DDOS ATTACK CLASSIFICATION WITH HYPERPARAMETER TUNING AND ENCHANCED PREDICTION TECHNIQUE

A PROJECT REPORT

Submitted by

GRAYSON P [211421104081]
GIRITHARAN K [211421104075]
GANESHRAGAVAN S [211421104073]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2025

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"DDOS ATTACK CLASSIFICATION WITH HYPERPARAMETER TUNING AND ENHANCED PREDICTION TECHNIQUE"** is the bonafide work of **"GRAYSON P (211421104081), GIRITHARAN K (211421104075), GANESHRAGAVAN S (211421104073)"** who carried out the project work under my supervision.

Signature of the HOD with date	**Signature of the Supervisor with date**
DR L. JABASHEELA M.E., Ph.D.,	
	SUPERVISOR
PROFESSOR AND HEAD,	**<<Academic Designation of Supervisor>>**
Department of Computer Science and Engineering, Panimalar Engineering College, Chennai - 123	Department of Computer Science and Engineering, Panimalar Engineering College, Chennai - 123

Certified that the above candidate(s) was examined in the End Semester Project Viva-Voce

Examination held on ....

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We Grayson P (211421104081), Giritharan K (211421104075), Ganeshragavan S (211421104073) hereby declare that this project report titled “Rule-Based with machine Learning IDS for DDoS Attack Detection”, under the guidance of HEMALATHA DEVI is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project.

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Project Coordinator Dr. Senthil Kumar** and **Project Guide Dr. Hemalatha Devi** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

NAME OF THE STUDENT(S)

GRAYSON P [211421104081]
GIRITHARAN K [211421104075]
GANESHRAGAVAN S [211421104122]

ABSTRACT

Data privacy is crucial in the financial sector to safeguard clients' sensitive information, prevent financial fraud, ensure regulatory compliance, and protect intellectual property. With the rise of internet usage and digital transactions, maintaining privacy has become increasingly challenging. Distributed Denial of Service (DDoS) attacks pose a significant threat to client privacy, necessitating effective detection and prevention measures. Machine Learning (ML) offers a promising approach for enhancing cyber-attack detection systems. This paper proposes a hierarchical ML-based hyperparameter optimization technique for classifying network intrusions. Utilizing the CICIDS dataset, which includes logs of various attacks, the proposed method involves preprocessing the data with min-max scaling and SMOTE. Feature selection is carried out to identify the most significant features. Classification is then performed using XGBoost, LGBM, CatBoost, Random Forest (RF), and Decision Tree (DT) algorithms. The models' performance is evaluated using recall, precision, accuracy, and F1-score metrics.

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN		Convolutional Neural Network
LCNN		Lookup based Convolutional Neural Network
RNN	-	Recurrent Neural Network
DEX	-	Dalvik Executables
TCP	-	Transmission Control Protocol
IP	-	Internet Protocol
HTTP	-	Hyper Text Transfer Protocol
ADT	-	Android Development Tool

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 ABOUT THE PROJECT

To enhance DDoS attack detection by implementing a machine learning system with hyperparameter optimization and advanced prediction techniques, utilizing the CICIDS dataset to achieve high classification accuracy and improve network security.

1.2 SYNOPSIS

The increasing threat of DDoS attacks demands efficient and scalable detection systems to ensure network security. Existing methods, while effective to some extent, face challenges related to accuracy, scalability, and real-time performance. This research aims to overcome these limitations by proposing a hierarchical machine learning approach with hyperparameter optimization, ensuring high performance and adaptability in detecting and classifying DDoS attacks.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Exiting has only binary classification, malicious and normal. And various strategies, including machine learning (ML) techniques, have been employed for DDoS attack detection. Notable approaches include deep learning models, ensemble methods, and feature selection techniques. While these methods achieve reasonable accuracy, they often struggle with issues like class imbalance, scalability, and real-time adaptability.

2.2 PROBLEM DEFINITION

- Accuracy limitations, particularly with imbalanced datasets.
- Inadequate scalability for real-time environments.
- Higher false positive rates in certain cases.

2.3 PROPOSED SYSTEM

The proposed system aims to enhance DDoS attack multi classification using the CICIDS 2017 dataset. Key components include:

1. Data Preprocessing: Min-max scaling for normalization and SMOTE for class balancing.
2. Feature Selection: Identification of significant features to optimize model performance.

3. Classification Algorithms: XGBoost, LGBM, CatBoost, RF, and DT, coupled with hyperparameter optimization.
4. Real-Time Processing: Real-time classification of incoming data.
5. User Interface: A Flask-based web application for data upload and result display.
6. Database Connectivity: MySQL for managing user accounts and session data.

2.4 ADVANTAGES

- High classification accuracy with optimized ML models.
- Effective handling of class imbalance using SMOTE.
- Real-time DDoS detection and classification.
- User-friendly web interface for data uploads and results display.
- Scalable and efficient system design.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 INTRODUCTION

The fourth Industrial Revolution (4.0) has transformed factories into smart cyber-physical production systems (CPPS), where products, machines, and humans are interconnected across the whole supply chain. The infrastructure of CPPS leverages the utilization of various emerging technologies, such as the Internet of Things (IoT), cloud computing, edge computing, machine-to-machine (M2M) communication, and artificial intelligence (AI) [1]. These technologies have

transformed factories into massively interconnected CPPS, laying the foundation for smart factories where the whole chain, from the user to the production plant, is interconnected. This integration of networking, storage, and computing has enhanced the connectivity between different production processes and results in better product quality, increased productivity, reduced cost, and sustainability. The use of emerging technologies has also resulted in an increased security threat landscape in the operational environments of CPPS, which is constantly evolving. This integration has resulted in novel and non-negligible security challenges, such as an increase in cyber threats, financial losses, data breaches, operational disturbances, and reputation

damage. Moreover, since CPPS consists of different IoTbased systems, sensors, and smart devices from different vendors, the chances of showing additional vulnerabilities become much larger, thus leading to potential risks that malicious actors can exploit. In fact, these vulnerabilities can be exploited by the attacker to disrupt the normal operation of the system and consequently result in severe actions, such as shutting down the production line, compromising the quality of the products and causing damage to different assets that are part of the organization. Then it is desirable to deploy an IDS state-of-the-art solution that can protect the perimeter of the network and also allow the different sub-components of the industrial system to communicate with each other without causing any disturbance. In this study, we focus particularly on the application of CPPS in specifically within the farm-to-fork (F2F) specific supply chain. The F2F supply chain consists of a complete life cycle from the producers (farmers) to the end user who consumes it. As the food industry becomes more dependent on emerging digital technology, the vulnerability to cyber threats like distributed denial of service (DDoS) [2] attacks increases drastically which disturbs the whole supply chain and results in business interruption. The SecuFood project emphasizes the need and importance of analyzing the threats and vulnerabilities associated with the food and supply chain system [3]. The most common attacks that the food industry faces consist of ransomware and DDoS attacks [4]. In general, the advancement of CPPS amplifies the risk of DDoS attacks in the food industry and thus highlights the need for proactive solutions to maintain the integrity and continuation of the food supply chain. Protecting against these types of attacks is mandatory to ensure reliable communication. They can have undesirable effects on the communication infrastructure. The different stakeholders involved in the supply chain of the food industry will be unable to communicate with each other or will experience delays, which can lead to serious financial losses. The attacker can target the different components involved in the communication infrastructure which are connecting the different stakeholders of the food supply chain. The DDoS attacks can threaten the availability of production lines and overwhelm services and resources. A sudden increase in network traffic and resource utilization can affect the availability of the communication network, resulting in reduced or even worse no communication between different sub-components of the supply chain system. A DDOS attack aims to obstruct legitimate users from accessing the services and, in the worst-case scenario, results in crashing the services. The attacker can achieve its goals by flooding the service infrastructure with an intense number of packets, which can result in consuming all the network resources. In the origin of the denial-of-service (DOS) attack, the attacker emanates from a single point. It is quite straightforward to block the malicious IP as they originated from a single source, by making use of firewalls [5]. It is necessary to understand the tactics, techniques, and procedures used by the attacker to launch these attacks so that suitable mitigation actions can be taken. Various types of techniques can be used by the attacker to increase the frequency and volume of these attacks, for example, the attacker can make use of several infected devices, which are part of a botnet to target the system. An attacker can use several methodologies to launch these attacks, such as volume-based attacks, protocolbased attacks, and application layer-based attacks. Volumebased approaches focus on consuming network resources to overload the victims’ network bandwidth. Protocol-based attacks focus on exploiting vulnerabilities in the network. The application layer-based attacks disturb the communication based on the TCP/IP protocol stack. The attackers can make use of the new attack's techniques classified as zero-day DDoS by exploiting the vulnerabilities and security breaches that have not been discovered yet. The DynDNS was the largest infrastructure attack that resulted in the denial of important services [6]. Keeping the system up-to-date to avoid vulnerabilities and deploying an optimal IDS solution or firewalls such as pf-Sense, Suricate, and Wazhu, can help to protect against these attacks. In addition, the impact of these DDoS attacks and the related vulnerabilities continues to escalate uncontrollably in the context of large, distributed, and diverse systems like supply chains. This could potentially result in the emergence of new attack patterns. These attacks can cause failures in both software and hardware systems. DDoS attacks pose a significant threat to the food industry and can disrupt the entire communication infrastructure. The food industry supply chain usually has a centralized system to make communication between different stakeholders which make them more volatile towards the infrastructure DDoS attacks. They are particularly susceptible to DDoS attacks,therefore requiring advanced and modern solutions to defend against such targeted attacks. This is where the machine learning plays an important rule as it can learn malicious patterns and can handle the attacks sophisticated nature. Different types of approaches can be used to handle these attacks which includes IDS, deployment of the firewall and ML based solutions. A classical IDS solution makes use of signature-based detection or rule-based detection, which can be used as protection against known attack patterns. The current advancement into machine learning (ML) has gained popularity, particularly in the domain of anomaly detection, and consequently has shifted the trends towards the utilization of ML-based IDS. The existing research efforts [7], [8] [9] make use of ML-based techniques to handle sophisticated DDoS attacks targeting the industrial system. Major issues associated with these solutions revolve around the absence of real-time testing of developed techniques in real-time scenarios, training with only benign data, and the utilization of non-realistic attack data for training. They also faced the limitation of using attack traffic from different scenarios and the absence of crucial attack information to take the appropriate actions. Furthermore, there is no information available regarding the usability and practical utilization of these detection systems in a real-time industrial environment. Thus, it is without doubt that examining and providing protection against DDoS attacks targeting the supply chain systems demands advanced and robust solutions that can be trusted and can be used in real-time scenarios. In this paper, we developed an IDS solution for addressing the challenges associated with CPPS F2F supply chain systems against DDoS attacks. The proposed solution makes use of rule-based detection along with the utilization of ML approaches to protect against DDoS attacks. The CICFLOWMETER was used to extract the statistical features from the network traffic obtained from the F2F supply chain. These features were used to train the ML model to learn patterns about normal and malicious network flows. The ML models perform prediction at each flow level thus resulting in enhanced detection capability. While rule-based detection is used to find the known attack patterns. The utilization of rule-based detection helps in detecting known attack patterns, while ML approaches help in increasing the attack detection capability against complicated attack scenarios. The main objective of using both approaches is twofold: i) to enhance the detection capability of the IDS along with improving the decision-making process through reducing the false positive, and; ii) providing an extra check on the predictions made by ML models to make sure the results can be used to give a complete picture of the network situation. Utilization of ML techniques has achieved the detection of each individual malicious network flow but it results in generating the redundancy of alerts for the network administrator if they are not handled properly. Sometimes they can result in an increased number of false positives. The proposed methodology combines the advantages associated with rule-based detection and thus complements the detection capability of ML approaches against DDoS attacks. The ML models make use of the detection at each individual flow level and classify the normal and malicious network flows. The rule-based detection further complements these prediction results and tries to reduce the false positive generated by the ML models and provides comprehensive results along with assigning the different severity levels to the network flows based on the frequency of the flows. One main issue associated with the existing solution was that they used non-realistic data to train their system and their validation in real-time was questionable. This issue was overcome in the proposed work through the utilization of the real-time benign and attack traffic from a real-time scenario of the food industry supply chain to ensure that the proposed solution can be used effectively in real-time scenarios. Different standard evaluation matrices were used to check the efficiency of the proposed solution in a real-time manner. The main contributions of the paper can be summarized as follows: We reviewed the state-of-the-art approaches and solutions for handling the attacks in real-time industrial systems. The proposed study tried to overcome the limitations and challenges by leveraging the concepts of both traditional and novel approaches. We proposed an IDS system that was trained and validated through a real-time supply chain industrial system. A complete pipeline was developed which consisted of data acquisition, pre-processing, feature selection, model training, attack detection, and control actions. The proposed IDS system presented a hybrid approach consisting of both traditional rule-based detection and novel machine-learning approaches. The ML-based detection achieved detection at each flow level while the rule-based detection summarized the detection results. • We have evaluated the usability of our proposed IDS by deploying it in an industrial scenario F2F supply chain system. An adversarial scenario was developed to validate the efficiency of the IDS system. The system achieved an overall higher detection accuracy for both normal and adversarial scenarios. We designed an alert system, providing unified alerts along with recommended suggestions that can be used for attack mitigation. The structure of the paper is as follows. Section II provides a brief overview of existing solutions and stateof- the-art approaches for the detection of attacks on CPPS, including the studies for selected attacks. Section III presents the proposed IDS solution. This section briefly describes the working of the IDS. Section IV presents the proposed architecture deployment and the simulation-based test that was used for the validation of the trained models. Section V provides a detailed analysis of the performance matrices used for the evaluation of the trained models. Section VI provides a discussion of the overall work. Finally, Section VII gives the conclusion and describes the future work.

3.2 HARDWARE AND SOFTWARE SPECIFICATION

HARDWARE REQUIREMENTS

- Hard Disk: 500GB and above
- RAM: 4GB and above
- Processor: i3 and above

SOFTWARE REQUIREMENTS

- Programming Language: Python
- Framework: Flask (for web application)
- Database: MySQL

- Libraries: XGBoost, LGBM, CatBoost, Scikit-learn, Pandas, SMOTE, etc.

3.3 TECHNOLOGIES USED

- ➢ Python
- ➢ Machine Learning

PYTHON

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

WHAT CAN PYTHON DO?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

WHY PYTHON?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.

- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

GOOD TO KNOW

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.
- Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End Of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.
- Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python’s Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

PYTHON SYNTAX COMPARED TO OTHER PROGRAMMING LANGUAGES

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

PYTHON IS INTERPRETED

- Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
- This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
- One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
- For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
- Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

MACHINE LEARNING INTRODUCTION:

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task.
Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through learning. In its application across business problems, machine learning is also referred to as predictive analytics.

MACHINE LEARNING TASKS:

Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model from a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback. Semi algorithms develop mathematical models from incomplete training data, where a portion of the sample input doesn't have labels.

Classification algorithms and regression algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "spam" or "not spam", represented by the Boolean values true and false. Regression algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object.

In unsupervised learning, the algorithm builds a mathematical model from a set of data that contains only inputs and no desired output labels. Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as in feature learning. Dimensionality reduction is the process of reducing the number of "features", or inputs, in a set of data.

Active learning algorithms access the desired outputs (training labels) for a limited set of inputs based on a budget and optimize the choice of inputs for which it will acquire training labels. When used interactively, these can be presented to a human user for labeling. Reinforcement learning algorithms are given feedback in the form of positive or negative reinforcement in a dynamic environment and are used in autonomous vehicles or in learning to play a game against a human opponent. Other specialized algorithms in machine learning include topic modeling, where the computer program is given a set of natural language documents and finds other documents that cover similar topics. Machine learning algorithms can be used to find the unobservable probability density function in density estimation problems. Meta learning algorithms learn their own inductive bias based on previous experience. In developmental robotics, robot learning algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

SUPERVISED LEARNING:

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

In the case of semi-supervised learning algorithms, some of the training examples are missing training labels, but they can nevertheless be used to improve the quality of a model. In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

UNSUPERVISED LEARNING:

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

SEMI-SUPERVISED LEARNING:

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

CHAPTER 4

DESIGN REQUIREMENTS

4.1 DESIGN AND IMPLEMENTATION CONSTRAINTS

4.1.1 CONSTRAINTS IN ANALYSIS

- ♦ Constraints as Informal Text
- ♦ Constraints as Operational Restrictions
- ♦ Constraints Integrated in Existing Model Concepts
- ♦ Constraints as a Separate Concept
- ♦ Constraints Implied by the Model Structure

4.1.2 CONSTRAINTS IN DESIGN

- ♦ Determination of the Involved Classes
- ♦ Determination of the Involved Objects
- ♦ Determination of the Involved Actions
- ♦ Determination of the Require Clauses
- ♦ Global actions and Constraint Realization

4.1.3 CONSTRAINTS IN IMPLEMENTATION

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object-oriented methods.

4.2 OTHER NONFUNCTIONAL REQUIREMENTS

PERFORMANCE REQUIREMENTS

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service.

- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

SAFETY REQUIREMENTS

1. The software may be safety critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.

4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.

5. If a computer system is to run software of a high integrity level, then that system should not at the same time accommodate software of a lower integrity level.

6. Systems with different requirements for safety levels must be separated.

7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 5

PROPOSED SYSTEM ARCHITECTURE

5.1 ARCHITECTURE DIAGRAM:

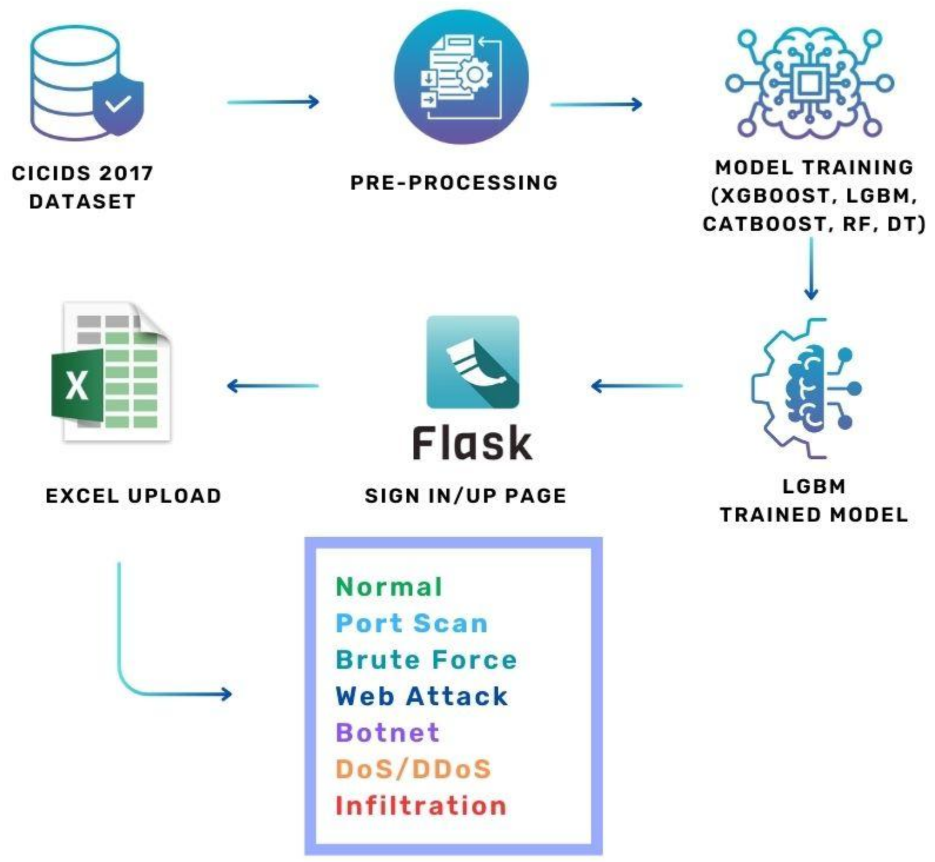

Fig: 5.1

5.2 SEQUENCE DIAGRAM:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

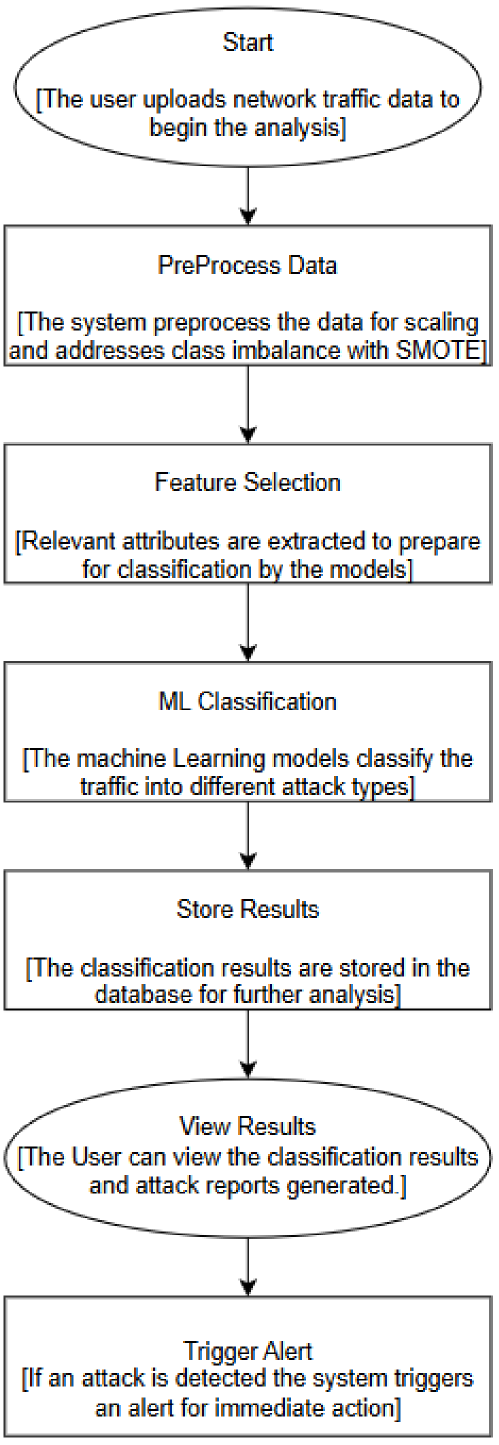
Start
[The user uploads network traffic data to begin the analysis]
PreProcess Data
[The system preprocess the data for scaling and addresses class imbalance with SMOTE]
Feature Selection
[Relevant attributes are extracted to prepare for classification by the models]
ML Classification
[The machine Learning models classify the traffic into different attack types]
Store Results
[The classification results are stored in the database for further analysis]
View Results
[The User can view the classification results and attack reports generated.]
Trigger Alert
[If an attack is detected the system triggers an alert for immediate action]

Fig 5.2

5.3 USE CASE DIAGRAM:

Unified Modeling Language (UML) is a standardized modeling language for software engineering, managed by the Object Management Group. It uses graphic notation to visualize, specify, modify, construct, and document object-oriented software systems.

USECASE DIAGRAM

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A sequence of actions that provide measurable value to an actor, represented as a horizontal ellipse.

Actor: A person, organization, or system interacting with the system.

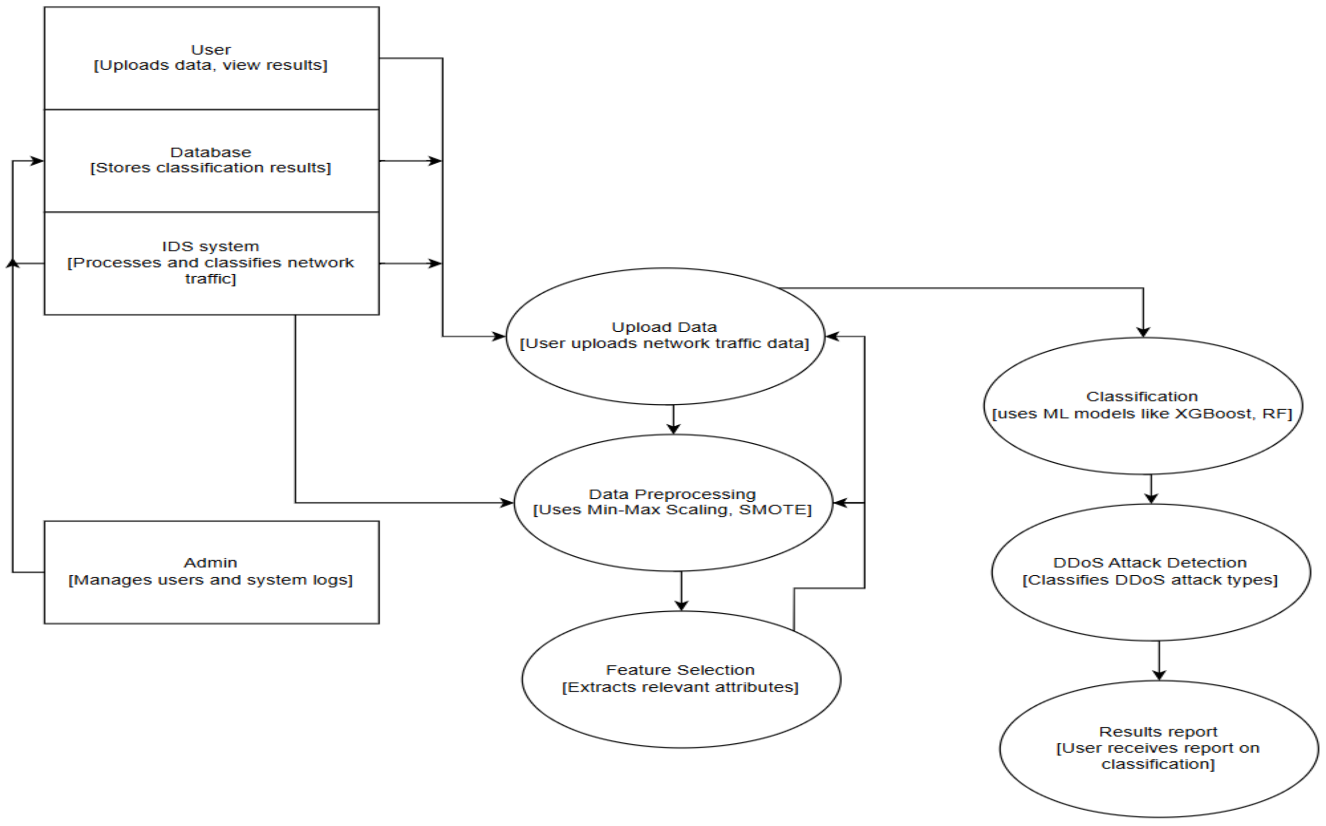
User
[Uploads data, view results]
Database
[Stores classification results]
IDS system
[Processes and classifies network traffic]
Admin
[Manages users and system logs]
Upload Data
[User uploads network traffic data]
Data Preprocessing
[Uses Min-Max Scaling, SMOTE]
Feature Selection
[Extracts relevant attributes]
Classification
[uses ML models like XGBoost, RF]
DDoS Attack Detection
[Classifies DDoS attack types]
Results report
[User receives report on classification]

Fig 5.3

5.4 ACTIVITY DIAGRAM:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

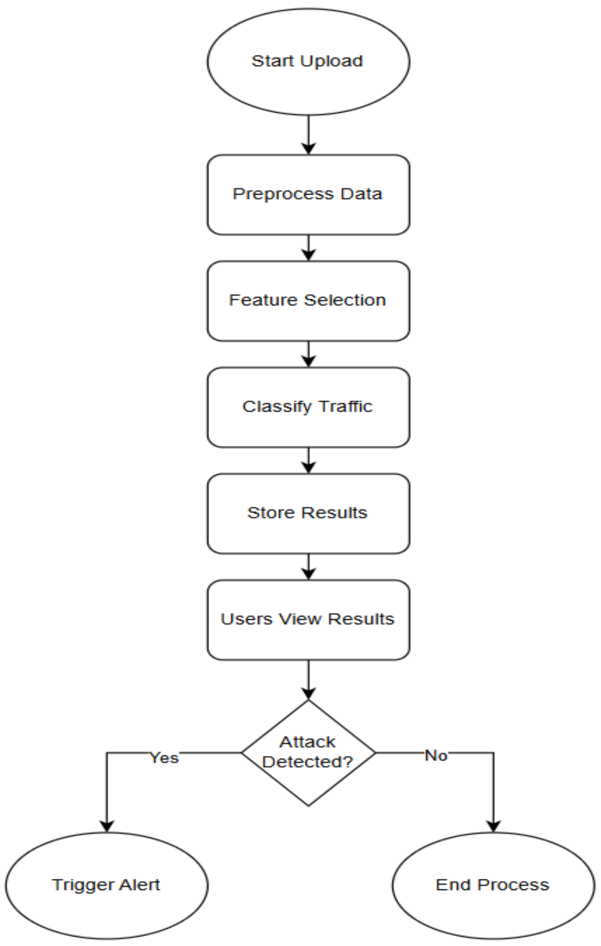

Fig 5.4

5.5 COLLABORATION DIAGRAM:

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

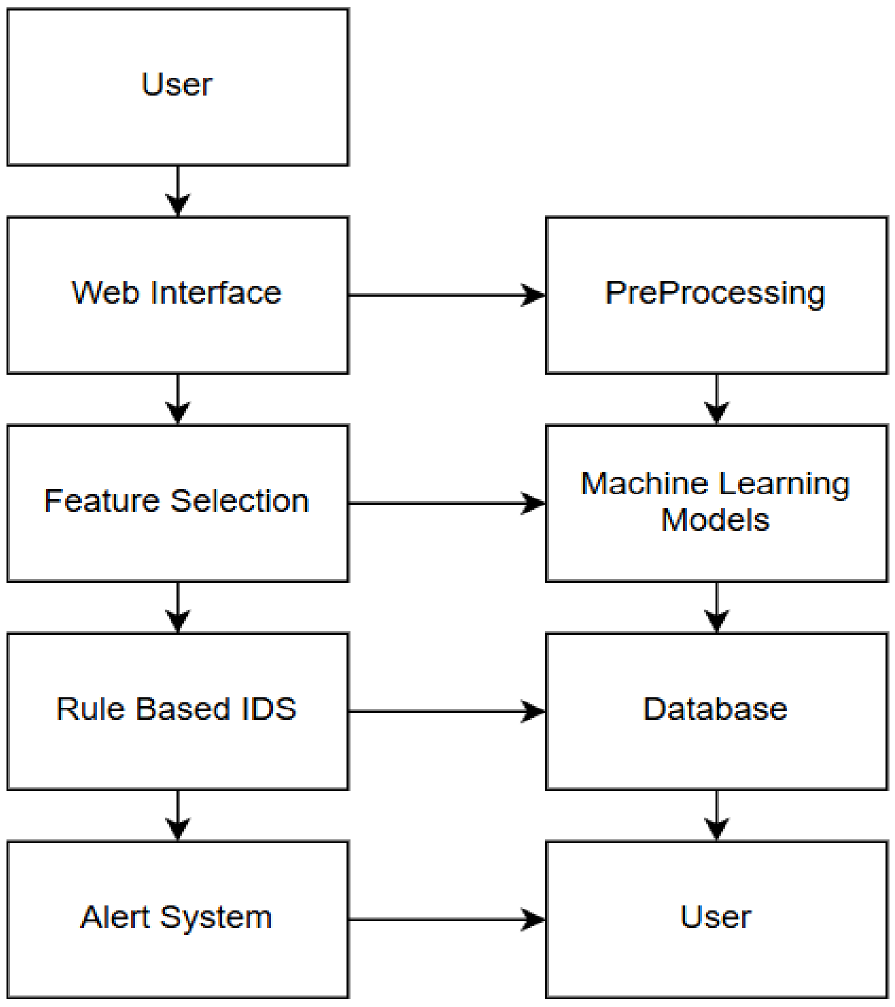

Fig 5.5

CHAPTER 6

SYSTEM DESIGNED-DETAILED

6.1 MODULES

1. **Data Preprocessing and Feature Selection**
2. **Model Training and Evaluation**
3. **Real-Time Classification and User Interface**

MODULE DESCRIPTION:

Module 1: Data Preprocessing and Feature Selection

- Data Collection: Using the CICIDS dataset for diverse attack types.
- Preprocessing: Min-max scaling for normalization and SMOTE for addressing class imbalance.
- Feature Selection: Identifying significant features for optimized performance.

Module 2: Model Training and Evaluation

- Algorithms Used: Implementation of XGBoost, LGBM, CatBoost, Random Forest (RF), and Decision Tree (DT).
- Evaluation Metrics: Accuracy, F1-score, recall, and precision to assess model performance.

Module 3: Real-Time Classification and User Interface

- Real-Time Processing: Classifying incoming data in real-time.
- User Interface (UI) with Flask: A user-friendly web application for data uploads and result display.

- Database Connectivity: MySQL for managing user accounts and session data.

CHAPTER 7

CODING AND TESTING

7.1 CODING

Once the design aspect of the system finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in each programming language. Therefore, a good coding style must be taken whenever changes are required it easily screwed into the system.

7.2 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand. Naming conventions

Value conventions Script and comment procedure Message box format Exception and error handling

NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So, it is customary to follow the conventions. These conventions are as follows:

CLASS NAMES

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

MEMBER FUNCTION AND DATA MEMBER NAME

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.

- Proper documentation of flag values.

SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

MESSAGE BOX FORMAT

When something must be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- X – User has performed illegal operation.
- ! – Information to the user.

7.3 TEST PROCEDURE

7.3.1 SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

7.4 TEST DATA AND OUTPUT

7.4.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white box testing techniques were heavily employed for unit testing.

7.4.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

7.4.3 PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

7.4.4 STRESS TEST

Stress Test is that test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the way a programmer in which a program unit breaks.

7.4.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been exercised at least once.

- Exercise all logical decisions on their true or false sides.

- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

7.4.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing and then combine them and then tested. This approach has evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files.

Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.5 TESTING TECHNIQUES / TESTING STRATERGIES

7.5.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet – undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created, and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

7.5.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been designed to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

7.5.1.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

7.5.2 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned and conducted systematically. For this reason, a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.

- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

7.5.2.1 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is "putting them together"- interfacing. There may be the chances of data lost across on another's sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

7.5.2.2 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only Errors in the condition of a program but also other a errors in the program.

7.5.2.3 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

7.5.2.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

* The function or performance characteristics confirm to specifications and are accepted.

* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

7.5.2.5 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

SOURCE CODE

SNAPSHOTS

REFERENCES

1.G. Karpagam, B. V. Kumar, J. U. Maheswari and X.-Z. Gao, Smart Cyber Physical Systems, New York, NY, USA:CRC Press, 2020.

2.H. C. Verma, S. Srivastava, T. Ahmed and N. A. Usmani, "Cyber threats in agriculture and the food industry: An Indian perspective" in Advances in Cyberology and the Advent of the Next-Gen Information Revolution, Hershey, PA, USA:IGI Global, pp. 109-122, 2023.

3.X. Koufteros and G. Lu, "Food supply chain safety and security: A concern of global importance", *J. Marketing Channels*, vol. 24, no. 3, pp. 111-114, 2017.

4. A. Yeboah-Ofori, S. Islam, S. W. Lee, Z. U. Shamszaman, K. Muhammad, M. Altaf, et al., "Cyber threat predictive analytics for improving cyber supply chain security", *IEEE Access*, vol. 9, pp. 94318-94337, 2021.

5. D. E. Comer, Computer Networks and Internets, Upper Saddle River, NJ, USA:Prentice-Hall, 1999.

6.J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow- based network traffic", *Proc. Comput. Sci.*, vol. 50, pp. 30-36, Jan. 2015.

7.F. B. Saghezchi, G. Mantas, M. A. Violas, A. M. de Oliveira Duarte and J. Rodriguez, "Machine learning for DDoS attack detection in industry 4.0 CPPSs", *Electronics*, vol. 11, no. 4, pp. 602, Feb. 2022.

8.I. A. Khan, N. Moustafa, D. Pi, Y. Hussain and N. A. Khan, "DFF-SC4N: A deep federated defence framework for protecting supply chain 4.0 networks", *IEEE Trans. Ind. Informat.*, vol. 19, no. 3, pp. 3300-3309, Mar. 2023.

9. K. Uszko, M. Kasprzyk, M. Natkaniec and P. Chołda, "Rule-based system with machine learning support for detecting anomalies in 5G WLANs", *Electronics*, vol. 12, no. 11, pp. 2355, May 2023.

10. R. R. R. Barbosa and A. Pras, "Intrusion detection in SCADA networks", *Proc. 4th Int.*

Conf. Auton. Infrastruct. Manag. Secur. (AIMS), pp. 163-166, Jun. 2010.

11. A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras and B. Stiller, "An overview of IP flow-based intrusion detection", *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 343-356, 3rd Quart. 2010.

12. P. Borges, B. Sousa, L. Ferreira, F. B. Saghezchi, G. Mantas, J. Ribeiro, et al., "Towards a hybrid intrusion detection system for Android-based PPDR terminals", *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, pp. 1034-1039, May 2017.

13. P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández and E. Vázquez, "Anomaly-based network intrusion detection: Techniques systems and challenges", *Comput. Secur.*, vol. 28, pp. 18-28, Feb./Mar. 2009.

14. S. S. Abosuliman, "Deep learning techniques for securing cyber-physical systems in supply chain 4.0", *Comput. Electr. Eng.*, vol. 107, Apr. 2023.

15. O. Linda, T. Vollmer and M. Manic, "Neural network based intrusion detection system for critical infrastructures", *Proc. Int. Joint Conf. Neural Netw.*, pp. 1827-1834, Jun. 2009.

16. M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection", *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 621-636, Mar. 2018.

17. *A realistic cyber defense dataset (CSE-CIC-IDS2018)*, Apr. 2023, [online] Available: https://registry.opendata.aws/cse-cic-ids2018.

18. I. Sharafaldin, A. H. Lashkari, S. Hakak and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy", *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, pp. 1-8, Oct. 2019.

19. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)", *Proc. Mil. Commun. Inf. Syst. Conf.*

(MilCIS), pp. 1-6, Nov. 2015.

20. N. Koroniotis, N. Moustafa, E. Sitnikova and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset", *Future Gener. Comput. Syst.*, vol. 100, pp. 779-796, Nov. 2019.

21. E. Chatzoglou, G. Kambourakis and C. Kolias, "Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset", *IEEE Access*, vol. 9, pp. 34188-34205, 2021.

22. E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment", *Sensors*, vol. 23, no. 13, pp. 5941, Jun. 2023.

23. J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu and A. O. Adetunmbi, "Network intrusion detection system using supervised learning paradigm", *Sci. Afr.*, vol. 9, Sep. 2020.

24. C. Ioannou and V. Vassiliou, "Classifying security attacks in IoT networks using supervised learning", *Proc. 15th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, pp. 652-658, May 2019.

25. A. S. Ahanger, S. M. Khan and F. Masoodi, "An effective intrusion detection system using supervised machine learning techniques", *Proc. 5th Int. Conf. Comput. Methodol. Commun. (ICCMC)*, pp. 1639-1644, Apr. 2021.

26. P. Casas, J. Mazel and P. Owezarski, "Unsupervised network intrusion detection systems: Detecting the unknown without knowledge", *Comput. Commun.*, vol. 35, pp. 772-783, Apr. 2012.

27. A. Hussain, F. Aguiló-Gost, E. Simó-Mezquita, E. Marín-Tordera and X. Massip, "An NIDS for known and zero-day anomalies", *Proc. 19th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, pp. 1-7, Apr. 2023.

28. N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets", *Sustain. Cities Soc.*, vol. 72, Sep. 2021.

29. A. Al-Abassi, H. Karimipour, A. Dehghantanha and R. M. Parizi, "An ensemble deep learning-based cyber-attack detection in industrial control system", *IEEE Access*, vol. 8, pp. 83965-83973, 2020.

30. A. Alzahrani and T. H. H. Aldhyani, "Design of efficient based artificial intelligence approaches for sustainable of cyber security in smart industrial control system", *Sustainability*, vol. 15, no. 10, pp. 8076, May 2023.