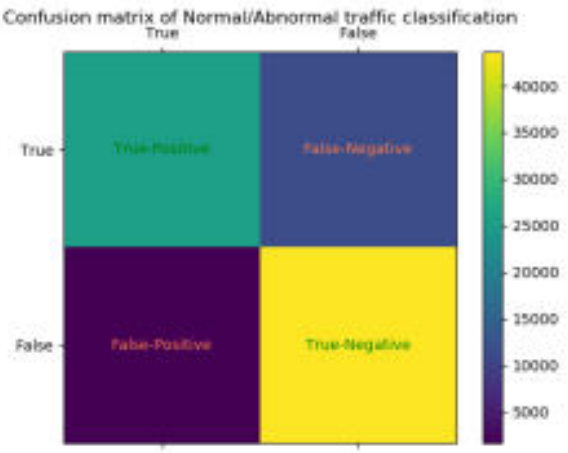

Fig 3. Confusion Matrix heat map

E. Methodology Comparisons

An accuracy-based comparison of the three algorithms is offered below to provide a better understanding of which method is judged more appropriate.

Table 2. Accuracy of every algorithm used in the NIDS project

TESTS	NORMAL/ABNORMAL CLASSIFICATION	ATTACK TYPE CLASSIFICATION
MLP	83.96%	82.81%
RF	81.33%	68.08%
SVM	76.10%	68.08%

The Multilayer Perceptron produces the best results of the three, as shown in the table, and we also present a bar plot to visually validate that MLP performed better. It is critical for this challenge that we have as few attacks classed as normal as possible.As a result, we included a confusion matrix statistic, which, as we can see, only produced roughly 1.4 percent false positive classifications.

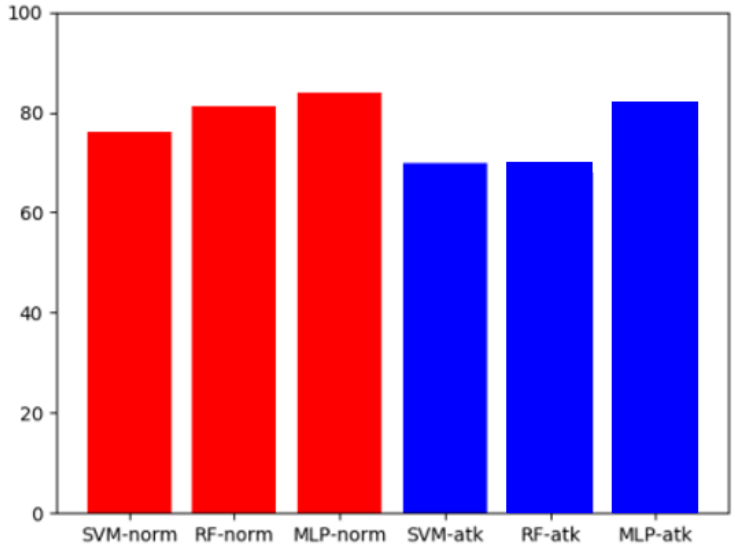

Fig 4. Bar plot for accuracy comparison between each algorithm

VII. PREDICTING IN REAL TIME

The paper's final point of interest is, of course, how this IDS model can forecast real-life occurrences.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005228 |

We talked a lot about a pipeline in section 3 that would allow us to predict data in real time and be actively secure by recognising network intrusions.After determining that MLP gives the best results for our situation, we exported a model and its weights as separate files, allowing us to reuse the model without having to train it each time.

After being successfully loaded using custom Python routines, the dataframe created in phase 3 of the pipeline (containing one row-instance every round) is utilised as input to this model.

The model, which is contained in an infinite while loop, predicts on these occurrences in a sequential manner and outputs the result of the prediction in the standard output.If a potential attack is detected, the programme notifies us of the classified attack type and exports the specific occurrence in a csv file for further testing to determine whether we have an attack or a false negative result.

Fig5. Sample output of the real time prediction. The instances classified as attacks are added to a .csv file for more analysis.

VII. CONCLUSION

We chose to use Machine Learning and Neural Networks to create a "smart" system that will analyse packet metadata and make security predictions for the network it is connected to. The objective is to contribute to the field of autonomous security, by developing a system that can successfully handle network security challenges for a single user or a company. The simulation results shows that the Multi Layer Perceptron is the best algorithm with Adam optimizer for Intrusion detection with least false positives and an attack prediction accuracy of 82.81%. Hence it is employed in the real time intrusion detection system.

REFERENCES

1. Srinivas Mukkamala, Guadalupe Janoski, Andrew Sung. Intrusion Detection: Support Vector Machines and Neural Networks. New Mexico Institute of Mining and Technology, 2002.
2. Mahdi Zamani, Mahnush Movahedi Machine Learning Techniques for Intrusion Detection. University of New Mexico, 2013.
3. Nour Moustafa, Abdelhameed Moustafa. Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic University of New South Wales, Australia, June 2017.
4. Feature Selector library : Will Koehrsen https://github.com/WillKoehrsen/feature-selector
5. UNSW-NB15 Dataset, ACCS https://www.unsw.adfa.edu.au/unsw-canberracyber/cybersecurity/ADFA-NB15-Datasets/
6. Gopacket package: packet decoding for Go language.https://godoc.org/github.com/google/gopacket
7. The argus tool. https://www.qosient.com/argus/publications.shtml

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.165

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details