

ENHANCING INTRUSION DETECTION SYSTEMS USING GENETIC ALGORITHM: A NOVEL APPROACH FOR NETWORK SECURITY

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

NAGARJUNA GK
(Reg No - 40110816)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A++" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Nagarjuna GK (40110816)** who carried out the Project entitled "**ENHANCING INTRUSION DETECTION SYSTEMS USING GENETIC ALGORITHM: A NOVEL APPROACH FOR NETWORK SECURITY**" Under my supervision from November 2023 to April 2024.

Jany
5/4/24
Internal Guide

Dr. S. L. JANY SHABU, M.Tech., Ph.D.

Lakshmanan
Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on 16/04/2024

Wan
16/04/24
Internal Examiner

M
16/4/24
External Examiner

DECLARATION

I, Nagarjuna GK (40110816) hereby declare that the Project Report entitled **Enhancing intrusion detection systems using genetic algorithm: A novel approach for network security** done by me under the guidance of **Dr.S.L.JANY SHABU, M.Tech., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 05/04/2024
PLACE: Chennai



SIGNATURE OF THE CANDIDATE

DECLARATION

I, **Nagarjuna GK (40110816)** hereby declare that the Project Report entitled **Enhancing intrusion detection systems using genetic algorithm: A novel approach for network security** done by me under the guidance of **Dr.S.L.JANY SHABU, M.Tech., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr.L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. L. Jany Shabu M.Tech, Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Intrusion detection systems (IDSs) are currently drawing a great amount of interest as a key part of system defence. IDSs collect network traffic information from some point on the network or computer system and then use this information to secure the network. To distinguish the activities of the network traffic that the intrusion and normal is very difficult and to need much time consuming. An analyst must review all the data that large and wide to find the sequence of intrusion on the network connection. Therefore, it needs a way that can detect network intrusion to reflect the current network traffics. In this study, a novel method to find intrusion characteristic for IDS using genetic algorithm & machine learning of data mining technique was proposed. Method used to generate of rules is classification by genetic algorithm of decision tree. These rules can determine of intrusion characteristics then to implement in the genetic algorithm as prevention. so that besides detecting the existence of intrusion also can execute by doing deny of intrusion as prevention.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
	2.1 Inferences from literature survey	5
	2.2 Open problems in Existing System	6
3	REQUIREMENTS ANALYSIS	7
	3.1 Feasibility Studies/Risk Analysis of the Project	7
	3.2 Software Requirements Specification Document	9
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 Selected Methodology or process model	11
	4.2 Architecture / Overall Design of Proposed System	12
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	19
	4.4 Project Management Plan	21
	4.5 Transition/ Software to Operations Plan	21
5	RESULTS AND DISCUSSION	23
6	SUMMARY AND CONCLUSION	26
	REFERENCES	27
	APPENDIX SOURCE CODE	30
	RESEARCH PAPER	60

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
4.1	Flow diagram	17
4.2	Architecture diagram	17
4.3	Sequence diagram	18
5.1	Graphical user interface	24
5.2	Model trained view	24
5.3	Attack detection and classification	25
5.4	Graphical representation	25

CHAPTER 1

INTRODUCTION

With the rapid expansion of Internet in recent years, computer systems are facing increased number of security threats. Despite numerous technological innovations for information assurance, it is still very difficult to protect computer systems. Therefore, unwanted intrusions take place when the actual software systems are running. While we are benefiting from the convenience that the new technology has brought us, computer systems are exposed to increasing security threats that originate externally or internally. Despite different protection mechanisms, it is nearly impossible to have a completely secured system.

Therefore, Intrusion detection is becoming an important technology that monitors network traffic and identifies network intrusions such as anomalous network behaviors, unauthorized network access and malicious attacks to computer systems. When an intruder attempts to break into an information system or performs an action not legally allowed, that is called as an intrusion. Intruders can be divided into two groups, external and internal. The former refers to those who do not have authorized access to the system and who attack by using various penetration techniques. The latter refers to those with access permission who wish to perform unauthorized activities. An Intrusion detection system is a system for detecting intrusions and reporting them accurately to the proper authority [3]. There are two general categories of intrusion detection systems: misuse detection and anomaly detection. Misuse detection system detects intruders with known patterns, and anomaly detection systems identify deviations from normal network behaviors and alert for potential unknown attacks [5]. Some IDS integrate both misuse and anomaly detection and form hybrid detection systems. The IDSs can also be classified into two categories depending on where they look for intrusions. Fig.1 Shows the Computer Network with Intrusion Detection Systems. A host-based IDS monitors activities associated with a particular host, and a network-based IDS listens to network traffic.

A number of soft computing-based approaches have been proposed for detecting network intrusions [1]. Soft computing refers to a group of techniques that exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve robustness and low solution cost. When used for intrusion detection, soft computing techniques are often used in conjunction with rule based expert systems acquiring expert knowledge [4] where knowledge is represented as a set of if-then rules.

Approaches for intrusion detection can be broadly divided into two types: misuse detection and anomaly detection. In misuse detection system, all known types of attacks (intrusions) can be detected by looking into the predefined intrusion patterns in system audit traffic. In case of anomaly detection, the system first learns a normal activity profile and then flags all system events that do not match with the already established profile. The main advantage of the misuse detection is its capability for high detection rate with a difficulty in finding the new or unforeseen attacks.

The advantage of anomaly detection lies in the ability to identify the novel (or unforeseen) attacks at the expense of high false positive rate. Network monitoring-based machine learning techniques have been involved in diverse fields. Using bi-directional long-short-term-memory neural networks, a social media network monitoring system is proposed for analyzing and detecting traffic accidents.

Generally, intrusion detection systems are divided into two main categories: host-based IDS (HIDS) and network-based IDS (NIDS). Network-based intrusion detection system checks and controls all incoming and outgoing traffic at one network component. This happens by placing one capture tool (sensor) like one sniffer on the choke point of the segment.

CHAPTER 2

LITERATURE SURVEY

This section briefly summarizes some of the applications of soft computing techniques for intrusion detection. However, a number of GA based IDSs are discussed in the later part of the paper in order to compare and contrast those work with our work. GAs and GP have been used for network intrusion detection in different ways. Some approaches directly use GAs to derive the classification rules [6,7], while some others use different AI methods for acquisition of rules, where GAs are used to select appropriate features or to determine the optimal parameters of some functions [4,8]. The early effort of using GAs for intrusion detection can be dated back to 1995, when Crosbie et al. [2] applied the multiple agent technology and GP to detect network anomalies. Each agent monitors one parameter of the network audit data and GP is used to find the set of agents that collectively determine anomalous network behaviors. This method has the advantage of using many small autonomous agents, but the communication among them is still a problem. Also, the training process can be time-consuming if the agents are not appropriately initialized. Bridges et al. [1] develop a method that integrates fuzzy data mining techniques and genetic algorithms to detect both network misuses and anomalies. In most of the existing GA based IDSs, the quantitative features of network audit data are either ignored or simply treated, though such features are often involved in intrusion detection. This is because of the large cardinalities of quantitative features.

The authors implemented a way to include quantitative features by introducing fuzzy numerical functions. Their preliminary experiments show that the inclusion of quantitative features and the fuzzy functions significantly improved the accuracy of the generated rules. In this approach, a GA is used to find the optimal parameters of the fuzzy functions as well as to select the most relevant network features. Lu et al. [7] present an approach that uses GP to directly derive a set of classification rules from historical network data. The approach employs the support confidence framework as the fitness function and is able to generally detect or precisely classify network intrusions. However, the use of GP makes implementation more difficult

and more data or time is required to train the system. Li [6] suggested a GA-based method to detect anomalous network behaviors. Both quantitative and categorical features of network data are included when deriving classification rules using GA. The inclusion of quantitative features may lead to increased detection rates. However, no experimental results are available yet. Xiao et al. [10] present an approach that uses information theory and GA to detect abnormal network behaviors. Based on the mutual information between network features and the types of network intrusions, a small number of network features are closely identified with network attacks.

Then a linear structure rule is derived using the selected features and a GA. The use of mutual information reduces the complexity of GA, and the single resulting linear rule makes intrusion detection efficient in real-time environment. However, the approach considers only discrete features. Literature survey shows that the Intrusion detection models insisted for R2L attacks failed to demonstrate desirable performance with high detection and low false alarm rates using KDD data set. This paper shall analyze selected DOS and R2L attacks. This paper studies warez master and smurf attacks as representative instances from the R2L and DOS category. Typical and relevant features must be observed present in the KDD data set [5] that can help with the detection of these attacks.

Wattanapongsakorn et al. [22] suggested a network-based Intrusion Detection and Prevention System (IDPS). The purpose of this system is to effectively detect known attack types and to take immediate action against attacks. The suggested approach can be used with different machine learning techniques and tested on an online network environment. The results show that the suggested IDPS can recognize normal events from attacks within seconds with high accuracy and automatically block the victim's computer network against attacks. Additionally, they applied the C4.5 Decision Tree algorithm with a insisted approach to detect unknown attack types and this algorithm can work effectively when faced with unknown types of network attacks. However, this study can be further improved by developing the approach for the detection of unknown attacks as well as the detection of known attacks.

Amaral et al. [23] implemented a network-based intrusion detection system for IPv6-enabled wireless sensor networks. The insisted system detects attacks by using traffic signatures and abnormal behaviors. insisted system consists of two components PPPSniffer and Finger2IPv6. In the suggested system, network nodes selected as observers are located by the intrusion detection system. In this way, packets exchanged in neighbors are observed and possible attack attempts are detected. The observed messages are compared with the rule set created by NIDS. If a match occurs, an alarm is generated and sent to the Event Management System. With this insisted system, possible misbehaviors can be detected instead of detecting predefined attacks. However, the system should be improved by adding new detection rules.

2.1 INFERENCES FROM LITREATURE SURVEY

Based on the literature survey, it is evident that various soft computing techniques, particularly Genetic Algorithms (GAs) and Genetic Programming (GP), have been successfully applied to network intrusion detection. These techniques have been utilized in different ways, including directly deriving classification rules, selecting features, and optimizing parameters of functions. Some early approaches date back to the mid-1990s, demonstrating the long-standing interest in using GAs for this purpose.

Additionally, it was observed that incorporating quantitative features, which are often critical in intrusion detection, is challenging due to their large cardinalities. However, innovative solutions, such as introducing fuzzy numerical functions, have been suggested to address this issue. This inclusion of quantitative features has shown promising results in improving the accuracy of generated rules.

Furthermore, certain studies focused on specific types of attacks, such as R2L and DOS attacks, and identified relevant features for their detection. Notably, the paper by Wattanapongsakorn et al. insisted a network-based Intrusion Detection and Prevention System (IDPS) capable of effectively detecting known attack types and taking immediate action against them. The system demonstrated high accuracy in distinguishing normal events from attacks.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Anomaly Detection Accuracy: The accuracy of anomaly detection can be improved. Anomalies can be diverse and context-dependent, making it challenging to distinguish between legitimate unusual behavior and actual attacks.

Reducing False Positives: Minimizing false positives is crucial in order to avoid alert fatigue and ensure that security teams focus on genuine threats. Fine-tuning algorithms and enhancing feature selection can contribute to reducing false alarms.

Multi-Modal Attacks: Detecting attacks that use a combination of different techniques (e.g., network-based and host-based) remains a challenge. Developing approaches that can effectively identify and respond to multi-modal attacks is essential.

Adversarial Attacks: Adversarial attacks involve attempts to deceive or manipulate the intrusion detection system. Research in robust machine learning techniques and adversarial training is necessary to mitigate the impact of such attacks.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

Technical Feasibility:

Hardware: The project requires a robust computing infrastructure capable of handling large volumes of network traffic data. Existing resources should be evaluated for adequacy or additional resources may need to be procured.

Software and Tools: Availability of necessary software tools and licenses for genetic algorithm implementation, machine learning, and data mining.

Datasets: Access to relevant and representative datasets for training and testing the enhanced IDS.

Expertise:

Skills Required: A team with expertise in genetic algorithms, machine learning, and data mining is essential. If in-house expertise is not available, arrangements for training or hiring skilled personnel must be made.

Economic Feasibility:

Budget:

Cost Estimation: Detailed cost estimation including hardware, software licenses, datasets, and any additional resources needed for the project.

ROI Analysis: Evaluate potential benefits in terms of enhanced security and potential cost savings from preventing intrusions.

Funding:

Source of Funding: Identify potential sources of funding, whether internal budget allocation, external grants, or partnerships.

Operational Feasibility:

Integration:

Integration with Existing Systems: Evaluate the compatibility and potential impacts of integrating the enhanced IDS with the current network infrastructure. Consider any necessary adjustments or modifications.

User Training: Assess the need for training end-users (security analysts, administrators) to effectively utilize the enhanced IDS.

Risk Analysis:

Technical Risks:

Algorithm Complexity: Genetic algorithms and machine learning techniques can be complex. Adequate expertise is crucial to ensure successful implementation.

Data Quality: The effectiveness of machine learning models heavily relies on the quality of training data. Steps must be taken to address potential issues with noisy or incomplete data.

Economic Risks:

Budget Overruns: Consider potential unforeseen costs in terms of hardware, software, or additional resources. Implement measures to monitor and control expenses.

Security Risks:

Over-reliance on Automation: Depending too heavily on automated prevention measures could lead to a lack of human oversight, potentially allowing new types of attacks to go unnoticed.

Malicious Actors Adapting: As the IDS improves, attackers may adapt their tactics to bypass the system, necessitating ongoing updates and adaptations.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION

Operating System: Windows 7 above

Language: Python

Tool: PyCharm

Library: Python (version 3.6.5)

Keras (version 2.1.6)

TensorFlow (version 1.7.0)

IMPLEMENTATION:

HOW TO INSTALL PYTHON IDE

Below is a step-by-step process on how to download and install Python on Windows:

Step 1) To download and install Python, visit the official website of Python <https://www.python.org/downloads/> and choose your version. We have chosen Python version 3.6.3

Step 2) Once the download is completed, run the .exe file to install Python. Now click on Install Now.

Step 3) You can see Python installing at this point.

Step 4) When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

HOW TO INSTALL PYCHARM

Here is a step-by-step process on how to download and install PyCharm IDE on Windows:

Step 1) To download PyCharm visit the website JetBrains and click the “**DOWNLOAD**” link under the **Community Section of Pycharm**.

Step 2) Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.

Step 3) On the next screen, Change the installation path if required. Click “Next”.

Step 4) On the next screen, you can create a desktop shortcut if you want and click on “Next”

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

The Prediction module aims to implement intrusion detection based on the trained model. It involves the following steps:

Feature Selection for Real-Time Detection: The objective is to choose a minimal set of features while maintaining high detection rates. This allows for efficient real-time detection of intrusions.

User Interface (GUI): A Graphical User Interface (GUI) will be provided for user interaction. This interface will facilitate the submission of network traffic data for intrusion detection. And user uses the interface to communicate and able to find different types of attacks.

Normal Data Submission: Users can submit normal network traffic data through the GUI to the trained model. The system will process this data and generate an output. In the interface the user trains the model with the normal data and once the system gets trained then the model is able to find the intrusions.

Attack Data Submission: once the model is trained users can also submit attack data for evaluation. The system will classify the type of attack (DoS or Probe) and display the results.

Graphical Representation: The system will generate a bar graph to visually represent the intrusion detection results. This graph will provide a clear indication of whether an intrusion has been detected and its type. The graphical representation contains three columns which represents three data (Normal, DOS, probe).

Log Generation: The system will generate a log file containing detailed information about the intrusion detection process. This log file can be saved for future reference or analysis. Users would easily ably known about the intrusion detection process using the log file.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

DATASET COLLECTION

NSL-KDD Data Set

- DoS is an attack that tries to shut down traffic flow to and from the target system.
- Probe or surveillance is an attack that tries to get information from a network. The goal here is to act like a thief and steal important information, whether it be personal information about clients or banking information.
- The feature types in this data set can be broken down into 4 types:
 - ✓ 4 Categorical (Features: 2, 3, 4, 42)
 - ✓ 6 Binary (Features: 7, 12, 14, 20, 21, 22)
 - ✓ 23 Discrete (Features: 8, 9, 15, 23–41, 43)
 - ✓ 10 Continuous (Features: 1, 5, 6, 10, 11, 13, 16, 17, 18, 19)

PREPROCESSING:

- In data normalization, the value of each continuous attribute is scaled between 0 and 1 such that the result of attributes does not dominate each other.
- The transform(data) method is used to perform scaling using mean and std dev.

MODEL TRAINING:

- In the Decision Tree algorithm, the attributes are tested on internal nodes, the outcomes of the tests are represented by branches, and leaf nodes hold labels of the classes.
- Attribute selection methods are used for identifying nodes. Those selected attributes minimize the information that is required for tuple classification in the resulted partition.

- Hence, reflecting the minimum uncertainty or impurity in those partitions. Therefore, minimizing the projected number of tests required for tuple classification.
- In this research, ID3 algorithms utilize entropy class to determine which attributes should be queried on, at every node of those decision trees.

Genetic Algorithm (GA) field is one of the up-coming fields in computer security, especially in Intrusion Detection Systems (IDS). For the purpose of processing network data in real time and performing efficient intrusion detection, the most important piece of information should be extracted that can be employed for efficient detection of network attacks. Principal Component Analysis (PCA) which is also known as Karhunen Loe've transform, is used in order to extract the most relevant features of the data. The goal of PCA is to decrease the dimension of the data by making an effort to discern a few orthogonal linear combinations of the variables that have the largest variance. Although an effective machine learning algorithm has been devised in the intrusion detection fields and related work, improving the detection rate with low false alarm is still required. As compared to other approaches, the new GA approach brings about higher detection rates and lower false alarm in identifying anomaly-based network intrusions.

Genetic Algorithms are search algorithms which function based on the principles of natural selection and genetics. GA develops a population of initial individuals to a population of high-quality individuals, where each individual signifies a solution of the problem to be solved. Each individual is called chromosome and comprises predetermined number of genes. The quality of each rule is measured by a fitness function as the quantitative representation of each rule's adaptation to a particular environment.

The procedure starts from an initial population of randomly generated individuals. The population is then evolved for a number of generations while gradually improving the qualities of the individuals in the sense of increasing the fitness value as the measure of quality. During each generation, three basic genetic operators are sequentially applied to each individual with certain probabilities, i.e., selection, crossover and mutation.

Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection.

Genetic algorithms are used in artificial intelligence like other search algorithms are used in artificial intelligence to search a space of potential solutions to find one which solves the problem

“Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, looking for optimal combinations of things, solutions you might not otherwise find in a lifetime.”

three main component or genetic operation in generic algorithm are crossover, mutation and selection of the fittest.

The basic process for a genetic algorithm is:

Initialization - Create an initial population. ...

Evaluation - Each member of the population is then evaluated and we calculate a 'fitness' for that individual. ...

Selection - We want to be constantly improving our populations overall fitness.

KDD Cup '99 dataset is used to verify the validity of this approach. As some of the network characteristics have higher possibilities to be involved in network intrusions, PCA approach is used to identify these characteristics. The PCA algorithm was implemented and deployed over the training dataset in order to define the features that are most frequently involved in a machinery of an attack.

According to the results, three features out of forty-one are selected to describe each connection of KDD Cup '99 dataset. The purpose was to select the smallest possible number of the features while maintaining high detection rate of intrusions. As such, detection could be performed as a real-time one. Table1 represents the features selected as well as their explications. Every feature represents one gene of the chromosome.

Table 1: Selected network features

Name of feature	Explication	No. of genes
Duration	Length (number of seconds) of the connection	1
Src_bytes	Number of data bytes from source to destination	1
dst_host_srv_error_rate	Percentage of connections that have "SYN" errors	1

As one byte is being used to represent every feature, i.e., every gene, a chromosome that represents each individual is composed of three bytes.

Every rule for intrusion detection is simple if-then clause. Features from [Table 1](#) are connected using an and function thus forming the conditional part of a rule. The result of every rule is the confirmation of an intrusion. For example, one rule could be:

- If (duration = "1" and src_bytes= "0" and dst_host_srv_error_rate= "0") then intrusion

To determine a fitness value of each rule, the following fitness function is deployed:

$$\text{Fitness} = a/A - b/B \quad \text{Fitness function} \quad (1)$$

where, a is the number of correctly detected attacks, A is the whole number of attacks in the training dataset, b is the number of normal connections that are falsely identified as attacks, i.e., false-positives and B is the total number of normal connections in the training dataset. Scale of fitness values is [-1, 1], where -1 and 1 represent the lowest and highest values, respectively. High detection rate and low rate of false-positives contribute to a high fitness value. On the other hand, low detection rate and high rate of false-positives bring about a low fitness value.

The algorithm for generating new rules is performed as follows. The first step is initialization of an initial population during which each gene is given a random value. Then the parameters of genetic algorithm (crossover and mutation rate, size of

population, end of evolution of rules) are identified and the network audit data is being loaded. Next, the initial population is being evolved for a number of generations. In every generation, the quality of every rule, i.e., fitness value, is calculated according to the fitness function, then a number of rules with the highest fitness values are selected and the genetic operators (crossover and mutation) are finally performed with a certain probability. The output of the algorithm generates rules for intrusion detection.

MODEL TESTING

EVALUATION METRICS

- A confusion matrix is used for the comparison of the performance of machine learning algorithms. This matrix is used for the creation of different metrics by the combination of the values of True Negative (TN), True Positive (TP), False Negative (FN) and False Positive (FP). Below are some of the performance measures to evaluate models by the use of the confusion matrix.
- Accuracy shows the correctness or closeness of the approximated value to the actual or true value of the model which means a portion of the total samples that are classified correctly.
- Precision shows which portion of relevant instances is actually positive among the selected instances.
- Recall or True Positive Rate (TPR) calculates the fraction of actual positives that are correctly identified.
- F1-score is interpreted as the harmonic mean of precision and recall means it combines the weighted average of precision and recall

PREDICTION

- The purpose was to select the smallest possible number of the features while maintaining high detection rate of intrusions. As such, detection could be performed as a real-time one.
- Submit normal data in trained model of GUI and plot the graph for the intrusion detection which display the bar graph and save log to display the output.

- Submit attack data in trained model of GUI and plot the graph for the intrusion detection which display the bar graph of dos and probe attack and also, we can save log to display the output.

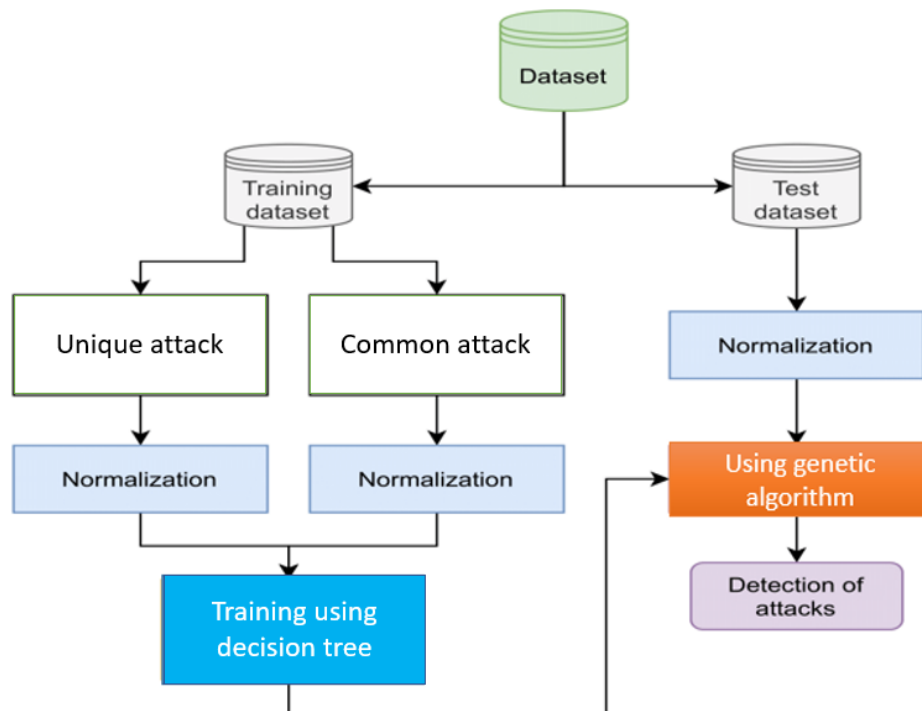


Fig 4.1: Flow Diagram

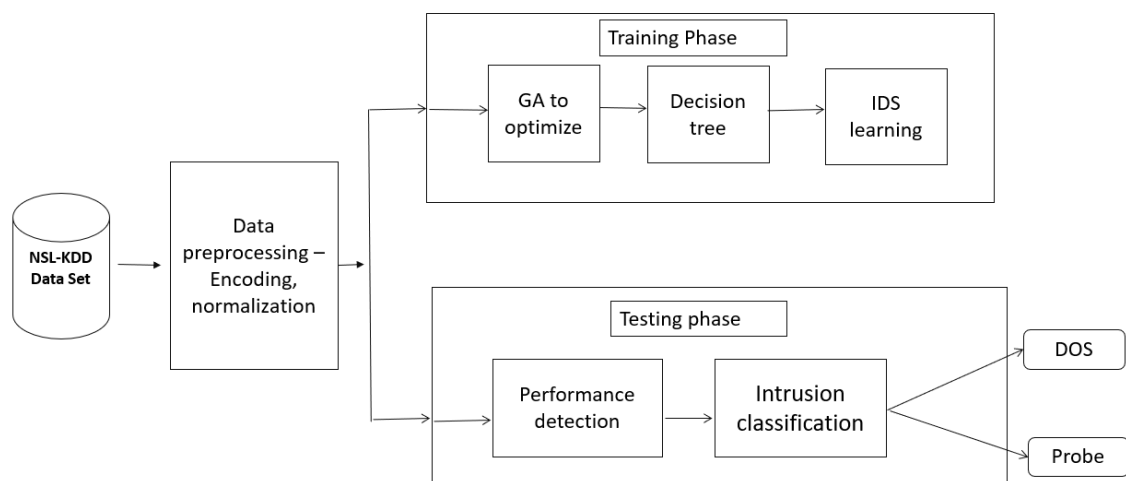


Fig 4.2: Architecture Diagram

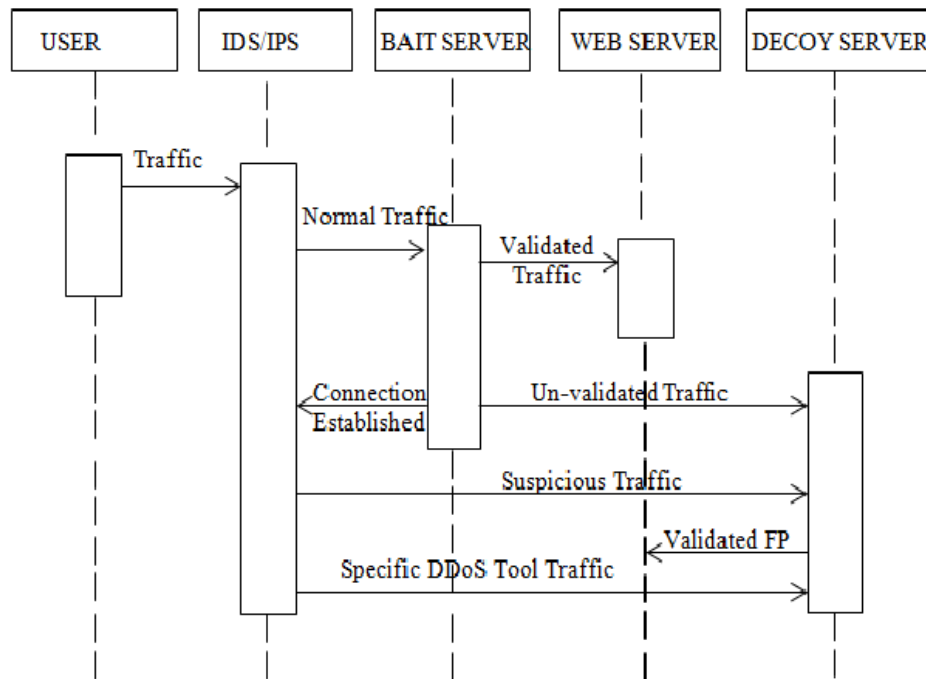


Fig 4.3: Sequence Diagram

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

1. User Interface (GUI):

The GUI provides an intuitive platform for users to interact with the system. It allows users to submit network traffic data for intrusion detection.

2. Dataset Collection Module:

This module is responsible for acquiring and preparing the dataset for training and testing the intrusion detection model. In this case, the NSL-KDD dataset is utilized, which contains data related to different types of network activities, including normal and attack scenarios.

3. Preprocessing Module:

The preprocessing module normalizes the data to ensure that the values of each continuous attribute are scaled between 0 and 1. This prevents any single feature from dominating the others, which is crucial for accurate intrusion detection.

4. Model Training Module:

a. Decision Tree Algorithm:

The Decision Tree algorithm is implemented to classify network traffic into different categories. It utilizes attribute selection methods to determine which attributes to query at each node, ultimately minimizing uncertainty or impurity in the partitions.

b. Genetic Algorithm (GA):

Genetic Algorithms are used to evolve a population of rules for intrusion detection. The GA employs selection, crossover, and mutation operators to gradually improve the quality of rules over multiple generations.

c. Principal Component Analysis (PCA):

PCA is used to extract the most relevant features from the data, reducing its dimensionality. This helps in efficient detection of network attacks.

5. Model Testing Module:

This module involves evaluating the performance of the trained model using evaluation metrics such as accuracy, precision, recall, and F1-score. A confusion matrix is used to compare the performance of different machine learning algorithms.

6. Prediction Module:

The Prediction module is designed for real-time intrusion detection. It allows users to submit normal and attack data through the GUI, and provides graphical feedback indicating the detection results.

The testing plan for the proposed intrusion detection system involves several stages to ensure its effectiveness and reliability:

1. Unit Testing:

Test individual modules for correctness and accuracy. Verify that each module performs its specific function as expected.

2. Integration Testing:

Verify that different modules integrate seamlessly and communicate effectively. Ensure that data flows smoothly between modules.

3. Dataset Testing:

Use a variety of test datasets, including normal and attack data, to evaluate the system's ability to accurately classify network traffic.

4. Preprocessing Validation:

Verify that the preprocessing module scales data appropriately, ensuring that continuous attributes do not dominate one another.

5. Model Training Validation:

Validate the Decision Tree algorithm, Genetic Algorithm, and Principal Component Analysis for their ability to extract meaningful features and generate effective rules for intrusion detection.

6. Model Testing and Evaluation:

Assess the performance of the trained model using evaluation metrics like accuracy, precision, recall, and F1-score. Verify that the system can correctly identify intrusions while minimizing false positives.

4.4 PROJECT MANAGEMENT PLAN

4.1 Milestones

Data Collection and Preprocessing (Week 1-2)

Model Training and Testing (Week 3-5)

GUI Development (Week 6-7)

Real-Time Intrusion Detection (Week 8-9)

Log Generation (Week 10)

4.2 Dependencies

Data Collection depends on dataset availability.

Model Training depends on completed data preprocessing.

GUI Development depends on available features for real-time interaction.

4.5 TRANSITION/ SOFTWARE TO OPERATIONS PLAN

Hardware and Software Requirements

- Ensure that the hardware infrastructure meets the system requirements, including processing power, memory, and storage capacity.
- Verify that all necessary software components, including operating systems, libraries, and dependencies, are installed and configured.

Data Integration

- Establish procedures for integrating the IDS with existing network infrastructure and data sources.

User Training

- Conduct training sessions for end-users, including security analysts and administrators, to familiarize them with the system's features and functionalities.

Deployment Process

Software Installation

- Install the IDS software on designated servers or network devices according to the deployment architecture.

Configuration and Customization

- Configure the system settings, including network parameters, data sources, and any specific rules or policies required for the organization.

Testing and Validation

- Conduct thorough testing to ensure that the IDS functions as expected in the operational environment. This includes both simulated and real-world traffic scenarios.

Integration with Security Infrastructure

- Integrate the IDS with existing security tools and systems, such as firewalls, SIEMs, and incident response platforms.

CHAPTER 5

RESULTS AND DISCUSSION

Differentiating between network traffic activities that constitute an intrusion and those that are regular requires a lot of time and effort. An analyst must look through all the huge and diverse data in order to determine when exactly the network link was compromised. A technique of network intruder identification is necessary for reflecting the actual network activity at present. As a result of the development of public and private cloud computing, networks have today become a crucial component of public infrastructures. The conventional networking strategy has gotten very complicated. This complexity has made it more difficult to link data centers, integrate businesses, and develop new, innovative services within a single data center. It has also made it more difficult for the Internet to flourish in general. The genetic algorithm, which makes use of natural selection, is widely used machine learning methods for intrusion detection. The leaf node indicates whether a classifier's output during the decision node denotes checking a single property of the offered examples, whereas the classification phase belongs to the normal or intrusion (any of the potential attacks) category. There was a recommendation that the decision tree machine learning data mining approach be used to identify attack traits for IDS. The genetic algorithm of a decision tree is a method for generating rules. Manual or automatic intrusion detection is available. IDS must be able to handle a substantial and continuously growing volume of alerts and events. It is more important to use decision trees. It is becoming more and more important to have automated systems for monitoring and responding to intrusions. Hence it can be inferred that usage of IDS, genetic algorithm is a great approach to get better insights of the security vulnerabilities in the network, and decision tree helps us in the classification process and the synergy of these things would be a great approach in network security.

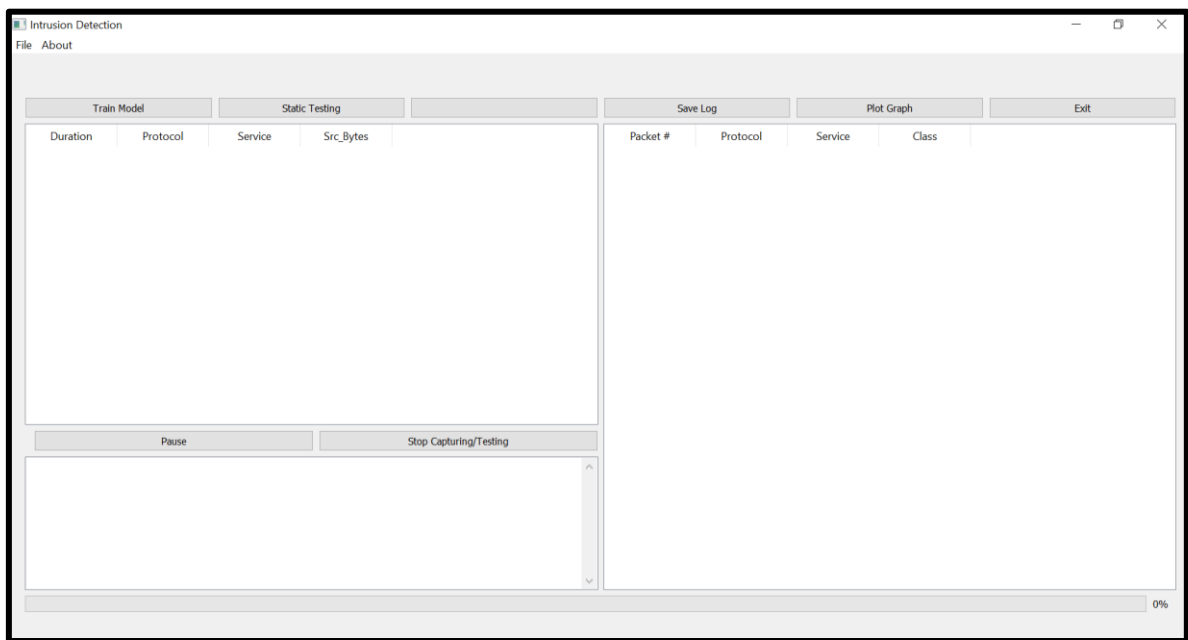


Fig 5.1: Graphical user interface

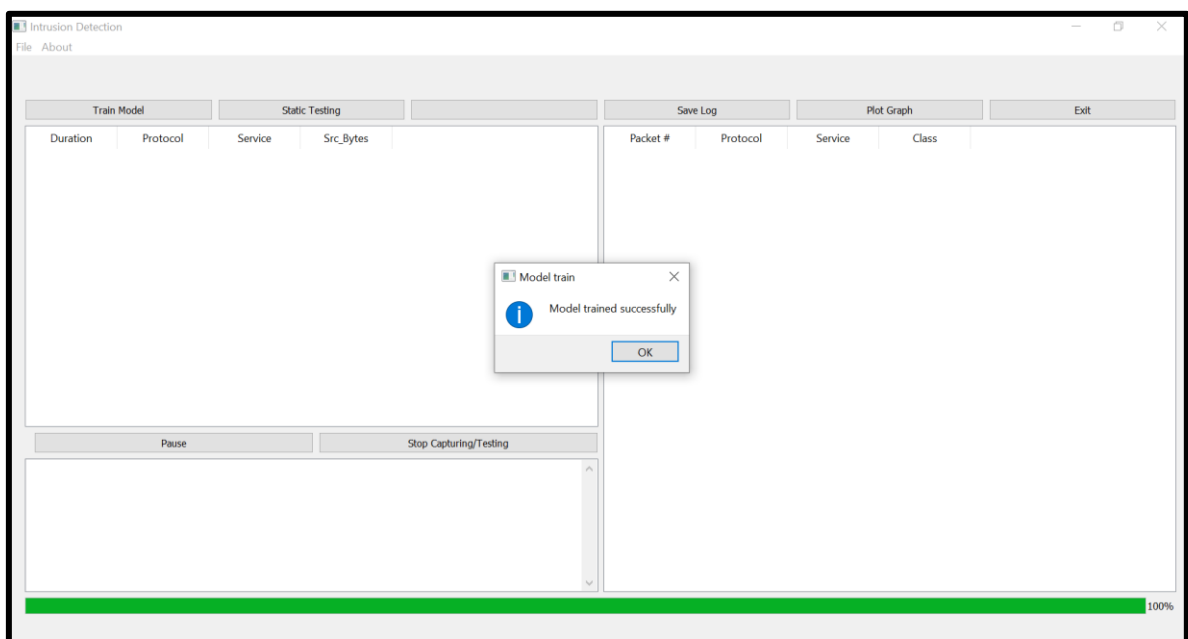


Fig 5.2: Model trained view

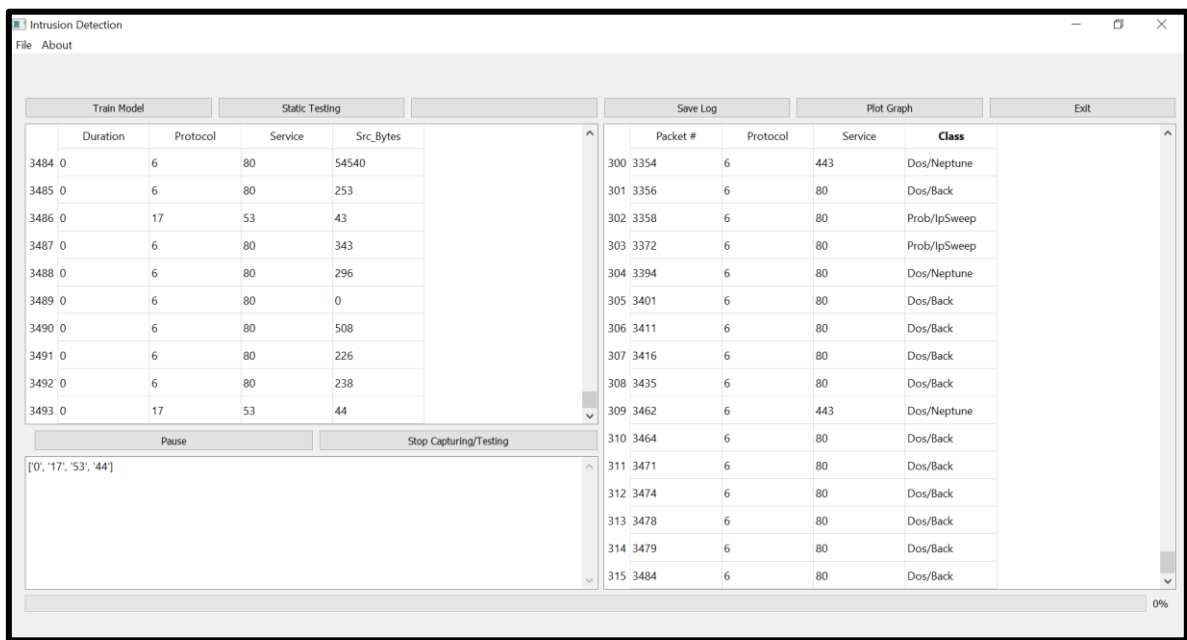


Fig 5.3: Attack detection and classification

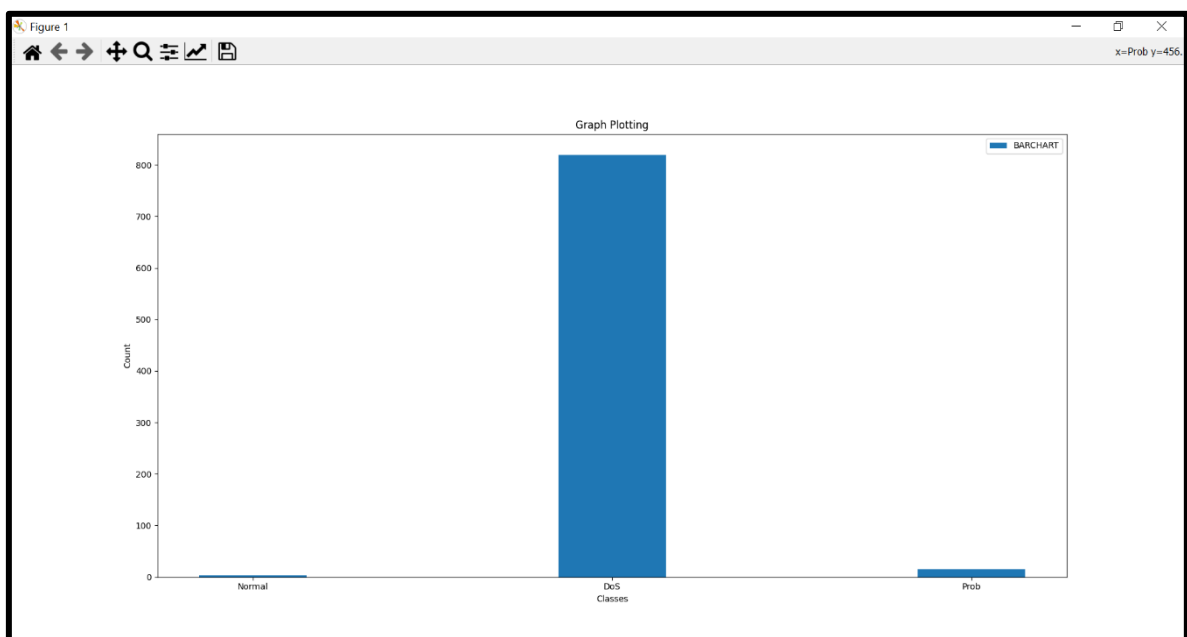


Fig 5.4: Graphical representation

CHAPTER 6

SUMMARY AND CONCLUSION

In this study, intrusion detection is carried out using a genetic algorithm and a decision tree technique. The suggested approach's software implementation is shown. Using a genetic algorithm, classification criteria for intrusion detection were created, and Key features of interconnection networks were identified using principal component analysis. It is simple to develop and maintain because the GA-approach accurately and clearly displays the required rules and fitness functions. Additionally, given the right attack taxonomy and training data-set, the system's adaptability allows it to serve multiple purposes. Because the point of incursion detection is to catch attacks as they happen and stop them before they can do any harm, the method doesn't care how the attacks are classified. The benefits of employing this technique for intrusion detection instead of a complimentary technique, which is frequently employed with other soft-computing techniques, are a high rate of assault detection and a low percentage of false positives. The system simply makes use of three network connection features while retaining high detection rates, allowing it to carry out the intrusion detection procedure more quickly and be used with high-speed networks.

References

- [1] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A new feature selection model based on ID3 and bee's algorithm for intrusion detection system," p. 8.
- [2] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.
- [3] Choudhary S, Kesswani N. 2020. Analysis of KDD-Cup' 99, NSLKDD and UNSW-NB15 datasets using deep learning in IoT. *Procedia Computer Science* 167(2019):1561-1573.
- [4] D. Agrawal, C. Agrawal, and H. Yadav, "A Machine Learning Based Intrusion Detection Framework Using KDDCUP 99 Dataset," vol. 4, no. 6, p. 11.
- [5] D. M. Abdulqader, A. M. Abdulazeez, and D. Q. Zeebaree, "Machine Learning Supervised Algorithms of Gene Selection: A Review," vol. 62, no. 03, p. 13, 2020.
- [6] D. M. Manimekalai and G. Anupriya, "A Novel Intrusion Detection System using Data Mining Techniques," vol. 6, no. 6, p. 5, 2019.
- [7] Dahiya P, Srivastava DK. 2018. Network intrusion detection in big dataset using spark. *Procedia Computer Science* 132(1–3):253-262.
- [8] Dlamini G, Fahim M. 2021. DGM: a data generative model to improve minority class presence in the anomaly detection domain. *Neural Computing and Applications* 33(20):13635-13646.
- [9] Faker O, Dogdu E. 2019. Intrusion detection using big data and deep learning techniques.
- [10] Lianming Zhang, Kui Liu, Xiaowei Xie, Wenji Bai, Baolin Wu, Pingping Dong, A data-driven network intrusion detection system using feature selection and deep learning, *Journal of Information Security and Applications*, Volume 78, 2023, 103606, ISSN2214-2126, <https://doi.org/10.1016/j.jisa.2023.103606>.
- [11] L. Yang, A. Moubayed, I. Hamieh and A. Shami, "Tree-Based Intelligent Intrusion Detection System in Internet of Vehicles," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013892.

- [12] M. Nobakht, V. Siva Raman, and R. Borelli, "A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using Open Flow," presented at the 2016 11th International Conference on Availability, Reliability and Security (ARES), 2016.
- [13] M. S. Essayed, N. Le-Khan, S. Dev., and A. D. Jurua, "Machine Learning Techniques for Detecting Attacks in SDN," in 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), 19-20 Oct. 2019 2019, pp. 277-281, doi: 10.1109/ICCSNT47585.2019.8962519.
- [14] Pampapathi B M, Nageswara Guptha M, M S Hema, Towards an effective deep learning-based intrusion detection system in the internet of things, Telematics and Informatics Reports, Volume 7, 2022, 100009, ISSN 2772-5030, <https://doi.org/10.1016/j.teler.2022.100009>.
- [15] Ramkumar M. P., P.V. Bhaskar Reddy, J.T. Thirukrishna, Ch. Vidyadhari, Intrusion detection in big data using hybrid feature fusion and optimization enabled deep learning based on spark architecture, Computers & Security, Volume 116, 2022, 102668, ISSN 0167-4048, <https://doi.org/10.1016/j.cose.2022.102668>.
- [16]. R. Vinayakumar, Mamoun Alazab, K. P. Soman. "Robust Intelligent Malware Detection Using Deep Learning," IEEE Access, vol. 7, pp. 46717- 46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
- [17] R. Baimukashev, A. Kamalkhan, A. Kazybek and M. Begenov, "Intrusion Detection System for Wireless Networks," 2021 16th International Conference on Electronics Computer and Computation (ICECCO), Kaskelen, Kazakhstan, 2021, pp. 1-5, doi: 10.1109/ICECCO53203.2021.9663787.
- [18] R. Taheri, M. Ahmadzadeh, and M. R. Kharazmi, "A New Approach For Feature Selection In Intrusion Detection System," p. 15.
- [19] R. Vijayanand, D. Devaraj, and B. Kannapiran, "A Novel Deep Learning Based Intrusion Detection System for Smart Meter Communication Network," in 2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), Tamilnadu, India, Apr. 2019, pp. 1–3.
- [20] S. A. Albelwi, "an intrusion detection system for identifying simultaneous attacks using multi-task learning and deep learning," 2022 2nd international conference on computing and information technology (iccit), tabuk, saudi arabia, 2022, pp. 349-353, doi: 10.1109/iccit52419.2022.9711630.

[21] X. Y. Li, R. Tang and W. Song, "Intrusion Detection System Using Improved Convolution Neural Network," 2022 11th International Conference of Information and Communication Technology (ICTech)), Wuhan, China, 2022, pp. 97-100, doi: 10.1109/ICTech55460.2022.00028.

APPENDIX

A. SOURCE CODE

```
from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtGui import QIcon, QPixmap

from PyQt5.QtWidgets import
qApp, QFileDialog, QMessageBox, QMainWindow, QDialog, QDialogButtonBox, QVBox
Layout, QHeaderView, QMessageBox

import os

import time

import pyshark

import matplotlib.pyplot as plt

import threading

import packet as pack

import tree

import Preprocess as data

import classifier


class Ui_MainWindow(object):

    def __init__(self):

        self.tree_classifier = classifier.DecisionTree()

        self.packet = pack.Packet()

        self.trained = False

        self.stop = False
```

```

self.threadActive = False

self.pause = False

def plot_graph(self):

    x = ['Normal','DoS','Prob']

    normal,dos,prob = self.tree_classifier.get_class_count()

    y = [normal,dos,prob]

    plt.bar(x,y,width=0.3,label="BARCHART")

    plt.xlabel('Classes')

    plt.ylabel('Count')

    plt.title('Graph Plotting')

    plt.legend()

    plt.show()


def train_model(self):

    try:

        train_dataset, train_dataset_type =
QFileDialog.getOpenFileName(MainWindow, "Select Training Dataset","", "All Files
(*);;CSV Files (*.csv)")

        if train_dataset:

            os.chdir(os.path.dirname(train_dataset))

            test_dataset, test_dataset_type =
QFileDialog.getOpenFileName(MainWindow, "Select Testing Dataset","", "All Files
(*);;CSV Files (*.csv)")

            if train_dataset and test_dataset:

                generation = 0

```

```

train_dataset = data.Dataset.refine_dataset(train_dataset, "Train
Preprocess.txt")

test_dataset = data.Dataset.refine_dataset(test_dataset, "Test
Preprocess.txt")

#Start tree

ga =
tree.tree(train_dataset,test_dataset,population_size=5,mutation_rate=65)

ga.initialization() # if error occur due to invalid dataset population needs
to be clear to avoid append of new population

ga.calculate_fitness()

while(ga.population.max_fitness<93 and generation<1):

    print(f"Generation = {generation}")

    generation+=1

    parents = ga.selection()

    ga.cross_over(parents)

    ga.mutation()

    ga.calculate_fitness()

max_fitest = ga.population.max_fitest

max_fitness = round(ga.population.max_fitness,1)

self.tree_classifier.train_classifier(train_dataset,max_fitest)

self.trained = True

ga.clear_population()

self.progressBar.setProperty("value", 100)

self.showdialog('Model train',f'Model trained successfully ',1)

```

```

except:

    try:

        ga.clear_population()

    except:

        print("Err 00")

    finally:

        self.showdialog('Model train','Model trained unsuccessfully',2)

def static_testing(self):

    if self.isModelTrained():

        if (self.threadActive):

            self.showdialog('Warning','Please stop currently testing',3)

        else:

            test_dataset, train_dataset_type =
QFileDialog.getOpenFileName(MainWindow, "Select Testing Dataset","", "All Files
(*);;CSV Files (*.csv)")

            if test_dataset:

                try:

                    test_dataset = data.Dataset.refine_dataset(test_dataset, "Test
Dataset.txt")

                    t1 = threading.Thread(target=self.static_testing_thread, name =
'Static testing', args=(test_dataset,))

                    t1.start()

                    self.threadActive = True

                except:

                    self.showdialog('Error','Invalid Dataset',2)

```

else:

self.showdialog('Warning','Model not trained',3)

def static_testing_thread(self,dataset):

row = 0

self.reset_all_content()

with open(dataset,"r") as file:

for line in file.readlines():

try:

line = line.split(',')

result, result_type = self.tree_classifier.test_dataset(line)

self.insert_data(line,result,result_type,row)

row+=1

if self.pause:

while(self.pause):

pass

if self.isStop():

self.stop=False

break

time.sleep(0.05)

except:

print("Err")

```
self.threadActive = False
```

```
def realtime_testing(self):  
    if self.isModelTrained():  
        if (self.threadActive):  
            self.showdialog('Warning','Please stop currently testing',3)  
        else:  
            t2 = threading.Thread(target=self.realtime_testing_thread, name =  
'Realtime testing')  
            t2.start()  
            self.threadActive = True  
        else:  
            self.showdialog('Warning','Model not trained',3)  
def realtime_testing_thread(self):  
    self.reset_all_content()  
    self.packet.initiating_packets()  
    t1 = time.time()  
    attr_list = list()  
    capture = pyshark.LiveCapture(interface='Wi-Fi')  
    row = 0  
    try:  
        for p in capture.sniff_continuously():  
            try:
```

```

if "<UDP Layer>" in str(p.layers) and "<IP Layer>" in str(p.layers):

    attr_list = self.packet.udp_packet_attributes(p)

    result, result_type = self.tree_classifier.test_dataset(attr_list)

    self.insert_data(attr_list,result,result_type,row)

    print(attr_list)

    row+=1

elif "<TCP Layer>" in str(p.layers) and "<IP Layer>" in str(p.layers):

    attr_list = self.packet.tcp_packet_attributes(p)

    result, result_type = self.tree_classifier.test_dataset(attr_list)

    self.insert_data(attr_list,result,result_type,row)

    print(attr_list)

    row+=1

if (time.time()-t1) > 5 and not self.isStop: # 5Seconds

    print("Updateing List")

    self.packet.initiating_packets()

    t1 = time.time()

if self.pause:

    while(self.pause):

        pass

if self.isStop():

    self.stop=False

    break

except :

```

```

        print("Err")

    except :

        print("Error in looooooop")


def pause_resume(self):

    if self.pause:

        self.pause = False

        self.btn_start.setText("Pause")

    else:

        self.pause = True

        self.btn_start.setText("Resume")


def save_log_file(self):

    log = self.tree_classifier.get_log()

    url = QFileDialog.getSaveFileName(None, 'Save Log', 'untitled', "Text file
(*.txt);;All Files (*)")

    if url[0]:

        try:

            name = url[1]

            url = url[0]

            with open(url, 'w') as file:

                file.write(log)

            self.showdialog('Saved',f'File saved as {url}',1)

```

```

except:

    self.showdialog('Error','File not saved',2)

def stop_capturing_testing(self):

    if self.pause:

        self.pause = False

        self.btn_start.setText('Pause')

    if not self.stop:

        self.stop = True

    if self.threadActive:

        self.threadActive = False

def reset_all_content(self):

    if self.pause:

        self.pause = False

        self.btn_start.setText('Pause')

    self.stop=False

    self.tree_classifier.reset_class_count()

    self.panel_capturing.clearContents()

    self.panel_capturing.setRowCount(0)

    self.panel_result.clearContents()

    self.panel_result.setRowCount(0)

    self.panel_testing.clear()

```

```

def insert_data(self,line,result,result_type,row):

    self.panel_capturing.insertRow(row)

    for column, item in enumerate(line[0:4:1]):

self.panel_capturing.setItem(row,column,QtWidgets.QTableWidgetItem(str(item)))

        self.panel_capturing.scrollToBottom()

    self.panel_testing.clear()

    self.panel_testing.addItem(str(line[0:4:1]))

    if not result==0:

        result_row = self.panel_result.rowCount()

        self.panel_result.insertRow(result_row)

        x = [row+1, line[1], line[2], result_type]

        for column, item in enumerate(x):

self.panel_result.setItem(result_row,column,QtWidgets.QTableWidgetItem(str(item
)))

            self.panel_result.scrollToBottom()


def clickexit(self):

    buttonReply = QMessageBox.question(MainWindow, 'Exit', "Are ou sure to
exit?", QMessageBox.Yes | QMessageBox.No, QMessageBox.No)

    if buttonReply == QMessageBox.Yes:

        if self.threadActive:

            self.pause = False

            self.stop = True

```

```

        qApp.quit()

    else:

        print('No clicked.')

def isStop(self):

    return self.stop

def showdialog(self,title,text, icon_type):

    msg = QMessageBox()

    if icon_type==1:

        msg.setIcon(QMessageBox.Information)

    elif icon_type==2:

        msg.setIcon(QMessageBox.Critical)

    elif icon_type==3:

        msg.setIcon(QMessageBox.Warning)

    msg.setText(text)

    msg.setWindowTitle(title)

    msg.setStandardButtons(QMessageBox.Ok)

    msg.buttonClicked.connect(self.msgbtn)

    retval = msg.exec_()

def msgbtn(self):

self.progressBar.setProperty("value", 0)

def isModelTrained(self):

    return self.trained

def setupUi(self, MainWindow):

```

```

MainWindow.setObjectName("MainWindow")

path = os.path.dirname(os.path.abspath(__file__))

MainWindow.setWindowIcon(QtGui.QIcon(os.path.join(path, 'icon.png')))

MainWindow.resize(908, 844)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())

MainWindow.setSizePolicy(sizePolicy)

MainWindow.setIconSize(QtCore.QSize(30, 30))

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)

self.gridLayout.setObjectName("gridLayout")

spacerItem = QtWidgets.QSpacerItem(10, 10,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)

self.gridLayout.addItem(spacerItem, 1, 0, 1, 1)

spacerItem1 = QtWidgets.QSpacerItem(20, 20,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Maximum)

self.gridLayout.addItem(spacerItem1, 4, 1, 1, 1)

spacerItem2 = QtWidgets.QSpacerItem(20, 10,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)

self.gridLayout.addItem(spacerItem2, 6, 1, 1, 1)

```

```

self.horizontalLayout_2 = QtWidgets.QHBoxLayout()

self.horizontalLayout_2.setObjectName("horizontalLayout_2")

spacerItem3 = QtWidgets.QSpacerItem(15, 10,
QtWidgets.QSizePolicy.Ignored, QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout_2.addItem(spacerItem3)

self.btn_start = QtWidgets.QPushButton(self.centralwidget)

self.btn_start.setObjectName("btn_start")

self.btn_start.setText('Pause')

self.btn_start.clicked.connect(self.pause_resume)

self.horizontalLayout_2.addWidget(self.btn_start)

# #####

self.btn_pause = QtWidgets.QPushButton(self.centralwidget)

self.btn_pause.setText("Stop Capturing/Testing")

self.btn_pause.setObjectName("btn_pause")

self.btn_pause.clicked.connect(self.stop_capturing_testing)

self.horizontalLayout_2.addWidget(self.btn_pause)

self.gridLayout.addLayout(self.horizontalLayout_2, 8, 1, 1, 1)

self.horizontalLayout = QtWidgets.QHBoxLayout()

self.horizontalLayout.setObjectName("horizontalLayout")

# #####

self.btn_modeltrain = QtWidgets.QPushButton(self.centralwidget)

self.btn_modeltrain.setText("Train Model")

self.btn_modeltrain.setObjectName("btn_modeltrain")

```

```

self.btn_modeltrain.clicked.connect(self.train_model)

self.horizontalLayout.addWidget(self.btn_modeltrain)

# #####

self.btn_statictesting = QtWidgets.QPushButton(self.centralwidget)

self.btn_statictesting.setText("Static Testing")

self.btn_statictesting.setObjectName("btn_statictesting")

self.btn_statictesting.clicked.connect(self.static_testing)

self.horizontalLayout.addWidget(self.btn_statictesting)

# #####

self.btn_realtimetesting = QtWidgets.QPushButton(self.centralwidget)

self.btn_realtimetesting.setText("")

self.btn_realtimetesting.setObjectName("")

self.btn_realtimetesting.clicked.connect(self.realtime_testing)

self.horizontalLayout.addWidget(self.btn_realtimetesting)

# #####

self.btn_savelog = QtWidgets.QPushButton(self.centralwidget)

self.btn_savelog.setText("Save Log")

icon5 = QtGui.QIcon()

self.btn_savelog.setObjectName("btn_savelog")

self.btn_savelog.clicked.connect(self.save_log_file)

self.horizontalLayout.addWidget(self.btn_savelog)

```

```

#####

self.btn_graph = QtWidgets.QPushButton(self.centralwidget)

self.btn_graph.setText("Plot Graph")

self.btn_graph.setObjectName("btn_graph")

self.btn_graph.clicked.connect(self.plot_graph)

self.horizontalLayout.addWidget(self.btn_graph)

#####

self.btn_exit = QtWidgets.QPushButton(self.centralwidget)

self.btn_exit.setText("Exit")

self.btn_exit.setObjectName("btn_exit")

self.btn_exit.clicked.connect(self.clickexit)

self.horizontalLayout.addWidget(self.btn_exit)

#####

self.gridLayout.addLayout(self.horizontalLayout, 3, 1, 1, 2)

spacerItem4 = QtWidgets.QSpacerItem(20, 10,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)

self.gridLayout.addItem(spacerItem4, 8, 1, 1, 1)

spacerItem5 = QtWidgets.QSpacerItem(20, 10,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)

self.gridLayout.addItem(spacerItem5, 0, 1, 1, 1)

self.panel_capturing = QtWidgets.QTableWidget(self.centralwidget)

```

```

        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)

        sizePolicy.setHorizontalStretch(10)

        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.panel_capturing.sizePolicy().hasHeightForWidth
())

        self.panel_capturing.setSizePolicy(sizePolicy)

        self.panel_capturing.setRowCount(0)

        self.panel_capturing.setColumnCount(4)

        self.panel_capturing.setObjectName("panel_capturing")

        item = QtWidgets.QTableWidgetItem()

        self.panel_capturing.setHorizontalHeaderItem(0, item)

        item = QtWidgets.QTableWidgetItem()

        self.panel_capturing.setHorizontalHeaderItem(1, item)

        item = QtWidgets.QTableWidgetItem()

        self.panel_capturing.setHorizontalHeaderItem(2, item)

        item = QtWidgets.QTableWidgetItem()

        self.panel_capturing.setHorizontalHeaderItem(3, item)

        self.gridLayout.addWidget(self.panel_capturing, 4, 1, 4, 1)

        self.label = QtWidgets.QLabel(self.centralwidget)

        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)

        sizePolicy.setHorizontalStretch(0)

        sizePolicy.setVerticalStretch(0)

```

```

sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())

self.label.setSizePolicy(sizePolicy)

self.label.setLayoutDirection(QtCore.Qt.LeftToRight)

self.label.setAutoFillBackground(False)

self.label.setText("")

path = os.path.dirname(os.path.abspath(__file__))

path = path + r'\icons'

self.label.setPixmap(QtGui.QPixmap(os.path.join(path,'logo.jpg')))

self.label.setScaledContents(True)

self.label.setAlignment(QtCore.Qt.AlignCenter)

self.label.setObjectName("label")

self.gridLayout.addWidget(self.label, 1, 1, 1, 1)

spacerItem6 = QtWidgets.QSpacerItem(10, 20,
QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)

self.gridLayout.addItem(spacerItem6, 2, 1, 1, 1)

self.panel_testing = QtWidgets.QListWidget(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.panel_testing.sizePolicy().hasHeightForWidth())

self.panel_testing.setSizePolicy(sizePolicy)

self.panel_testing.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)

self.panel_testing.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAsNeeded)

```

```

self.panel_testing.setObjectName("panel_testing")

self.gridLayout.addWidget(self.panel_testing, 9, 1, 1, 1)

self.progressBar = QtWidgets.QProgressBar(self.centralwidget)

self.progressBar.setProperty("value", 0)

self.progressBar.setObjectName("progressBar")

self.gridLayout.addWidget(self.progressBar, 10, 1, 1, 2)

# ----- #

self.panel_result = QtWidgets.QTableWidget(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)

sizePolicy.setHorizontalStretch(10)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.panel_result.sizePolicy().hasHeightForWidth())

self.panel_result.setSizePolicy(sizePolicy)

self.panel_result.setRowCount(0)

self.panel_result.setColumnCount(4)

self.panel_result.setObjectName("panel_result")

item = QtWidgets.QTableWidgetItem()

self.panel_result.setHorizontalHeaderItem(0, item)

item = QtWidgets.QTableWidgetItem()

self.panel_result.setHorizontalHeaderItem(1, item)

item = QtWidgets.QTableWidgetItem()

self.panel_result.setHorizontalHeaderItem(2, item)

```

```

item = QtWidgets.QTableWidgetItem()

self.panel_result.setHorizontalHeaderItem(3, item)

self.gridLayout.addWidget(self.panel_result, 4,2,6,1)

# ----- #

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 908, 26))

self.menubar.setObjectName("menubar")

self.menuFile = QtWidgets.QMenu(self.menubar)

self.menuFile.setObjectName("menuFile")

self.menuAbout = QtWidgets.QMenu(self.menubar)

self.menuAbout.setObjectName("menuAbout")

MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

MainWindow.setStatusBar(self.statusbar)

self.actionNew = QtWidgets.QAction(MainWindow)

self.actionNew.setObjectName("actionNew")

self.actionOpen = QtWidgets.QAction(MainWindow)

self.actionOpen.setObjectName("actionOpen")

self.actionExit = QtWidgets.QAction(MainWindow)

self.actionExit.setObjectName("actionExit")

self.actionHelp = QtWidgets.QAction(MainWindow)

```

```

self.actionHelp.setObjectName("actionHelp")

self.menuFile.addAction(self.actionNew)

self.menuFile.addAction(self.actionOpen)

self.menuFile.addSeparator()

self.menuFile.addAction(self.actionExit)

self.actionExit.triggered.connect(qApp.quit)

self.menuAbout.addAction(self.actionHelp)

self.menubar.addAction(self.menuFile.menuAction())

self.menubar.addAction(self.menuAbout.menuAction())

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

```

```

def retranslateUi(self, MainWindow):

    _translate = QtCore.QCoreApplication.translate

    MainWindow.setWindowTitle(_translate("MainWindow", "Intrusion
Detection"))

    self.btn_start.setStatusTip(_translate("MainWindow", "Pause/Resume"))

    self.btn_pause.setStatusTip(_translate("MainWindow", "Stop"))

    self.btn_modeltrain.setStatusTip(_translate("MainWindow", "Train Model"))

    self.btn_statictesting.setToolTip(_translate("MainWindow", "Stactic Testing"))

    self.btn_statictesting.setStatusTip(_translate("MainWindow", "Static Testing"))

    self.btn_savelog.setToolTip(_translate("MainWindow", "Real Time
Capturing"))

```

```

self.btn_savelog.setStatusTip(_translate("MainWindow", "Real Time
Capturing"))

self.btn_graph.setStatusTip(_translate("MainWindow", "Graph"))

self.btn_exit.setStatusTip(_translate("MainWindow", "Exit"))

item = self.panel_capturing.horizontalHeaderItem(0)
item.setText(_translate("MainWindow", "Duration"))

item = self.panel_capturing.horizontalHeaderItem(1)
item.setText(_translate("MainWindow", "Protocol"))

item = self.panel_capturing.horizontalHeaderItem(2)
item.setText(_translate("MainWindow", "Service"))

item = self.panel_capturing.horizontalHeaderItem(3)
item.setText(_translate("MainWindow", "Src_Bytes"))

# ----- #

item = self.panel_result.horizontalHeaderItem(0)
item.setText(_translate("MainWindow", "Packet #"))

item = self.panel_result.horizontalHeaderItem(1)
item.setText(_translate("MainWindow", "Protocol"))

item = self.panel_result.horizontalHeaderItem(2)
item.setText(_translate("MainWindow", "Service"))

item = self.panel_result.horizontalHeaderItem(3)
item.setText(_translate("MainWindow", "Class"))

# ----- #

self.menuFile.setTitle(_translate("MainWindow", "File"))

self.menuAbout.setTitle(_translate("MainWindow", "About"))

```

```

        self.actionNew.setText(_translate("MainWindow", "New"))

        self.actionOpen.setText(_translate("MainWindow", "Open"))

        self.actionExit.setText(_translate("MainWindow", "Exit"))

        self.actionHelp.setText(_translate("MainWindow", "Help"))


if __name__ == "__main__":

    import sys

    app = QtWidgets.QApplication(sys.argv)

    MainWindow = QtWidgets.QMainWindow()

    ui = Ui_MainWindow()

    ui.setupUi(MainWindow)

    MainWindow.show()

    sys.exit(app.exec_())


#CLASS

import Individual

import random


class Population():

    individual = list()

    max_fitness = 0

    max_fittest = list()

    childrens = list()

    def __init__(self,train_dataset,test_dataset,population_size=5,
gene_length=18):

```

```

self.population_size = population_size

self.gene_length=gene_length

self.train_dataset = train_dataset

self.test_dataset = test_dataset

self.no_of_child = self.population_size if (self.population_size%2==0) else
self.population_size-1

def initialize_population(self):

    for x in range(self.population_size):

self.individual.append(Individual.Individual(self.train_dataset,self.test_dataset))

def calculate_fitness(self):

    for x in range(len(self.individual)):

        self.individual[x].calculate_fitness()

    self.individual = sorted(self.individual, key=self.__get_fitness, reverse=True)
#descending sorting

    self.individual = self.individual[0:self.population_size:1] #cut the extra
individual with less fitness

    self.max_fittest = self.individual[0].chromosome

    self.max_fitness = self.individual[0].fitness

def __get_fitness(self,ind):

    return ind.fitness

def cross_over(self,parents):

    cut_point = 9 # Or can be generated randomly between range (0 -
sizeOfGene)

```

```

c1 = 0

c2 = 1

for x in range(self.no_of_child):

self.childrens.append(Individual.Individual(self.train_dataset,self.test_dataset))

    for parent in parents:

        p1 = self.individual[parent[0]]

        p2 = self.individual[parent[1]]

        self.childrens[c1].chromosome = p2.chromosome[cut_point:] +
p1.chromosome[cut_point:]

        self.childrens[c2].chromosome = p1.chromosome[cut_point:] +
p2.chromosome[cut_point:]

        c1 = c1+2

        c2 = c2+2

def mutation(self,mutation_rate):

    mutation_rate = float(mutation_rate/100)

    for child in self.childrens:

        gene = child.chromosome

        gene = [self.__flip_bit(gene[x]) if (random.uniform(0,1)<mutation_rate) else
gene[x] for x in range(len(gene))]

        child.chromosome = gene

    self.individual.extend(self.childrens)

    self.childrens.clear() # Clear childrens

def __flip_bit(self, bit):

    """process"""

```

```

        return 0 if bit==1 else 1

def clear_population(self):
    self.individual.clear()

#CLASSIFIER

from sklearn import tree

import pandas

import string

class DecisionTree():

    def __init__(self):
        self.tree_classifier = tree.DecisionTreeClassifier()

        self.__dos = 0

        self.__prob = 0

        self.__normal = 0

    def train_classifier(self,dataset, attributes):
        header = list(string.ascii_lowercase[0:19])

        kdd_train = pandas.read_csv(dataset, names=header)

        self.selected_attributes = [x for x,y in enumerate(attributes) if y==1]

        self.selected_index= [header[x] for x, y in enumerate(attributes) if y==1]

        var_train, res_train = kdd_train[self.selected_index], kdd_train[header[18]]

        self.tree_classifier.fit(var_train, res_train)

    def test_dataset(self,packet):

```

```

packet_list = list()

packet_list.append([packet[x] for x in self.selected_attributes])

result = self.tree_classifier.predict(packet_list)

result = int(result[0])

packet_list.clear()

return self.__classification(result)

def __classification(self, status):

    if status == 0:

        self.__normal+=1

        result = 0

    elif status in range(1,6):

        self.__dos+=1

        result = 1

    else:

        self.__prob+=1

        result = 2

    classification = {

        '0' : 'Normal',

        '1' : 'Dos/Neptune',

        '2' : 'Dos/Back',

        '3' : 'Dos/Apache2',

        '4' : 'Dos/Phf',

        '5' : 'Dos/Saint',

```

```

        '6' : 'Prob/IpSweep',

        '7' : 'Prob/PortSweep',

        '8' : 'Prob/Satan',

        '9' : 'Prob/Nmap'

    }

    result_class = classification[str(status)]

    return (result, result_class)

def reset_class_count(self):

    """Reset the no.of dos,prob and normal count to zero"""

    self.__dos = 0

    self.__prob = 0

    self.__normal = 0

def get_class_count(self):

    return (self.__normal, self.__dos,self.__prob)

def get_log(self):

    total = self.__dos+self.__prob+self.__normal

    log = f'Total = {total}\nNormal = {self.__normal}\nDoS = {self.__dos}\nProb = {self.__prob}'

    return log

    @staticmethod

def get_fitness(var_train, res_train, var_test, res_test):

    #Consume ram<100MB, processor<15%

    clf = tree.DecisionTreeClassifier()

    clf.fit(var_train, res_train)

```

```
return round(clf.score(var_test, res_test),3)
```

```
#DECISION TREE
```

```
import numpy as np
```

```
import pandas as pd
```

```
from sklearn.kernel_approximation import RBFSampler
```

```
from sklearn.linear_model import SGDClassifier
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn import svm
```

```
from sklearn.metrics import classification_report
```

```
from sklearn import metrics
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import (precision_score, recall_score, f1_score,  
accuracy_score, mean_squared_error, mean_absolute_error)
```

```
from sklearn.ensemble import AdaBoostClassifier
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn import preprocessing
```

```
from sklearn.preprocessing import Normalizer
```

```
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.svm import SVC
```

```
from sklearn.metrics import confusion_matrix
```

```

from sklearn.metrics import (precision_score, recall_score, f1_score,
accuracy_score, mean_squared_error, mean_absolute_error, roc_curve,
classification_report, auc)

import warnings

warnings.filterwarnings("ignore")

traindata = pd.read_csv('Dataset/tr.csv', header=None)

testdata = pd.read_csv('Dataset/ts.csv', header=None)

X = traindata.iloc[:,1:42]

Y = traindata.iloc[:,0]

C = testdata.iloc[:,0]

T = testdata.iloc[:,1:42]

scaler = Normalizer().fit(X)

trainX = scaler.transform(X)

scaler = Normalizer().fit(T)

testT = scaler.transform(T)

traindata = np.array(trainX)

trainlabel = np.array(Y)

testdata = np.array(testT)

testlabel = np.array(C)

print("*****")

model = DecisionTreeClassifier()

model.fit(traindata, trainlabel)

print(model)

# make predictions

```

```

expected = testlabel

predicted = model.predict(testdata)

np.savetxt('predictedDT.txt', predicted, fmt='%01d')

# summarize the fit of the model

accuracy = accuracy_score(expected, predicted)

recall = recall_score(expected, predicted, average="binary")

precision = precision_score(expected, predicted , average="binary")

f1 = f1_score(expected, predicted , average="binary")

cm = metrics.confusion_matrix(expected, predicted)

print(cm)

tpr = float(cm[0][0])/np.sum(cm[0])

fpr = float(cm[1][1])/np.sum(cm[1])

print("%.3f" %tpr)

print("%.3f" %fpr)

print("Accuracy")

print("%.3f" %accuracy)

print("precision")

print("%.3f" %precision)

print("recall")

print("%.3f" %recall)

print("f-score")

print("%.3f" %f1)

print("*****")

```

RESEARCH PAPER

Detection Method for Software Defined Network Intrusion

Nagarjuna G. K

*Department of Computer Science
and Engineering,
Sathyabama Institute of Science and
Technology
Tamil Nadu 600119, India
murthyng2002@gmail.com*

Nandigam Kumar Satya Ram

*Department of Computer Science
and Engineering,
Sathyabama Institute of Science and
Technology
Tamil Nadu 600119, India
nkumarsatyaram@gmail.com*

Dr. S. L. Jany Shabu

*Department of Computer Science
and Engineering,
Sathyabama Institute of Science and
Technology
Tamil Nadu 600119, India
janyshabu.cse@sathyabama.ac.in*

Abstract— There is a great deal of interest in intrusion detection systems (IDSs) as a key component of system defense at the moment. To secure a network IDS analyzing network traffic from a location on the network or computer system. It is very difficult and time-consuming to distinguish between network traffic that is intrusive and normal. An analyst must examine that much and such a wide range of data. It is necessary to determine the sequence in which intrusions have occurred on the network connection. Current network traffic activity must be reflected by a way to detect network intrusions. Using genetic algorithm and machine learning approaches, a novel approach was developed to uncover intrusion characteristics for intrusion detection systems. The rules are generated using a genetic algorithm for decision trees. Rules can be applied to determine intrusion characteristics, and then implemented into a genetic algorithm for protection. In this way, in addition to identifying the presence of an intrusion, one may stop the incursion by rejecting it.

Keywords - Intrusion detection systems (IDSs), Genetic algorithms, Machine learning, Traffic flow, Neural Networks.

I. INTRODUCTION

The Internet has been growing rapidly in recent years, making computer systems more and more susceptible to security breaches. Even with major technological advancements in information assurance, protecting computer systems is still quite challenging. As a result, when the software systems are actually running, unwanted invasions take place. While we indulge, the simplicity that modern technology has given us has made computer systems more vulnerable to both internal and external security problems. Despite many security measures, it is practically impossible to have a system that is totally secure.

Since illegal network access, unusual network behaviour, and hostile attacks on computer systems are all examples of network intrusions, intrusion detection is becoming a crucial technique for monitoring network traffic. An intrusion occurs when a person tries to access a system of information or engages in activity that is illegal. The two types of intruders are exterior and inside. Assuming the first scenario, an attacker who is not authorized to access the system will employ a number of different methods to gain entry and initiate an assault. The second category includes people who

have access to rights yet want to engage in illegal activity. A technique for locating intrusions and reliably reporting them to the relevant authority is an intrusion detection system.

To discover the purpose of discovering network intrusions, many techniques based on soft computing have been proposed. A group of techniques known as "soft computing" takes use of the ability to approximate, parse truth, and tolerance for uncertainty in order to achieve affordability and robustness. Soft computing approaches are frequently used with rule-based expert systems that learn subject-matter expertise and represent that information as a collection of if-then rules when utilized for intrusion detection.

Intruders can be spotted in two primary ways: through the detection of misuse and through the detection of anomalies. The abuse detection system can locate every known sort of assault by studying the established intrusion patterns in the system audit flow (intrusions). A high false positive rate is a disadvantage of anomaly detection, but it has the advantage of being able to spot new or unexpected threats. Machine learning methods based on network monitoring have applications in many different industries.

With the use of query-based crawling, the suggested method collects sentences about any traffic-related occurrences, such as traffic jams, road closures, etc., from social media (Facebook and Twitter). A number of pre-processing techniques, including segmentation, reorganization, steaming, and POS tagging, are then used to convert the obtained data into a structured format. In order to automatically classify the data as "traffic" or "non-traffic", the Latent Dirichlet Allocation (LDA) technique is then utilized. We look at data kinds with positive, negative, and neutral traffic labels. This stage results in a sentence that bears labels indicating whether it is or is not a traffic sentence as well as the polarity of that traffic sentence (positive, negative, or neutral).

After learning, the neural networks use the soft-max layer to perform multi-class classification, which classifies the text into different categories based on location, traffic event, and polarity kinds. In terms of accuracy, F-score, and other metrics, the suggested method contrasts a wide range of traditional machine learning and cutting-edge deep learning

techniques. IDS is a software application built using python that has a security feature that can detect any hacker entry into such as unauthorized access, misuse, and other intrusions into the computer system or network. All activity on the target computer or network is first gathered by the system. Using the patterns, it has learned, it then produces new patterns, stores them in a database as examples, examines and keeps track of all incoming activity, and finally generates alarms to alert the administrator of potential risks. Based on the requirements from the organization, scalability of the IDS and using signature-based methodology we will be doing the selection.

II. LITERATURE SURVEY

Li Yang, Abdallah Shami [1] developed an open-source code repository for building IDSs utilizing traditional and advanced ML algorithms on public network traffic datasets. The python-based code repository with optimized ML models for identifying various types of cyber-attacks. The model outperformance enables easy implementation and reproduction of IDS on intrusion detection datasets, contributing to cybersecurity solution.

Lianming Zhang, KuiLiu, Xiaowei Xie [2] developed a model to improve data quality and detection accuracy in NIDS using feature selection and deep learning techniques. The model utilizes standard deviation and association rule mining for feature selection and implements a three-layer neural network architecture with minimized neurons. This proposed system achieved an enhanced detection performance with a reduced set of traffic features.

Pampapathi B M, Nageswara Guptha M [3] developed an effective IDS for distributed sensor network with a focus on handling heterogeneous data sources and discovering zero-day attacks. The model utilizes a five-phase approaches like initialization, cluster formation, connectivity, attack detection and data broke. The model achieved a high accuracy of 96.12% outperforming existing deep learning neural network and artificial neural network approaches.

Ramkumar M. P, P.V.Bhaskar Reddy, J.T. Thirukrishna, Ch . Vidyadhari [4] developed an effective IDS using hybrid feature fusion and optimization enabled deep learning on spark architecture. The model is RV coefficient-based hybrid feature fusion incorporating wrapper class-wise information gain (CIG), and Canberra distance. Data augmentation through oversampling. Utilization of DRN classifier. Training with ExpSLO algorithm (a hybrid of EWMA and SLnO). The model achieved high precision (0.8800), recall (0.8822), and F-measure (0.8822) without attacks using dataset-2.

R. Vinayakumar, Mamoun Alazab, K. P. Soman [5] implemented a deep neural network a type of deep learning model for developing a flexible and IDS capable of detecting unforeseen and unpredictable cyberattacks. The DNN model is claimed to learn abstract and high-dimensional features representation of IDS data, contributing to the effectiveness of cyberattack detection.

Rashid Baimukashev, Artykbayev Kamalkhan, Adam Kazybek, Mels Begenov [6] developed a model that enhances network security in wireless networks by designing and evaluating an intrusion detection system. The model

utilizes deep learning approaches including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM). - Employs traditional machine learning algorithms such as Support Vector Machines (SVM), Random Forest, and XGBoost. - Evaluation conducted using the NSL-KDD dataset.

Saleh A. Albelwi [7] developed a model that address two critical challenges in intrusion detection system. 1. The evolving nature of malicious activity, where data flows and packets can contain multiple types of attacks simultaneously. 2. the limitation of the publicly available datasets that evaluate only one type of attack. The paper proposes a multi-task learning model based on deep neural networks to simultaneously detect several types of attacks. The proposed multi-task learning-based IDS using DNNs effectively addresses the challenge of detecting multiple types of attacks simultaneously. The experimental results show that the system maximizes performance in this context.

Xue Ying Li, Rui Tang, Wei Song [8] have implemented the application of deep learning techniques to network intrusion detection and proposes an improved IDS method based on convolution neural network (CNN) with an inception module for optional intrusion feature extraction. The model introduces an inception module into the traditional convolution neural network architecture to optimize intrusion feature extraction. Employs parallel convolution structures with different filter sizes for improved feature identification.

III. EXISTING WORK AND THEIR LIMITATIONS

The traditional methodologies are having some limitations and these limitations are as follows:

Anomaly detection accuracy: The accuracy of anomaly detection can be improved anomalies can be diverse and context-dependent, making it challenging to distinguish between legitimate unusual behaviour and actual attacks.

Reducing false positives: minimizing false positives is crucial in order to avoid alert fatigue and ensure that security teams focus on genuine threats. Fine-tuning algorithms and enhancing feature selection can contribute to reducing false alarms.

Multi-model attacks: detecting attacks that use a combination of different techniques remains a challenge. Developing approaches that can effectively identify and respond to multi-modal attacks is essential.

Adversarial attacks: Adversarial attacks involve attempts to deceive or manipulate the intrusion detection system. Research in robust machine learning techniques and adversarial training is necessary to mitigate the impact of such attacks.

IV. PROPOSED SYSTEM

The proposed system consists of genetic algorithm which is one of the best machine learning algorithms in the field of intrusion detection, which consist of its natural selection. The decision node represents testing a single attribute of the given instances whereas the leaf node presents the idea of whether the output classifier falls into either the normal or intrusion (any of the possible attacks) category during the classification phase. This model is a novel method to find intrusion characteristics for IDS using decision tree machine learning of data mining technique was proposed. The method used to generate rules is classification by the genetic algorithm of the decision tree. The behaviour of the attacker is obtained from

the historical data of the NSL-KDD dataset. The proposed systems provide normal, probe and denial of service (DOS) attacks at the end. Signature based methodology will be used to analyze activities on the network interface of a particular asset.

V. WORKING

A. Data-set collection

NSL-KDD Data Set

NSL-KDD is the updated version of KDD99 data set. This data set is useful in comparing Intrusion detection methods. The required data set was found in the Kaggle web application. The two data set files obtained are KDDTrain+.txt and KDDTest+.txt are the training and testing data set file which are in the text format and the difficulty levels of data are given in the csv format.

- Dos attacks aim to stop traffic flowing to and from their target system.
- Attacks that try to gather information from a network are known as probing or surveillance attacks. Here, impersonating a thief is intended in order to obtain critical data, such as banking or client personal information.

In this data set, there are four different sorts of feature types:

- Four categorized (Features: 3, 4, 6, 32)
- Six Binaries (Features: 3, 8, 10, 16, 17, 18)
- Twenty-three Delicate (Features: 3, 5, 9, 22–40, 42)
- Ten continuous (Functions: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

B. Preprocessing

- Each continuous attribute's value is scaled from 0 to 1, during data normalization.
- The dataset contains a lot of irrelevant data we remove those columns by a library known as pandas.
- Once the dataset only has the relevant data, we will create another data set file which contains only the required columns for the training and testing purpose.
- Scaling is carried out using mean and standard deviation utilizing the transform (data) approach.

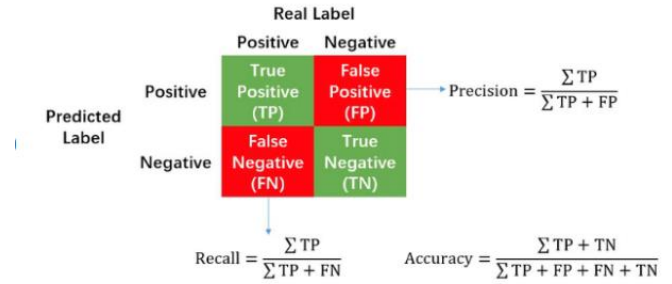
C. Model Training

- The attributes are tested on internal nodes, when using the decision tree approach. The test results are displayed as branches, and class labels are located on leaf nodes. Nodes are identified using attribute selection methods.
- These chosen attributes indicate the least amount of ambiguity or reducing the quantity of information in the partitions to reductionism needed for tuple categorization in the resulting partition.
- Decreasing the expected quantity of tests required for tuple categorization. This study's decision trees employ entropy class to help ID3 algorithms choose which characteristics to query on for each node.
- Since the dataset is a labeled one, classification of the intrusion is already done in the training phase.
- While training we will select a target variable and choosing it as a pivot entire training process will be done.

D. Model Testing

Evaluation Metrics

- The efficiency of various machine learning techniques is compared using a confusion matrix. In this matrix, the True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP) numbers are columns that can be added to produce other metrics (FP). Here are a few performance metrics for using the confusion matrix to assess models.



- Accuracy refers to how closely the model's estimated value matches the actual or true value, or how many out of all the samples are properly classified.
- Precision identifies the percentage of relevant instances that are truly positive among the instances that were chosen.
- True Positive Rate (TPR), also known as recall, is the percentage of real positives that are accurately detected.
- Based on the training done we can classify the intrusion and based on the accuracy results we can conclude if the classification is done properly or not.
- Since we have done feature extraction in the training phase, based on the results we can check if the attributes are properly predicted or not.

E. Prediction

- The goal was to choose the fewest features possible while maintaining a high rate of intrusion detection. As a result, real-time detection could be carried out.
- Enter regular data into the GUI's trained model to create a graph for intrusion detection that shows a bar graph and records a log to demonstrate the results.
- Enter attack information into the GUI's trained attack model, plot the graph for intrusion detection, which shows a bar graph of the dos and probing attacks, and save the log so that the result can be seen.

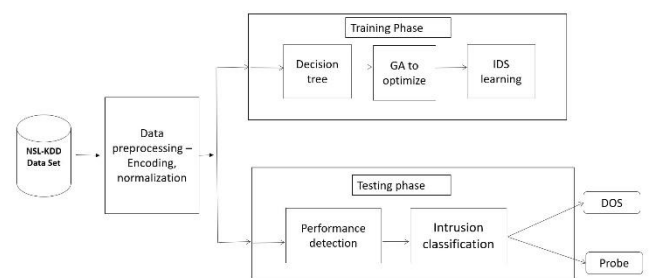


Fig1. Network Intrusion Detection System

Some IDS combine anomaly and abuse detection to create hybrid detection systems. Depending on where they hunt for intrusions, the IDS's can also be divided into two groups. The computer network intrusion detection systems is shown in Fig. 1, IDS is a network-based monitoring of network traffic, whereas a host-based IDS keeps an eye on actions related to a specific host. These systems have a flaw in that they can only detect assaults that are already present in the attack database. This model must therefore be updated frequently, but it does have the advantage of having a very low false positive rate. Without any prior awareness of them, anomaly detection systems are capable of spotting deviations from the norm as well as potential new or unexpected assaults. Although they have a greater risk of false alerts, they can identify unknown assaults and search for deviations much more quickly.

Genetic Algorithm (GA)

The most crucial piece of data that can be used for effective network attack detection should be retrieved in order to effectively execute intrusion detection by analyzing network data in real-time. The most important components of the data are extracted using Principal Component Analysis, often known as the Karhunen Loeve transform. By attempting to extract a small number of orthogonal linear combinations of the variables with the highest variance, PCA attempts to reduce the dimension of the data. Even if the research and industry that deal with intrusion detection have developed a powerful machine learning technique, there is still a need to increase the detection rate with minimal false alarms. The novel GA approach, when compared to prior approaches, results in detecting anomaly-based network intrusions, greater detection rates and fewer false alarms are required.

A genetic algorithm's fundamental steps are as follows:

- Initialization: Establish a starting population.
- Evaluation: After evaluating each population member, we determine that person's "fitness."
- Selection : We want to continually improve the general fitness of our population.

Using the KDD Cup '99 data-set, the effectiveness of this method is examined. The PCA technique is used to detect certain network characteristics because they are more likely to be implicated in network intrusions. Finding the features that are commonly used in the attack's machinery requires applying the PCA approach to the training data set. Out of the 41 features that were considered, three are determined to be the best ones to define each link in the KDD Cup '99 data set. In order to get an intrusion detection of a high rate, the fewest features possible had to be chosen. As a result, real-time detection could be carried out. The features chosen as well as their explanations are shown in Table 1.

Table 1: Selected network feature

Name of feature	Explication	No. of genes
Duration	Length (number of seconds) of the connection	1
Src_bytes	Number of data bytes from source to destination	1
dst_host_srv_	Percentage of connections that have "SYN" errors	1
seeror_rate		

Each trait or gene is symbolized by a single byte, so each chromosome is made up of three bytes. There isn't any ambiguity in the if-then constructions of an intruder detection

algorithm. Features from Table 1 are linked to a function to create the conditional part of a rule. In fact, every law serves to justify a violation of privacy. For instance, a rule could be as follows: A breach has occurred if (length = "1," src bytes = "0," and dst_ host srv seeror rate = "0."

High detection rates and low false-positive rates are key factors in determining a high fitness value. In contrast, a low fitness value is the outcome of a high rate of false-positives and a low detection rate. The following is how the algorithm for creating new rules work. The first phase entails initializing a population of starting genes, where each gene is assigned a random value. The evolutionary algorithm's parameters (crossover and mutation rates, population size, and pointing rules stop evolving) are then determined, and data for the network audit is loaded. The founding population then goes through several generations of evolution. The fitness function is used to calculate each rule's quality, or fitness value, for each generation. Once the rules with the highest fitness values have been chosen, the genetic operators (crossover and mutation), which are implemented with a specified probability, are Intrusion detection rules are produced as a result of the algorithm.

VI. RESULTS AND DISCUSSION

Differentiating between network traffic activities that constitute an intrusion and those that are regular requires a lot of time and effort. An analyst must look through all the huge and diverse data in order to determine when exactly the network link was compromised. A technique of network intruder identification is necessary for reflecting the actual network activity at present.

As a result of the development of public and private cloud computing, networks have today become a crucial component of public infrastructures. The conventional networking strategy has gotten very complicated. This complexity has made it more difficult to link data centers, integrate businesses, and develop new, innovative services within a single data center. It has also made it more difficult for the Internet to flourish in general.

The genetic algorithm, which makes use of natural selection, is widely used machine learning methods for intrusion detection. The leaf node indicates whether a classifier's output during the decision node denotes checking a single property of the offered examples, whereas the classification phase belongs to the normal or intrusion (any of the potential attacks) category. There was a recommendation that the decision tree machine learning data mining approach be used to identify attack traits for IDS. The genetic algorithm of a decision tree is a method for generating rules.

Duration	Protocol	Service	Src_Bytes	Packet #	Protocol	Service	Class
20 20	6	80	0	20 20	6	80	DoS/NegHunt
21 21	6	80	0	21 21	6	80	DoS/Back
22 22	6	80	54540	22 22	6	80	DoS/Back
23 23	6	80	0	23 23	6	80	DoS/Back
24 24	6	80	0	24 24	6	80	DoS/Back
25 25	6	80	54540	25 25	6	80	DoS/NegHunt
26 26	6	80	54540	26 26	6	80	DoS/NegHunt
27 27	6	80	54540	27 27	6	80	DoS/NegHunt
28 28	6	80	0	28 28	6	80	DoS/NegHunt
29 29	6	80	0	29 29	6	80	DoS/NegHunt
30 30	6	80	0	30 30	6	80	DoS/Back
31 31	6	80	0	31 31	6	80	DoS/NegHunt
32 32	6	80	0	32 32	6	80	DoS/Back
33 33	6	80	0	33 33	6	80	DoS/Back
34 34	6	80	0	34 34	6	80	DoS/Back
35 35	6	80	0	35 35	6	80	DoS/Back
36 36	6	80	0	36 36	6	80	DoS/Back
37 37	6	80	0	37 37	6	80	DoS/NegHunt

Fig2. Attack data detection in GUI

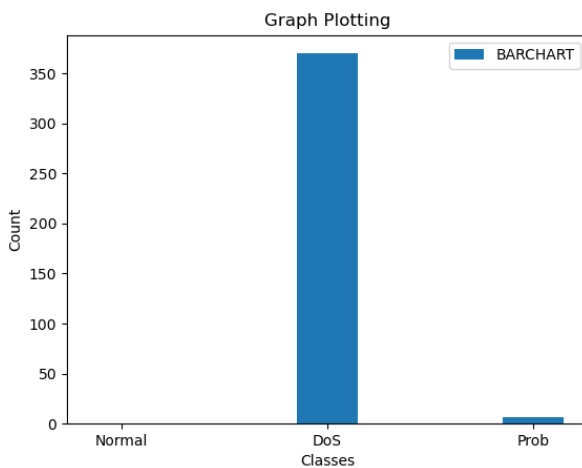


Fig3. Graphical plot for Attack data

Manual or automatic intrusion detection is available. IDS must be able to handle a substantial and continuously growing volume of alerts and events. It is more important to use decision trees. It is becoming more and more important to have automated systems for monitoring and responding to intrusions. Hence it can be inferred that usage of IDS, genetic algorithm is a great approach to get better insights of the security vulnerabilities in the network, and decision tree helps us in the classification process and the synergy of these things would be a great approach in network security.

VII. CONCLUSION

In this study, intrusion detection is carried out using a genetic algorithm and a decision tree technique. The suggested approach's software implementation is shown. Using a genetic algorithm, classification criteria for intrusion detection were created, and Key features of interconnection networks were identified using principal component analysis. It is simple to develop and maintain because the GA-approach accurately and clearly displays the required rules and fitness functions. Additionally, given the right attack taxonomy and training data-set, the system's adaptability allows it to serve multiple purposes. Because the point of incursion detection is to catch attacks as they happen and stop them before they can do any harm, the method doesn't care how the attacks are classified. The benefits of employing this technique for intrusion detection instead of a complimentary technique, which is frequently employed with other soft-computing techniques, are a high rate of assault detection and a low percentage of false positives. The system simply makes use of three network connection features while retaining high detection rates, allowing it to carry out the intrusion detection procedure more quickly and be used with high-speed networks.

REFERENCES

[1] L. Yang, A. Moubayed, I. Hamieh and A. Shami, "Tree-Based Intelligent Intrusion Detection System in Internet of Vehicles," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013892.

[2] Lianming Zhang, Kui Liu, Xiaowei Xie, Wenji Bai, Baolin Wu, Pingping Dong, A data-driven network intrusion detection system using feature selection and deep learning, *Journal of Information Security and Applications*, Volume

78, 2023, 103606, ISSN2214-2126, <https://doi.org/10.1016/j.jisa.2023.103606>.

[3] Pampapathi B M, Nageswara Guptha M, M S Hema, Towards an effective deep learning-based intrusion detection system in the internet of things, *Telematics and Informatics Reports*, Volume 7, 2022, 100009, ISSN 2772-5030, <https://doi.org/10.1016/j.teler.2022.100009>.

[4] Ramkumar M. P., P.V. Bhaskar Reddy, J.T. Thirukrishna, Ch. Vidyadhari, Intrusion detection in big data using hybrid feature fusion and optimization enabled deep learning based on spark architecture, *Computers & Security*, Volume 116, 2022, 102668, ISSN 0167-4048, <https://doi.org/10.1016/j.cose.2022.102668>.

[5]. R. Vinayakumar, Mamoun Alazab, K. P. Soman. "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, pp. 46717-46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

[6] R. Baimukashev, A. Kamalkhan, A. Kazybek and M. Begenov, "Intrusion Detection System for Wireless Networks," 2021 16th International Conference on Electronics Computer and Computation (ICECCO), Kaskelen, Kazakhstan, 2021, pp. 1-5, doi: 10.1109/ICECCO53203.2021.9663787.

[7] S. A. Albelwi, "An Intrusion Detection System for Identifying Simultaneous Attacks using Multi-Task Learning and Deep Learning," 2022 2nd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 2022, pp. 349-353, doi: 10.1109/ICCIT52419.2022.9711630.

[8] X. Y. Li, R. Tang and W. Song, "Intrusion Detection System Using Improved Convolution Neural Network," 2022 11th International Conference of Information and Communication Technology (ICTech), Wuhan, China, 2022, pp. 97-100, doi: 10.1109/ICTech55460.2022.00028.

[9] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A new feature selection model based on ID3 and bee's algorithm for intrusion detection system," p. 8.

[10] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21954-21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[11] Choudhary S, Kesswani N. 2020. Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT. *Procedia Computer Science* 167(2019):1561-1573.

[12] D. Agrawal, C. Agrawal, and H. Yadav, "A Machine Learning Based Intrusion Detection Framework Using KDDCUP 99 Dataset," vol. 4, no. 6, p. 11.

[13] D. M. Abdulqader, A. M. Abdulazeez, and D. Q. Zeebaree, "Machine Learning Supervised Algorithms of Gene Selection: A Review," vol. 62, no. 03, p. 13, 2020.

[14] D. M. Manimekalai and G. Anupriya, "A Novel Intrusion Detection System using Data Mining Techniques," vol. 6, no. 6, p. 5, 2019.

[15] Dahiya P, Srivastava DK. 2018. Network intrusion detection in big dataset using spark. *Procedia Computer Science* 132(1-3):253-262.

[16] Dlamini G, Fahim M. 2021. DGM: a data generative model to improve minority class presence in the anomaly detection domain. *Neural Computing and Applications* 33(20):13635-13646.

[17] Faker O, Dogdu E. 2019. Intrusion detection using big data and deep learning techniques.

[18] M. Nobakht, V. Siva Raman, and R. Borelli, "A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT Using Open Flow," presented at the 2016

11th International Conference on Availability, Reliability and Security (ARES), 2016.

[19] M. S. Essayed, N. Le-Khan, S. Dev., and A. D. Jurua, "Machine Learning Techniques for Detecting Attacks in SDN," in 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), 19-20 Oct. 2019, pp. 277-281, doi: 10.1109/ICCSNT47585.2019.8962519.

[20] R. Taheri, M. Ahmadzadeh, and M. R. Kharazmi, "A New Approach For Feature Selection In Intrusion Detection System," p. 15.

[21] R. Vijayanand, D. Devaraj, and B. Kannapiran, "A Novel Deep Learning Based Intrusion Detection System for Smart Meter Communication Network," in 2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), Tamilnadu, India, Apr. 2019, pp. 1–3.