

MACHINE LEARNING BASED REAL TIME INTRUSION DETECTION SYSTEM

Submitted in partial fulfillment of the requirements of the

degree of

Bachelor of Engineering in Information Technology

By

HRITHIK GAIKWAD 17101B0051

SURAJ BALVANSHI 17101B0066

ASHUTHOSH BIST 17101B0068

Under the Guidance of

Prof. YASH SHAH

Department of Information Technology



Vidyalankar Institute of Technology
Wadala(E), Mumbai - 400437

University of Mumbai

2020-2021

CERTIFICATE OF APPROVAL

This is to certify that the project entitled

**“MACHINE LEARNING BASED REAL TIME INTRUSION DETECTION
SYSTEM”**

is a bonafide work of

HRITHIK GAIKWAD 17101B0051

SURAJ BALVANSHI 17101B0066

ASHUTHOSH 17101B0068

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of
the

degree of

Undergraduate in “INFORMATION TECHNOLOGY”.

Guide
(Prof. Yash Shah)

Head of Department
(Dr. Deepali Vora)

Principal
(Dr.S.A.Patekar)

Project Report Approval for B. E.

This project report entitled **MACHINE LEARNING BASED REAL TIME INTRUSION DETECTION SYSTEM** by

1. HRITHIK GAIKWAD (Roll No. 17101B0051)

2. SURAJ BALVANSHI (Roll No. 17101B0066)

3. ASHUTHOSH BIST (Roll No. 17101B0068)

is approved for the degree of ***Bachelor of Engineering in Information Technology***

Examiners

1. _____

2. _____

Date:

Place:

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

	Name of Student	Roll No.	Signature
1.	Hrithik Gaikwad	17101B0051	
2.	Suraj Balvanshi	17101B0066	
3.	Ashuthosh Bist	17101B0068	

Date:

ACKNOWLEDGEMENT

We are honored to present “Machine Learning Based Real Time Intrusion Detection System” as our B.E Final Year Project. We are using this opportunity to express our profound gratitude to our principal “**Dr. Sunil Patekar**” for providing us with all proper facilities and support.

We express our deepest gratitude towards our HOD “**Dr. Deepali Vora**” for his valuable and timely advice during the various phases in our project. We would like to thank our project guide “**Prof. Yash Shah**” for support, patience and faith in our capabilities and for giving us flexibility in terms of working and reporting schedules. Finally, we would like to thank everyone who have helped us directly or indirectly in our project.

We would also like to thank our staff members and lab assistant for permitting us to use computer in the lab as when required. We thank our college for providing us with excellent facilities that helped us to complete and present this project.

Hrithik Gaikwad.

Suraj Balvanshi.

Ashuthosh Bist.

EXECUTIVE SUMMARY

Internet services have become essential to business commerce as well as to individuals. With the increasing reliance on network services, the availability, confidentiality, and integrity of critical information have become increasingly compromised by remote intrusions. Enterprises are forced to fortify their networks against malicious activities and network threats. Therefore, a network system must use one or more security tools such as a firewall or an intrusion detection system to protect important data or services from intruders.

Relying on a firewall system alone is not sufficient because a firewall cannot defend the network against intrusion attempts on open ports required for network services. Hence, an intrusion detection system (IDS) is usually installed to complement the firewall. An IDS collects information from a network or computer system and analyses the information for symptoms of system breaches. A network IDS monitors network data and gives an alarm signal to the computer user or network administrator when it detects antagonistic activity on an open port. This signal allows the recipient to inspect the system for more symptoms of unauthorized network activities.

In this paper, we focus on real-time network-based intrusion detection where the incoming network data is captured on-line and the detection result is reported instantaneously or within a fraction of a minute, so that the network administrator is notified and can stop the on-going attack. Our approach also could be applied as host-based detection. There are many possible features of network data that could serve as input to an IDS, we propose to consider only 12 features of network traffic data extracted from the headers of data packets. We show that these 12 features are effective in identifying normal network activity and classifying main Denial of Service (DoS). Using a small number of features reduces the complexity of data analysis and thus can increase the detection speed and reduce computer resource (CPU and memory) consumption.

TABLE OF CONTENTS

Acknowledgement.....	vi
Executive Summary	vii
Table of Contents	viii
List of Figures.....	ix
1. Introduction	1
1.1 Basic Concept.....	2
1.2 Scope	3
1.3 Problem Statement	3
1.4 Need	3
1.5 Aim & Objectives.....	4
2. Literature Survey	5
3. System Design	11
3.1 Block Diagram	11
3.2 Proposed System	11
3.3 Methodology	13
3.3.1 Data Mining.....	13
3.3.2 Machine Learning	17
3.4 Hardware & Software Analysis	19
3.4.1 Requirement Analysis	19
3.4.2 System Design.....	19
3.5 Process Model	21

3.6	Feasibility Study	24
3.7	Data Flow Diagrams	25
3.7.1	DFD Level 0	25
3.7.2	DFD Level 1	26
3.8	UML Diagrams	27
3.8.1	Use Case Diagram	27
3.8.2	Sequence Diagram	28
3.8.3	Activity Diagram	29
4.	Planning & Scheduling	30
4.1	Gantt Chart	31
4.2	Timeline	32
4.3	Work Breakdown Structure	33
5.	Testing & Maintainance	34
5.1	Testing.....	35
5.2	Maintenance	41
6.	System Implementation	43
6.1	Results	44
6.2	Test Cases	52
7.	Conclusion & Future Scope	54
7.1	Conclusion.....	55
7.2	Future Scope.....	55
8.	References.....	

Paper Published.....

Certificates.....

List Of Figures & Table

Figures

Sr No.	Content	Page No.
3.1	Block Diagram	11
3.2	Proposed System	12
3.3	3.3.1 Dataset creation	13
	3.3.2	
	(a) Machine Learning	17
	(b) Training Phase	18
	(c) Predicting Phase	18
3.4	System design	
3.5	Process Model	22
3.6	Feasibility Study	
3.7	3.7.1 DFD Level 0	25
3.8	3.8.1 Use Case Diagram	27
	3.8.2 Sequence Diagram	28
	3.8.3 Activity Diagram	29
4.1	Gantt Chart	31
4.2	Timeline	33
5.1	Software Testing Model	36

6.1	6.1.1 Home Page	44
	6.1.2 Table creation	44
	6.1.3 Display Connection	45
	6.1.4	
	(a): Prediction of attack	46
	(b): Display Attack	46

Table

Sr No.	Content	Page No.
6.2	Test Cases	52

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1 Basic Concept

In any organization value of data may range from miniscule to invaluable. Sensitive Data leaks cause massive loss to an organization. To prevent such scenarios, various organization has started researching in improving network security to protect its valuable or sensitive data. This can be achieved by using an Intrusion Detection System (IDS) that helps to detect malicious attacks and network intrusions. An Intrusion detection system is deployed at a network junction or on a host system, an IDS can either be a host or a network-based IDS which are used according to an organization's requirement. Once a malicious activity or an intrusion is detected, it is flagged, or an alarm is raised so that the attack can be isolate.

Machine Learning techniques are widely used to improve existing systems or environments, to achieve the best possible result. Similarly using Machine Learning technique in Networking is an emerging field. Various work has been done in this field using Machine Learning techniques to achieve better result than using traditional techniques. Exponential increase in information has attracted a lot of interest in Cyber Security. Various on-going research are focused on improving existing network tools to achieve better network security. Many studies are conducted to improve Intrusion Detection System (IDS) which helps to monitor a network or a host-based system and detects malicious activities. The existing IDS have low detection rate, high false alarm rate and are unable to detect novel attacks. To overcome these problems, many researches are focused on utilizing Machine Learning methods to discover abnormal patterns to improve Detection rate.

1.2 Scope

The goal of this project is to analyze and study various Machine Learning Algorithms and traditional IDS. And design IDS using Machine Learning to improve detection rate, reduce false alarm rate and a system with a better capability of distinguishing attacks from normal packets. The model will be integrated with a User Interface (UI) which will provide a more user-friendly way to monitor the network activity.

1.3 Problem Statement

Due to the excessive increase in volume of data, protection and security of such a sensitive information is major concern of organization. IDS being an of the important security tools to fight against such malicious activity. But the traditional IDS alarm report of intrusion to network and detection accuracy gets reduced with such a large network monitoring chunk data. This is one of the major issues when the system encounters unknown attacks or zero-day attack. Due to this, IDS has decreased accuracy rate and to Increased false alarm rate.

1.4 Need

Research for improving different network security tool to provide better security is the need of time. So, an Intrusion Detection System (IDS) using different Machine Learning algorithm which can automatically discover the essential differences between normal data and abnormal data with high accuracy. In addition, machine learning methods have strong generalizability, so they are also able to detect unknown attacks. And thereby increases detections rate of attacks and reduces false alarm rate in IDS which issues is being faced by the traditional IDS. This system can help us achieve better network security.

1.5 Aim & Objectives

- **Study and analyze various Machine Learning and Deep Learning Algorithms:**

To get a clear idea about different available technique and which technique we can use in our project we study various machine learning technique and implement some using KDD and our own generated dataset

- **Study working of IDS and Network:**

Studies the traditional IDS system identified shortcomings in the system and think about ways to overcome them such as Detection rate and False alarm rate.

- **Building network monitoring tool (IDS):**

Using Python to building a network monitoring program capable of sniffing packets.

- **Improving Security with integrating Machine Learning in IDS:**

Implementing ML technique on IDS to improve Detection rate and Reduce false positive

- **Design a GUI for end user:**

For end user build a GUI which provide a user-friendly way to operate the whole operation and which helps in easy monitoring of network.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

The artificial neural network approach is one of the most popular techniques for the design of IDS. Jirapummin et al. [1] proposed a hybrid neural network using a combination of Self-Organizing Map (SOM) and Resilient Back-Propagation Neural Network (BPNN). To evaluate their approach, they used an available wellknown preprocessed dataset which is KDD99 [2]. The KDD99 dataset is a network packet dataset consisting of normal network activity as well as many network attack types. The dataset is based on the DARPA98 dataset from MIT Lincoln laboratory, which provides answer class (labeled data) for evaluation of intrusion detection. Pan et al. [3] designed a hybrid system by using a BPNN and a C4.5 Decision Tree considering the KDD99 dataset. The results showed that using only a BPNN without C4.5 Decision Tree, their system could not detect the network attack types such as User to Root (U2R) and Root to Local (R2L) at all. Moradi and Zulkernine [4] used a Multi-Layer Perceptron (MLP) artificial neural network in off-line mode to classify normal network activity, Satan (Probe) attacks and Neptune attacks using the KDD99 dataset. Ngamwittayanon et al. [5] designed a multi-state IDS system to classify normal data and each attack type using the KDD99 dataset. Their results showed a higher detection rate in each classification category than when only a single state was used to classify all categories.

Labib and Vemuri [6] developed a real-time IDS using SelfOrganizing Maps (SOM) to detect normal network activity and DoS attack. They preprocessed their dataset to have 10 features for each data record. Each record contained information of 50 packets. Their IDS was evaluated by human visualization for different Fig. 1. Network intrusion detection system environment. 2228 P. Sangkatsanee et al. / Computer Communications 34 (2011) 2227–2235 characteristics of normal data and DoS attack. No detection rate was reported. Puttini et al. [7] used a Bayesian classification model for anomaly detection to classify normal network activity and attack using a 3-month training dataset and a 1-month test dataset. They evaluated their approach by adjusting a penalty value to see how it affected the classification results. They also needed human expert to visualize the normal and abnormal network behaviors. No detection rate was reported. Amini et al. [8] designed a real-time IDS using two unsupervised neural network algorithms which are Adaptive Resonance

Theory (ART) and Self-Organizing Map (SOM). They classified two attack types plus normal data during a 4-day experiment with a 27-feature dataset, where each feature captures number of occurrences of an event in each time interval. The detection results showed that the ART-2 gave higher detection speed and detection rate than the SOM. The detection rate was reported to be over 97%, separating normal traffic data from network attacks. However, the attacks were not classified into types or categories. Su et al. [9] created a real-time network IDS using fuzzy association rules and conducted their experiments by using four computers with 30 DoS attack types in WIN32. They could separate the normal network activity from network attacks but they did not identify the attack type. They preprocessed the network data to have 16 features. After testing, the results showed that the 30 DoS attack types have a similarity ratio of less than 0.4 while normal network activity gave a similarity ratio more than 0.75. The similarity ratio represents how close or similar the data is to normal data, i.e. 1.0 means that they are perfectly matched. Li et al. [10] developed a high-speed intrusion detection model using TCP/IP header information. However, their approach is limited to only one type of attack which is DoS. A well-known intrusion detection tool called SNORT was studied in [11]. SNORT has become a commercial tool. Its attack signature rules are available only to their registered customers. The signature rules or patches have to be frequently updated and installed in order to detect current attack types. In summary, most researchers proposed IDS classification algorithms based on machine learning techniques and used KDD 99 dataset to evaluate their IDS approaches in an off-line environment without considering real-time processing/detection. The KDD99 dataset, which was created about 10 years ago, is complex and lacks of many current attack types. While this approach is of theoretical interest, it provides only post hoc assistance to network administrators and thus is less useful for building practical tools. Although a few researchers have started investigating real-time intrusion detection systems using different techniques, most of the on-line IDS performance still needs further improvement. Some on-line IDSs require a human expert to visualize or identify the attacks, while the remaining systems have other limitations. For example, some IDS can separate normal network data from attack data, but cannot classify attack type. Some IDSs were designed to detect only one attack type.

The approach should be simple but efficient in detecting network intrusion in an

actual real-time environment. We describe the concept of information gain and machine learning methods which are applied to our real-time IDS approach in the following section

CHAPTER 3

SYSTEM DESIGN

SYSTEM DESIGN

3.1 Block Diagram

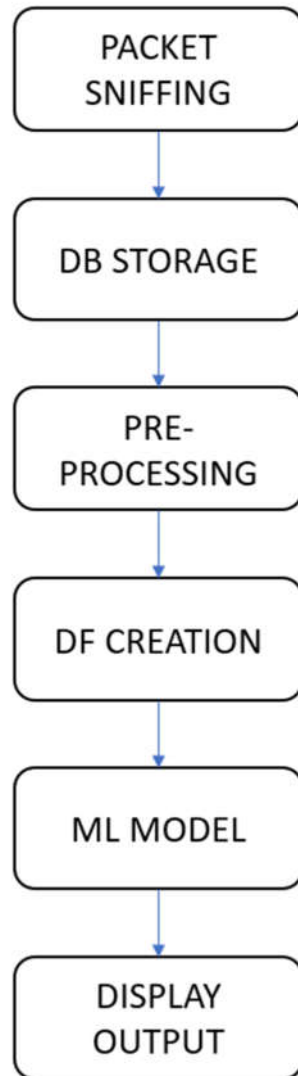


Figure 3.1: Block Diagram

3.2 Proposed System

Our Proposed system has 5 phases. 1st Phase (Packet sniffing) starts with network packet sniffing program here we used python programming language to build the network packets sniffer which scan network port on host machine and packets are sniffed. These packets are formatted respective with source IP addresses, destination IP address, protocol name and network features. The network packets are aggregated based on Source IP address and Destination IP address and these packet data will be captured and logged into a csv file in a designated folder this completes the 2nd face (DB Storage). This aggregated data in CSV will represent the network traffic flowing through the host's end. This CSV file becomes the input for the Machine Learning model which is the 3rd phase (ML model).

The 3rd phase actually has two sub process

- (a) Training: Here Machine Learning model which is trained on previously captured data by sniffer program. The csv individually produce by are collaborated and labeled together as Attack or normal (1 or 0) respectively and the ML model i.e., SVM is trained on this data set.
- (b) Implementing: The trained SVM model is used to predict whether its and attack or not and a new column is attached.

The 4th Phase (GUI): The model results are displayed on GUI which is built on python tkinter . Were the attack packets are classified as red and normal packets are shown colorless.

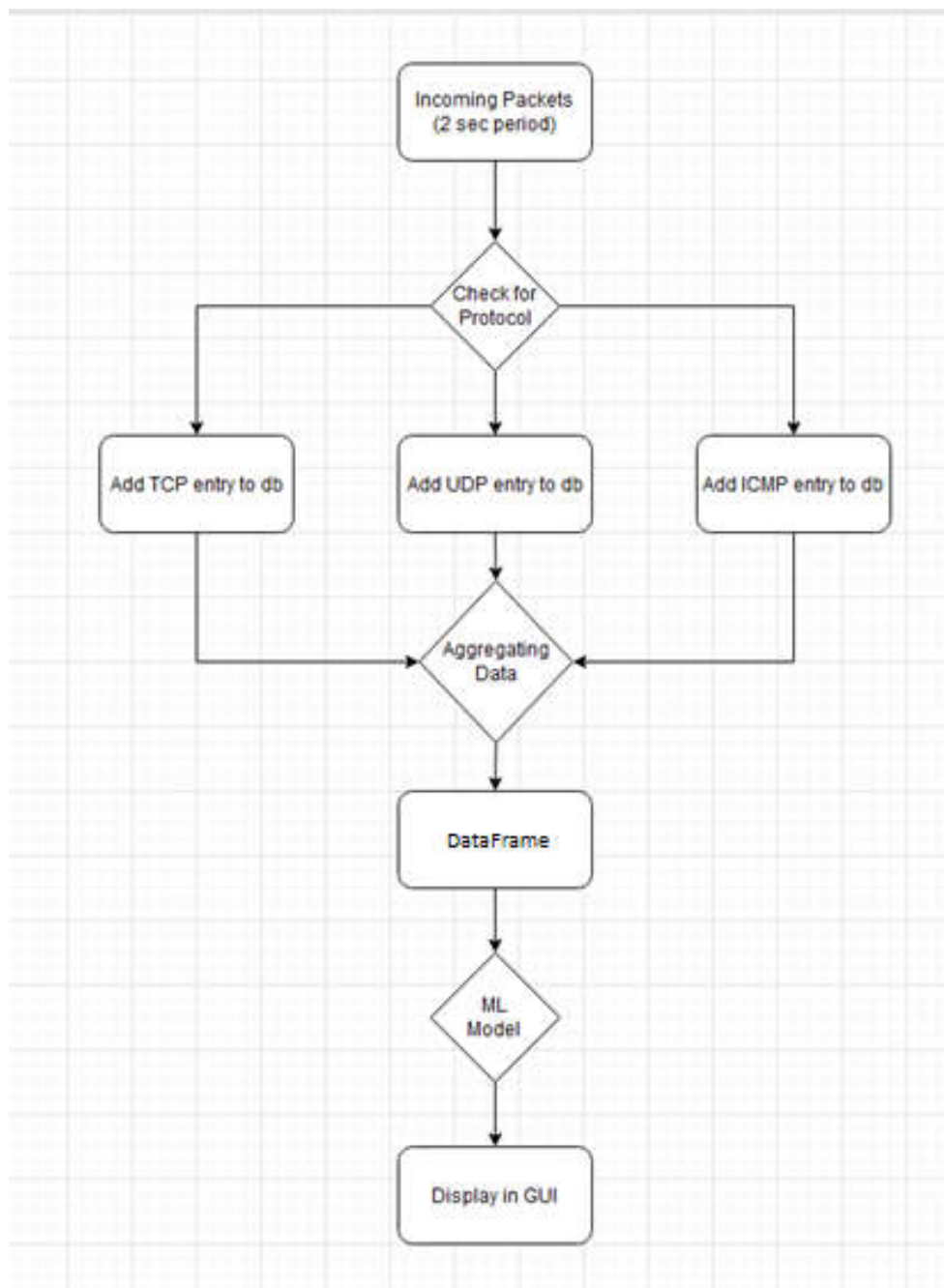


Figure 3.2: Proposed System

3.2 Methodology

The linear working of our project can be explained in the following manner:

- **Step1:** - Packet Sniffing on host machine.
- **Step 2:** - Storing data in csv
- **Step 3:** - Preprocessing data
- **Step 4:** - Machine learning to predict.
- **Step 5:** - GUI to show results.

3.3.1 Dataset Creation

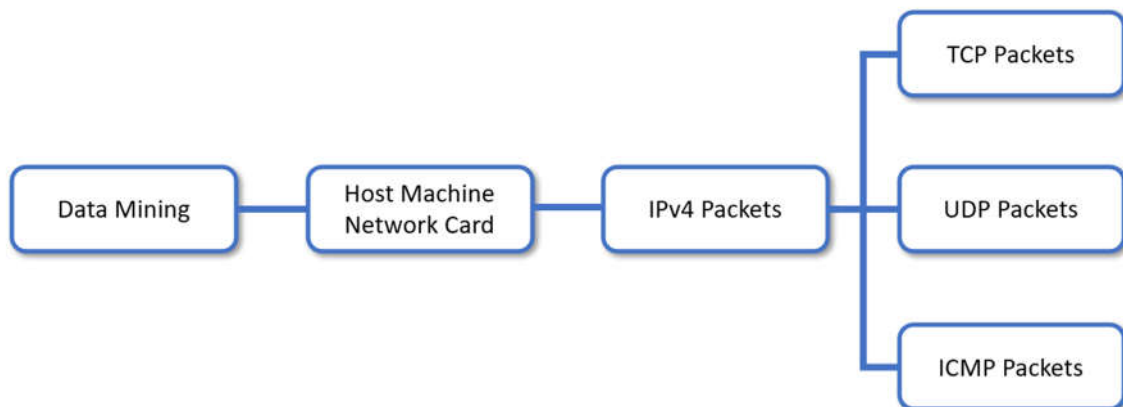


Figure 3.3.1: Data Mining Steps

- (1) We use a packet sniffer, which is built with Jpcap library, to extract network packet information including IP header, TCP header, UDP header, and ICMP header from each packet.
- (2) After that, the packet information is partitioned by considering connections between every pair of IP address (source IP and destination IP) and formed into a record by aggregating information every 2 s. Each record consists of data features considered as the key signature features representing the main characteristics of network data and activities. We performed extensive experiments to find key features that define the signatures of normal vs. attack network traffic.
- (3) We used information gain to select 12 essential features for our IDS approach. The features along with the data type and the information gain value.

No.	Feature description	Data type
1	Number of TCP packets	Integer
2	Number of TCP source port	Integer
3	Number of TCP destination port	Integer
4	Number of TCP fin flag	Integer
5	Number of TCP syn flag	Integer
6	Number of TCP push flag	Integer
7	Number of TCP ack flag	Integer
8	Number of TCP urgent flag	Integer
9	Number of UDP packets	Integer
10	Number of UDP source port	Integer
11	Number of UDP destination port	Integer
12	Number of ICMP packets	Integer

Figure 3.3.2(a): Machine Learning

- (4) We run sniffer program for 6 continues days to collect all the normal packets and attack data.
- (5) The data collected was formatted and stored in a combined csv.

3.3.2 Machine Learning

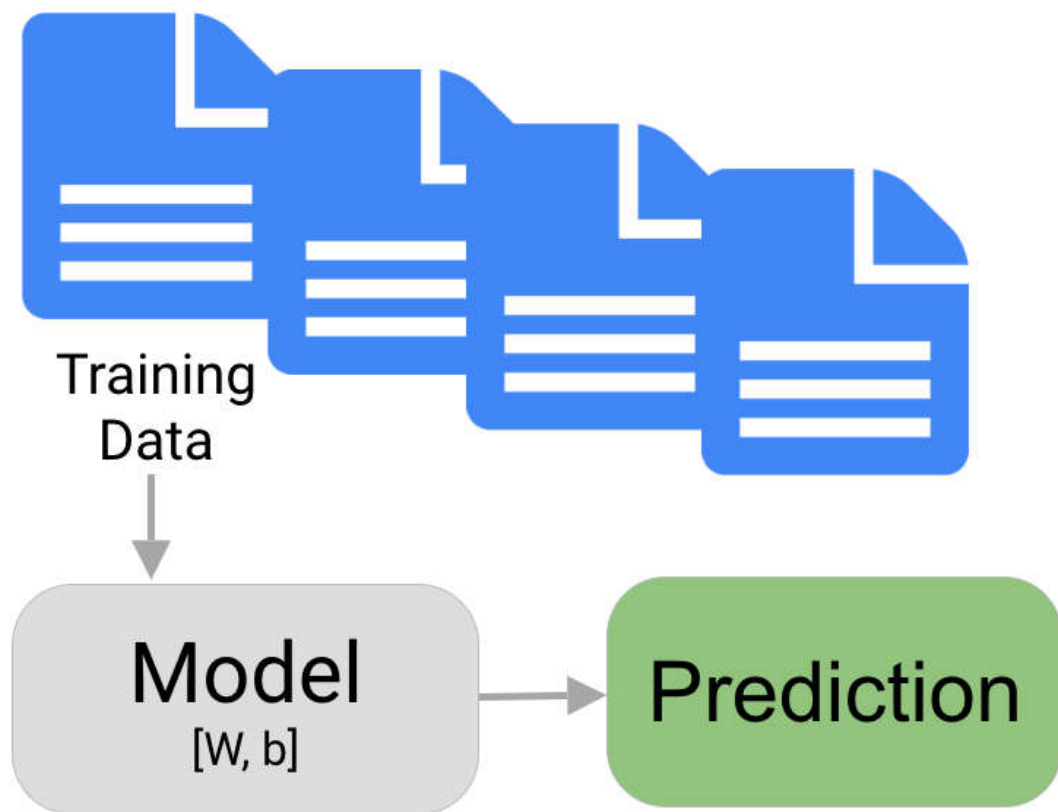


Figure 3.3.2(a): Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

An algorithm is set of instructions for solving a problem or accomplishing a task. One common example of an algorithm is a recipe, which consists of specific instructions for preparing a dish/meal. Every computerized device uses algorithms to perform its functions. ML is one of the most exciting technologies that one would have ever come across

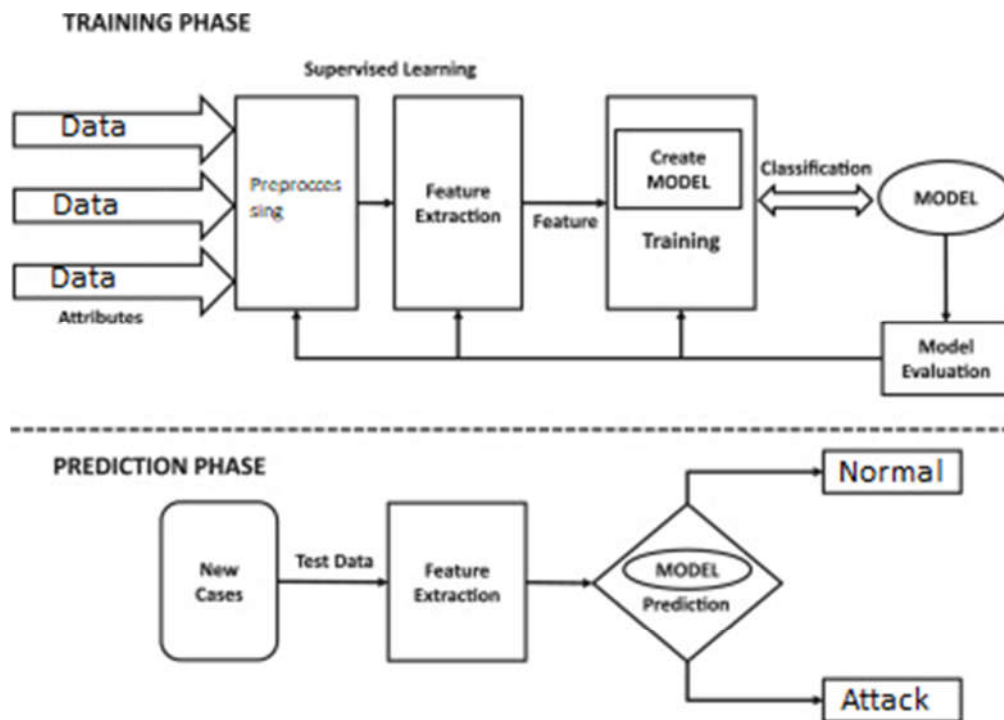


Figure 3.3.2(b): Training & Prediction Phase

Training Phase

Classification assumes labeled data: we know how many classes there are, and for each class we have examples (labeled data).

Prediction Phase

Prediction phase involves the prediction of unknown data sample.

3.3 Hardware & Software Requirements

3.3.1 Requirement Analysis

The requirement analysis lists all the required features of the project and characterizes their individual requirements. After performing all the analysis we point out the requirements for designing this software.

A] Hardware

Requirement

- Computers /Laptops
- Network Card
- Minimum 8 GB RAM.
- Quad core 2 Ghz

B] Software Requirement

- Linux OS
- Python 3.6
- Tkinter
- Vmware

3.4.1 System Design

- **Python:**

1. Pandas: The pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD.
2. Scikit-learn: Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy

- **Tkinter**

The tkinter package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.) Running `python -m tkinter` from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

1. Frames: The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way. It works like a container, which is responsible for arranging the position of other widgets. It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets. A frame can also be used as a foundation class to implement complex widgets

Syntax: `w = Frame (master, option, ...)`

2. Entry: The Entry widget is used to accept single-line text strings from a user. If you want to display multiple lines of text that can be edited, then you should use the Text widget. If you want to display one or more lines of text that cannot be modified by the user, then you should use the Label widget.

Syntax: `w = Entry (master, option, ...)`

3. The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

Syntax: `w = Button (master, option=value, ...)`

- **VMware Tools**

VMware Tools is a set of services and modules that enable several features in VMware products for better management of, and seamless user interactions with, guest operating systems. ... It includes a number of feature enhancements, driver-related enhancements, and support for new guest operating systems.

1. Kali VMware: Installing “Guest Tools”, gives a better user experience with VMware VMs. This is why since Kali Linux 2019.3, during the setup process it should detect if Kali Linux is inside a VM. If it is, then automatically install any additional tools (in VMware case, open-vm-tool).

3.5 Process Model

Process Model Used for the Project: **Waterfall Model**

The cascading effect from one phase to the other as is illustrated in figure. In this model each phase well defined starting and ending point, with identifiable deliveries to the next phase.

This model is sometimes referred to as the linear sequential model or the software life cycle.

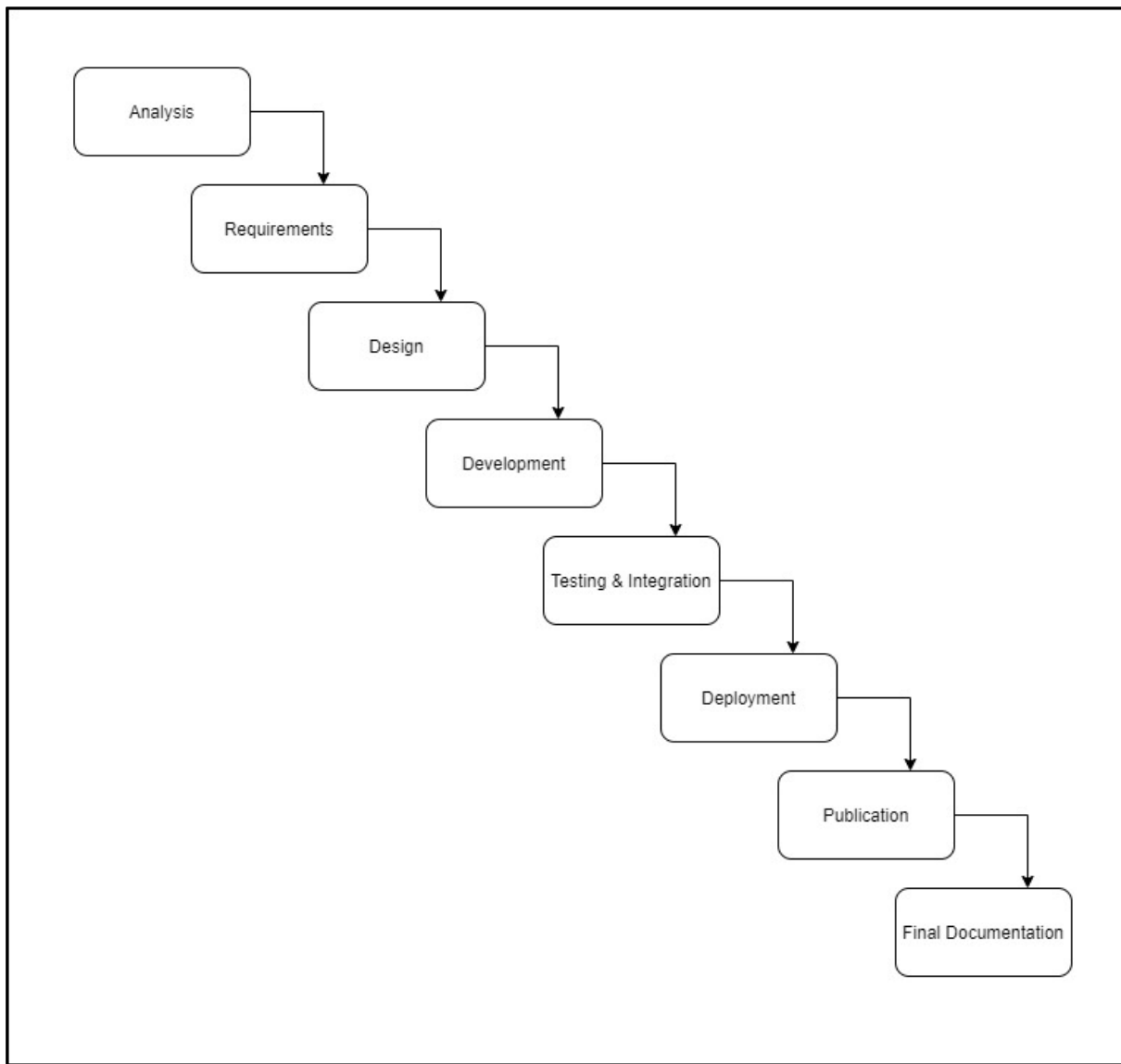


Figure 3.5 : Process Model

We intend to use the Waterfall Model in the development of our project. The reasons for using this model in our project are:

- In waterfall, development of one phase starts only when the previous phase is complete. Because of this nature, each phase of the waterfall model is quite precise well defined. Since the phases fall from a higher level to lower level, like a waterfall, it's named as the waterfall model.
- Since the requirements are stable and not changing frequently, the Waterfall Model is best suited for the project.
- It enables the monitoring and departmentalization. A timeline can be set with deadlines for each development step, and a product can proceed through the development process model phases one by one.
- The waterfall model progresses through easily understandable and

explainable phases and thus it is easy to use.

- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model, phases are processed and completed one at a time and they do not overlap.

3.6 Feasibility Study

System Feasibility:

The very first phase in any system development life cycle is early justification. In the preliminary investigations we examine the project viability, the possibility of the system being useful to the automation process. This project has been tested in the following areas of feasibility:

Operational Feasibility:

This assessment involves undertaking a study to analyse and determine whether—and how well—the organization's needs can be met by completing the project. The main objective of this project is reduced false alarm rate of IDS, increase detection rate of IDS, identify novel attacks and display report of attacks to user through a GUI.

.

Technical Feasibility:

Technical feasibility focuses on the technical resources (software and hardware) available to the organization and helps to determine whether the technical team can convert the ideas into working systems. The software required for our project are mainly open-source and readily available (e.g. Python, sklearn, tensorflow, etc.)

Economic Feasibility:

This assessment typically involves a cost/ benefits analysis of the project. The project will be developed in minimal cost as most of the software required are open source, the main cost incurred will be of the computer system and the servers to host the application.

Legal Feasibility:

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts, or social media laws. The project does not involve any legal concerns since all the licenses and laws will be respected and included in the project.

3.7 Data Flow Diagrams

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

3.7.1 DFD Level 0

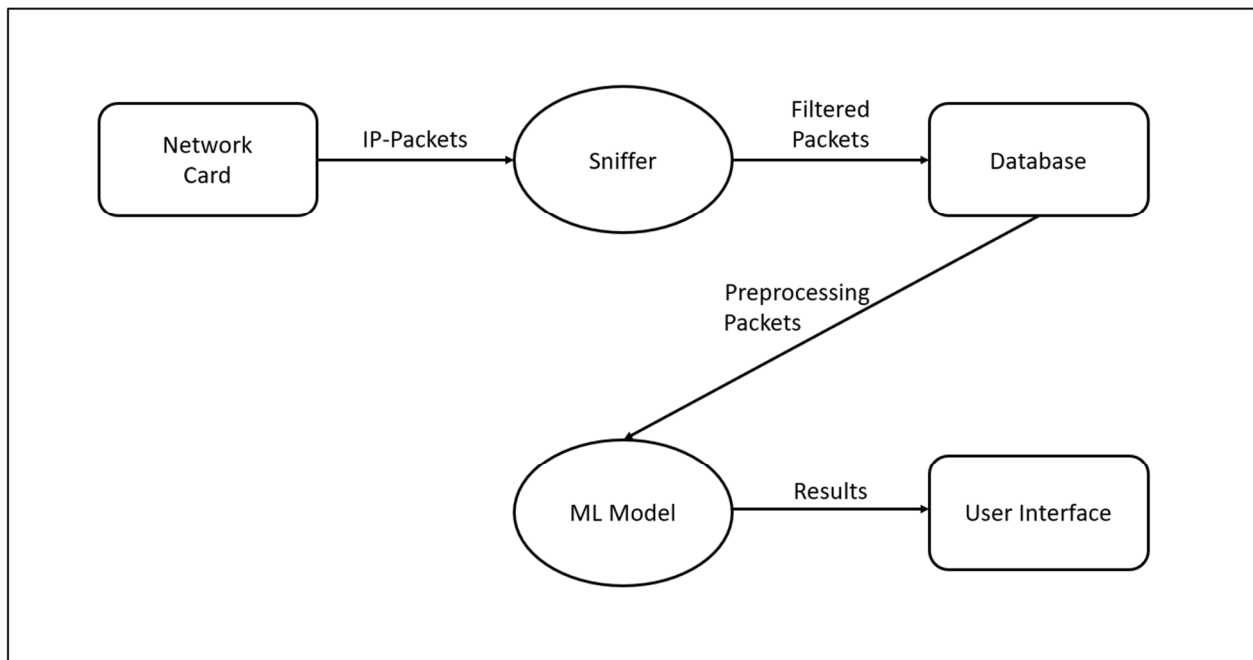


Figure 3.7.1: DFD level 0

3.8 UML Diagrams

3.8.1 Use Case Diagram

- (a) A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.
- (b) The UML diagram help us to understand how the software would be implemented and how it will be able to classify attacks and the product.
- (c) In our case the UML diagram has 4 character namely user, sniffer and converter, ML model working inside system. GUI is there for user which help him see how ML classify the attack and normal packets.
- (d) A Use Case diagram is a type of behavioral diagram defined by the UML created

From a use case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, and their goal represented as use case.

It is a type of diagram that shows a set of use cases, actors and their relationship. It should have a distinct name.

It commonly contains:

1. Use Cases
2. Actors (Primary and Secondary)
3. Dependency and Generalization.

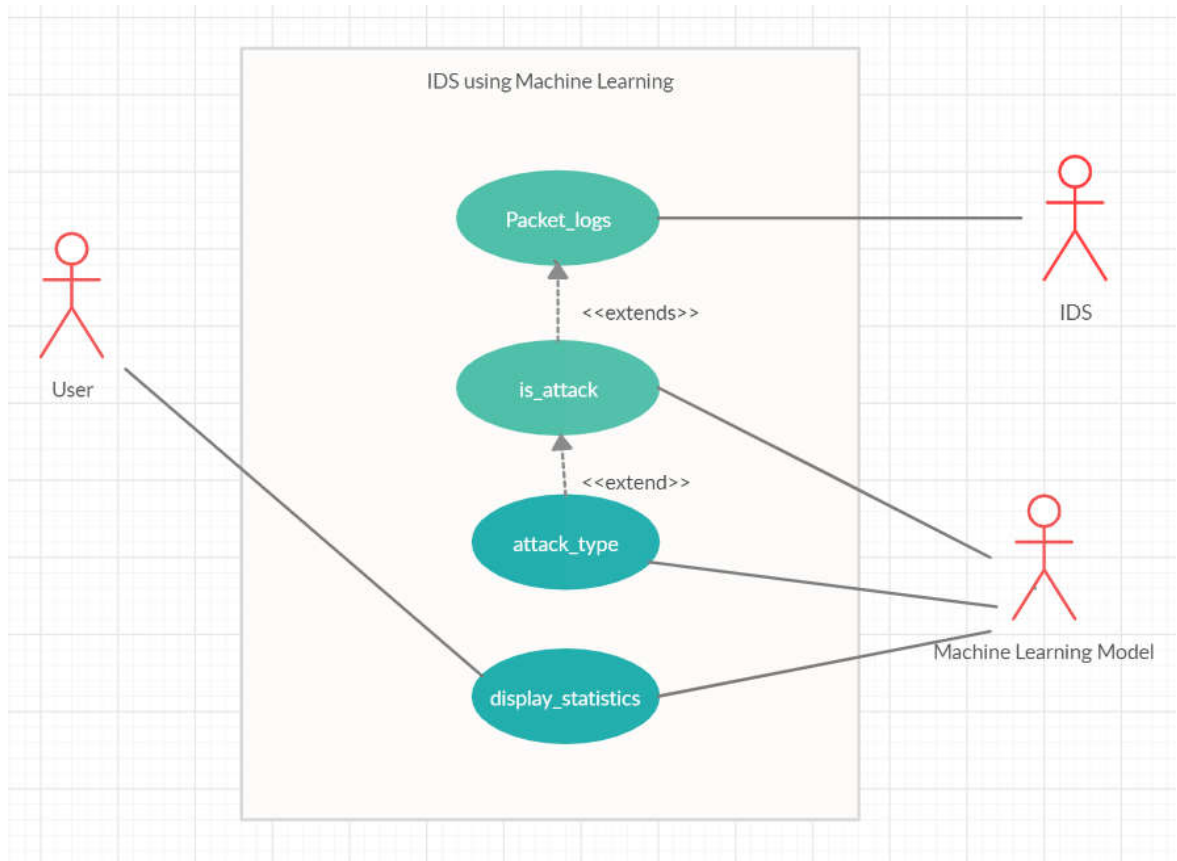


Figure 3.8.1: Use Case Diagram

3.8.2 Sequence Diagram

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. Sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagram is a structured representation of behavior as a series of sequential steps over time. It is used primarily to show the interactions between objects in the sequential order. The sequence diagram is also called as Message Sequence Chart.

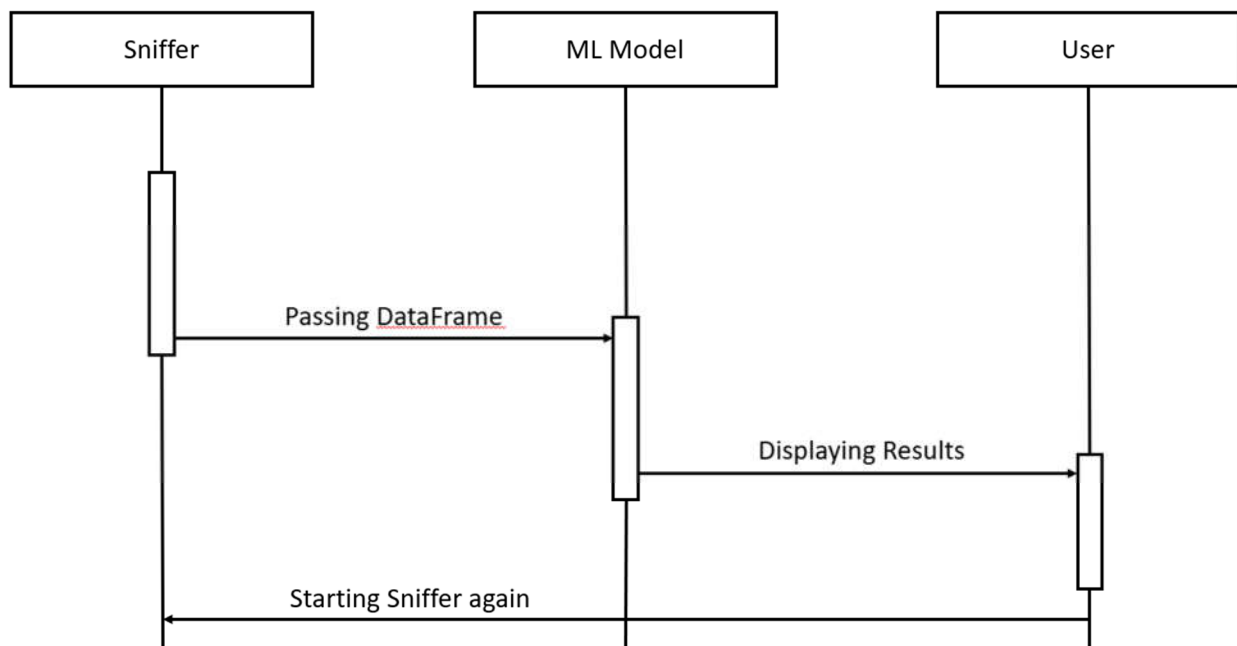


Figure 3.8.2: Sequence Diagram

3.8.3 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows).

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational process.

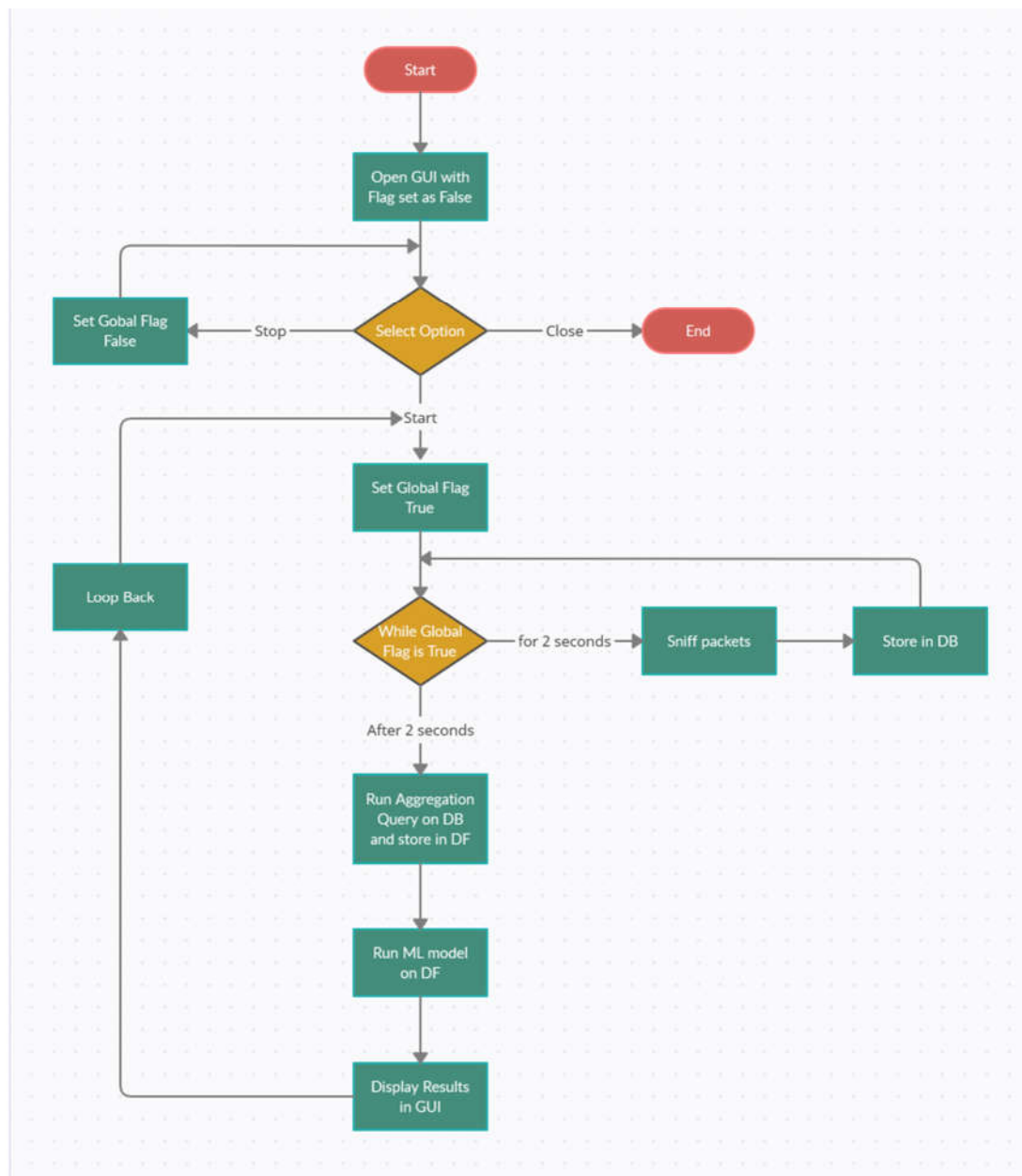


Figure 3.8.3.: Activity Diagram

CHAPTER 4

PLANNING & SCHEDULUNG

PLANNING AND SCHEDULING

4.1 GANTT CHART

A Gantt chart is a type of bar chart that illustrates a project schedule. They are the most powerful visual in project management. Gantt diagram the start date and the end date for each task that must be done in order to successfully complete a project. The scheduling included a plan for the first half of the year. The plan includes - project title finalization, literature survey, business case, project charter, requirement gathering, security planning, legal planning and user survey, implementation and testing of basic functionality, implementation and testing of GUI, implementation and testing of machine learning functionality, synopsis and report and final presentation.

GANTT chart The Gantt chart in the above diagram illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project.

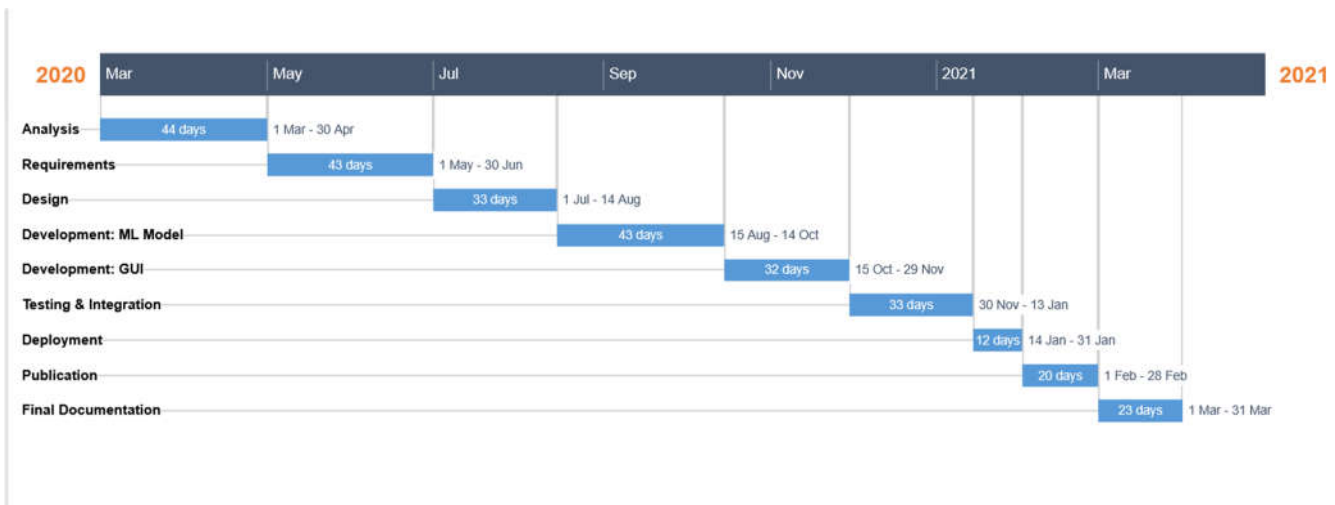


Figure 4.1 : Gantt Chart

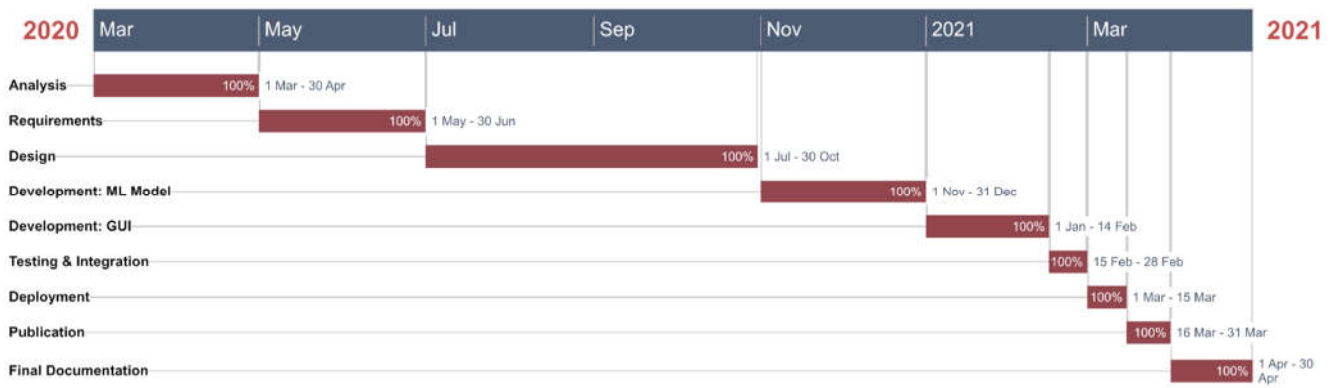


Figure 4.1 : Gantt Chart

4.3 Timeline

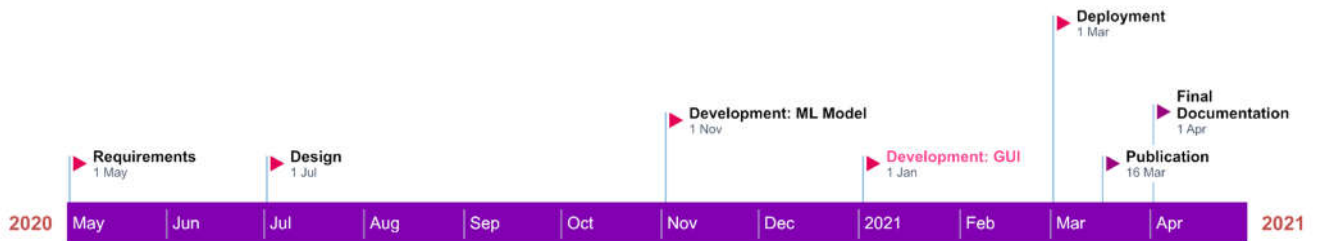


Figure 4.3 : Timeline

CHAPTER 5

TESTING & MAINTAINANCE

5.1 Testing

Methodologies Used For Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can also be stated as the process of validating and verifying that a software program/application/product:

- Meets the organizational requirement in that guided its design and development
- Works as expected; and
- Can be implemented with the same characteristics.

Primary Purpose:

Testing is to detect software failures so that defects may be discovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions.

Scope:

The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team.

Implementation:

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

Software Testing Model

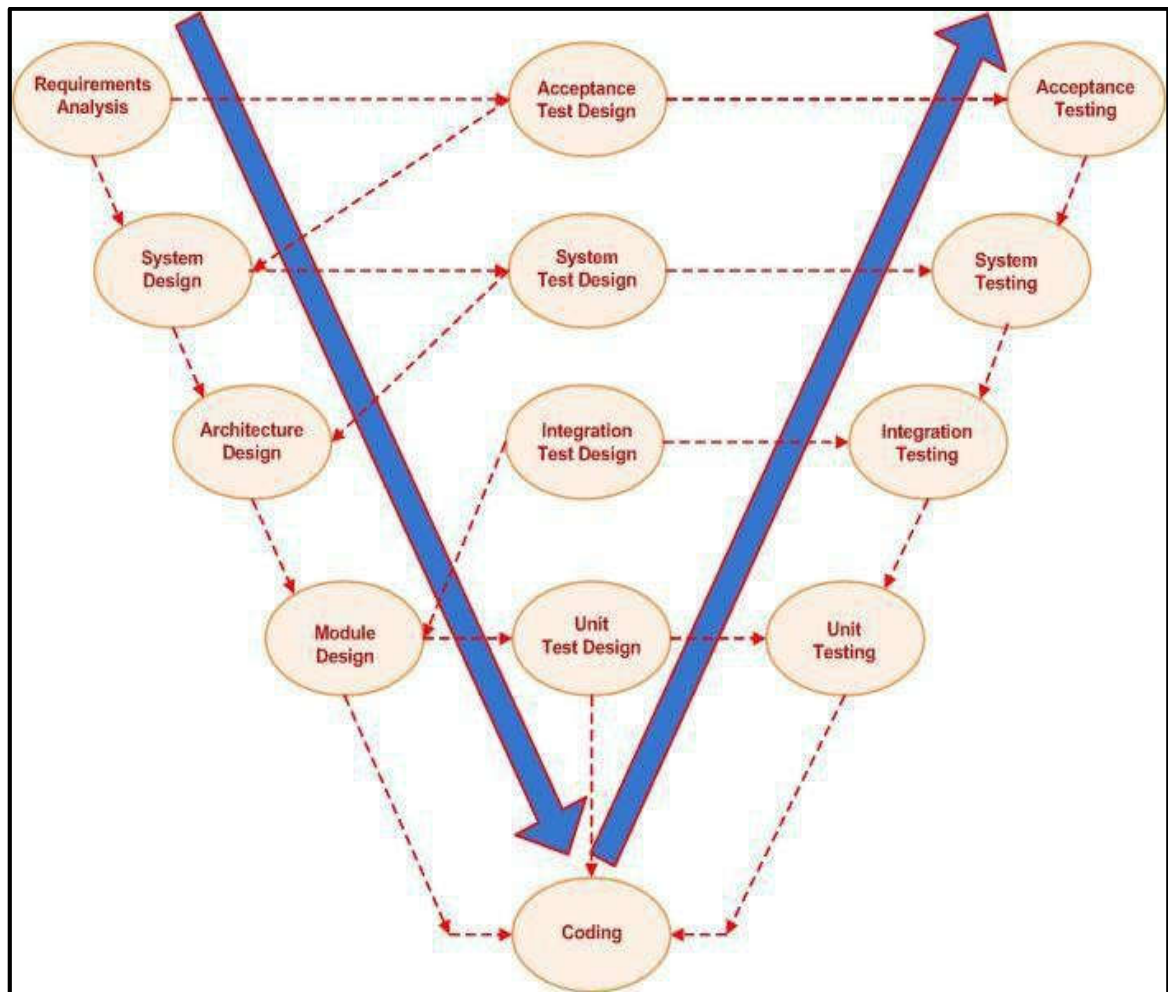


Figure 5.1 : Software Testing Model

The V-model involves building a logical V shape sequence where the testing techniques associated with the design are reflected as descending and are applied for the “verification” and connected to the requirements or specifications parts are reflected as ascending and are applied for “validation”.

The V-model ordains that the code testing documentation is written in tandem with the development phases that means, for instance, the integration tests should be documented as and when the high level design is finalized and the unit tests should be ready as and when the detailed specifications are laid down.

The idea of the V-model is to have a implementation plan for the software testing at each level namely component, interface, system, acceptance and release of the software project which need to be adhered to eliminate discrepancies in the software simultaneously rather

than waiting for the software development process to complete before handling it to the software testing professionals.

Testing Technology

System testing is a critical phase implementation. Testing of the system involves hardware device and debugging of the computer programs and testing information processing procedures. Testing can be done with text data, which attempts to stimulate all possible conditions that may arise during processing. If structured programming Methodologies have been adopted during coding the testing proceeds from higher level to lower level of program module until the entire program is tested as unit. The testing methods adopted during the testing of the system were unit testing and integrated testing.

Unit Testing:

Unit testing focuses first on the modules, independently of one another, to locate errors. This enables the tester to detect errors in coding and logical errors that is contained within that module alone. Those resulting from the interaction between modules are initially avoided.

Integration Testing:

Integration testing is a systematic technique for constructing the program structure while at the same time to uncover the errors associated with interfacing. The objective is to take unit- tested module and build a program structure that has been detected by designing. It also tests to find the discrepancies between the system and its original objectives. Subordinate stubs are replaced one at a time actual module. Tests were conducted at each module was integrated. On completion of each set another stub was replaced with the real module.

Functional Testing:

Functional testing is a technique in which all the functionalities of the program are tested to check whether all the functions that were proposed during the planning phase are full filled. This is also to check that if all the functions proposed are working properly.

This is further done in two phases:

- One before the integration to see if all the unit components work properly
- Second to see if they still work properly after they have been integrated to check some functional compatibility issues arise.

Performance Testing:

Expected Result

- The program can sniff network packet.
- Swift Pre-processing of data frame.
- ML mode predicting.
- Information passed to tkinter.

Observation

- Prefect sniffing of packets.
- No error while pre-processing.
- Accurately predicting Attack.
- Information passed without any error.

Load / Stress Testing :

Expected Result

- Response time should be unaffected irrespective of the no of packets.
- The introduction of the new attack should not make the system to work hap hazardously.
- Response time should not be degraded if there is congestion in network.

Observation

The speed of was fine even when the newer attacks were getting added. Therespone of was satisfying even with the introduction of congestion of network.

Type of Testing

White Testing

White box testing is performed based on the knowledge of how the system is implemented. White box testing includes analyzing data flow, control flow, information flow, coding practices, and exception and error handling within the system, to test the intended and unintended software behaviour. White box testing can be performed to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities.

White box testing is used to test areas that cannot be reached from a black box level. White box testing uses an internal perspective of the system to design test cases based on internal structure. It requires programming skills to identify all paths through the software. The tester chooses test case inputs to exercise paths through the code and determines the appropriate outputs.

Path coverage

Path coverage requires the execution of all different paths through the test object. This is important with respect to desktop application. Application should execute all the paths and should not crash in between.

Black Box Testing

Black-box testing is a method of software testing that tests the functionality of an application as opposed to its internal structures or workings. Specific knowledge of the application's code/internal structure and programming knowledge in general is not required.

Test cases are built around specifications and requirements, i.e. what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure. This method of test can be applied to all levels of software testing: unit, integration, functional, system and acceptance.

Black-Box testing helps to find errors such as-

- Incorrect or missing functions
- Interface errors
- Errors in data structures

Boundary value analysis

Faults often occur at the boundary of equivalence classes, because boundaries are not often defined clearly or misunderstood by programmers. Application having range fields such as date ranges are tested using this technique.

5.2 Maintenance

Maintenance is an enigma of the system development. It holds the software industry captive. Analysts spend more time in maintaining programs than coding them. Software maintenance denotes any changes made to the software product after it has been delivered to the customer. Most products need maintenance due to the wear and tear of the product. Software Maintenance can be divided into following types:

Evaluation

System Evaluation is termed as the task of evaluating the success and failure of the system. It is performed with the help of following two V's:

Verification

Verification determines whether the system is built correctly and does not contain technical errors. It also involves the review of the requirements, to verify that the right problem is being solved. Verification also ensures that the system is syntactically and logically correct and performs functionally as being specified. It is a static practice of verifying documents, design, code and program.

As verification relates to the humanized effort of checking the documents and files, we have taken utmost care to see to it that the application conforms to specifications. Reviews and inspections were carried out periodically. The tkinter application has been put through the process of Verification successfully.

Validation:

Validation on the other hand is a difficult task of insuring the meaning and content of the rules meet some carefully defined criteria of adequacy. Defining such criteria is

the key to successfully conduct Validation procedure and demonstrating the level of acceptability of the system.

As Validation is a dynamic mechanism of validating and testing the actual product; we have implemented the process of validation by executing the code thoroughly. By performing White Box as well as Black Box testing; along with Acceptance Testing, we have made sure that the application adheres to customer's expectation and requirements.

The target for validation was actual product-a unit, a module, a bent of integrated modules, and effective final product.

- **Verification** process describes whether the outputs are according to inputs or not.
- **Validation** process describes whether the software is accepted by the user or not.

CHAPTER 6

SYSTEM

IMPLEMENTATION

6.1 Results

6.1.1

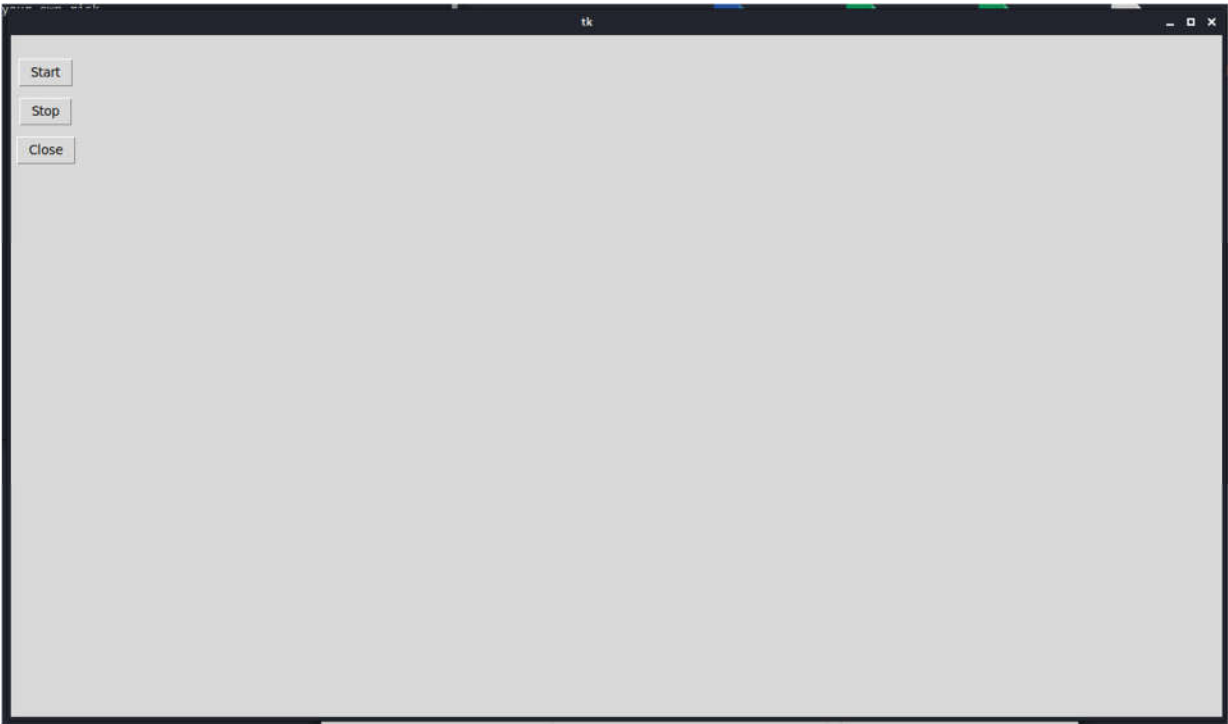


Figure 6.1.1: Home Page

6.1.2

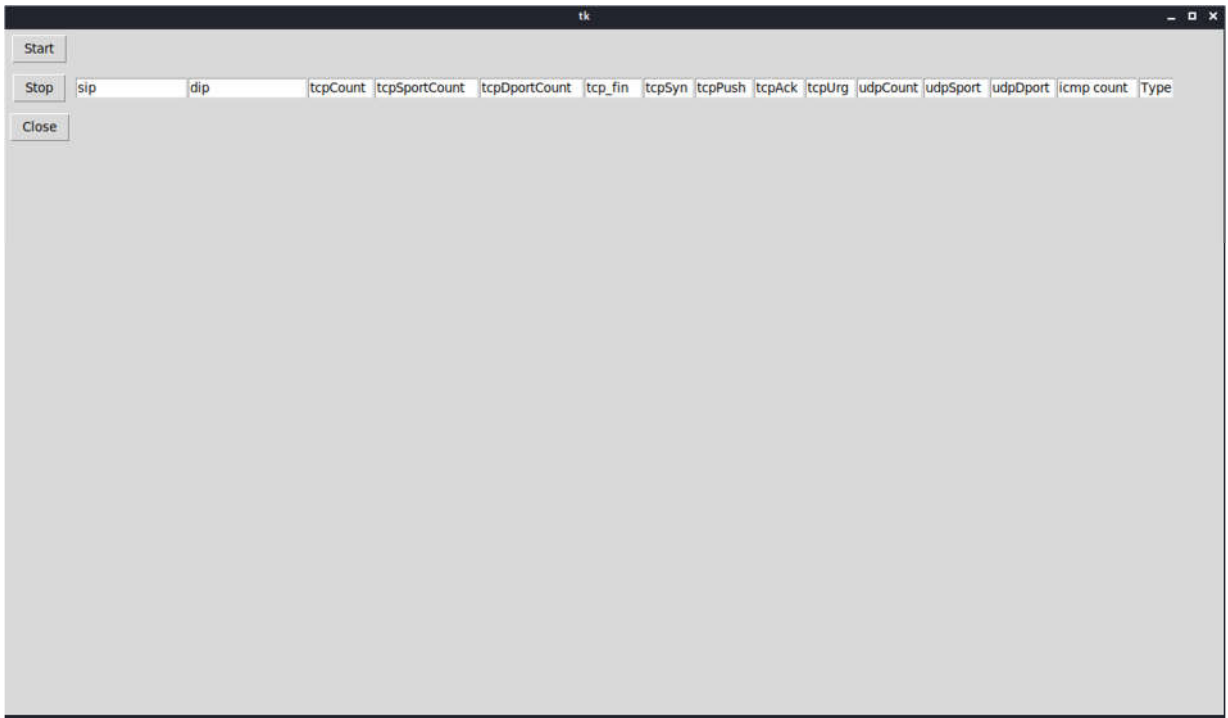


Figure 6.1.2: Table Creation

6.1.3

tk

	sip	dip	tcpCount	tcpSportCount	tcpDportCount	tcp_fin	tcpSyn	tcpPush	tcpAck	tcpUrg	udpCount	udpSport	udpDport	icmp count	Type
Start	142.250.183.174	192.168.44.129	20.0	1	1	0.0	1.0	12.0	20.0	0.0	0.0	0	0	0.0	0
	142.250.183.206	192.168.44.129	5.0	1	1	0.0	0.0	2.0	5.0	0.0	0.0	0	0	0.0	0
	142.250.192.131	192.168.44.129	13.0	1	1	0.0	0.0	8.0	13.0	0.0	0.0	0	0	0.0	0
Stop	142.250.192.35	192.168.44.129	17.0	1	1	0.0	0.0	5.0	17.0	0.0	0.0	0	0	0.0	0
	142.250.67.174	192.168.44.129	107.0	1	1	0.0	0.0	61.0	107.0	0.0	0.0	0	0	0.0	0
Close	142.250.67.238	192.168.44.129	3.0	1	1	0.0	0.0	2.0	3.0	0.0	0.0	0	0	0.0	0
	142.250.76.202	192.168.44.129	30.0	1	2	0.0	2.0	15.0	30.0	0.0	0.0	0	0	0.0	0
	142.250.77.46	192.168.44.129	26.0	1	1	0.0	0.0	9.0	26.0	0.0	0.0	0	0	0.0	0
	192.168.44.2	192.168.44.129	0.0	0	0	0.0	0.0	0.0	0.0	0.0	7.0	1	5	0.0	0
	216.58.196.68	192.168.44.129	11.0	1	1	0.0	0.0	6.0	11.0	0.0	0.0	0	0	0.0	0

Figure 6.1.3(a): Connection Display

tk

Start

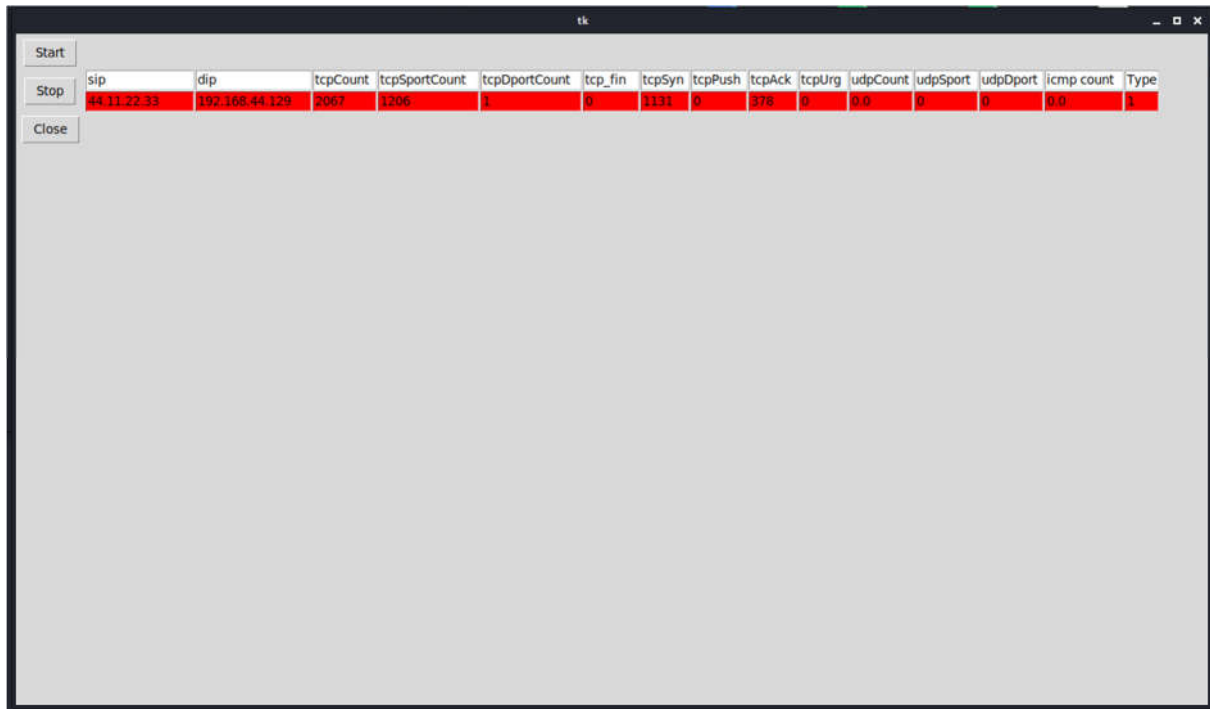
Stop

Close

sip	dip	tcpCount	tcpSportCount	tcpDportCount	tcp_fin	tcpSyn	tcpPush	tcpAck	tcpUrg	udpCount	udpSport	udpDport	icmp count	Type
142.250.183.202	192.168.44.129	1.0	1	1	0.0	0.0	1.0	1.0	0.0	0.0	0	0	0.0	0
142.250.192.131	192.168.44.129	2.0	1	1	0.0	0.0	2.0	2.0	0.0	0.0	0	0	0.0	0
142.250.192.46	192.168.44.129	5.0	1	1	0.0	0.0	4.0	5.0	0.0	0.0	0	0	0.0	0
142.250.67.174	192.168.44.129	26.0	1	1	0.0	0.0	25.0	26.0	0.0	0.0	0	0	0.0	0
142.250.76.206	192.168.44.129	14.0	1	1	0.0	0.0	11.0	14.0	0.0	0.0	0	0	0.0	0
192.168.44.2	192.168.44.129	0.0	0	0	0.0	0.0	0.0	0.0	0.0	1.0	1	1	0.0	0
44.11.22.33	192.168.44.129	1609.0	921	1	0.0	834.0	0.0	342.0	0.0	0.0	0	0	0.0	1

Figure 6.1.3(b): Prediction Of Attack

6.1.4



sip	dip	tcpCount	tcpSportCount	tcpDportCount	tcp_fin	tcpSyn	tcpPush	tcpAck	tcpUrg	udpCount	udpSport	udpDport	icmp count	Type
44.11.22.33	192.168.44.129	2067	1206	1	0	1131	0	378	0	0.0	0	0	0.0	1

Figure 6.1.4(a): Attack Packets Display

6.2 Test Cases

Test Case Id	Check Item	Test Case Objective	Steps to Execute	Expected Result	Actual Result	Result
TC-001	Sniffer	Reading TCP Packets	Start Sniffer Surf internet	Expected TCP packet header	TCP packet header recorded	PASS
TC-002	Sniffer	Reading UDP Packets	Start Sniffer Watch Stream	Expected UDP packet header	UDP packet header recorded	PASS
TC-003	Sniffer	Reading ICMP Packets	Start Sniffer Run Ping Test	Expected ICMP packet header	ICMP packet header recorded	PASS
TC-005	DB	Connection test	Connect Database and Create Cursor	A Cursor Variable to store unorganized data into the data base	Created Cursor to store unorganized data	PASS
TC-006	DB	Data Storage	Run DB method with query to store packet data into Database	Query to Store data according to TCP, UDP, ICMP header values	Created Queries to Store data according to TCP, UDP, ICMP header values.	PASS
TC-007	Aggregate Data	Data Modifying, Aggregating and fetching query	Add Aggregated query with the help of cursor	All TCP, UDP, ICMP packet converted into one dataframe	All TCP, UDP, ICMP packet converted into one dataframe	PASS

TC-008	Two Second Time Loop	Add Timer function to the Sniffer Loop	Add time variable into while loop	Import time and add time variable and integrate in while loop	Import time and add time variable and integrate in while loop	PASS
TC-009	ML	Known attack	Run Known Attack tools	Test model on Known attack tool	Test model on Known attack tool	PASS
TC-010	ML	Unknown attack	Run unknown attack tool	Test model on unknown attack tool	Test model on unknown attack tool.	PASS
TC-011	GUI	Start Button	Add button and create command for it	Change global flag to True.	Change global flag to True.	PASS
TC-012	GUI	Stop Button	Add button and create command for it	Change global flag to False	Change global flag to False	PASS
TC-013	GUI	Close Button	Add button and create command for it	Destroy root process	Destroy root process	PASS
TC-014	Table	Table without Data	Table Head Printing	Print Table Heads	Print Table Heads	PASS
TC-015	Table	Table with Data	Adding data to table	Add data after prediction to the table	Add data after prediction to the table	PASS
TC-016	GUI	Check Flag	Global Flag status for controlling Sniffer	Checks Flag after every 3 seconds	Checks Flag after every 3 seconds	PASS

CHAPTER 7
CONCLUSION &
FUTURE SCOPE

CONCLUSION & FUTURE SCOPE

7.1 Conclusion

We presented a practical and efficient real-time network intrusion detection system (RT-IDS) model which can be used with existing well-known machine learning algorithms. Our

RT-IDS model consists of three phases: the pre-processing phase, the classification phase, and the post-processing phase. We also presented how we preprocess the network packet header data into records of 12 essential features. We present an uncomplicated IDS model which can be easily applied with existing machine learning technique. We propose 12 essential features which are relevant to DoS. This small number of features can significantly improve the on-line (real-time) IDS detection speed and consumption of computer resources. We present a practical real-time, network-based IDS that not only can efficiently detect but also can classify network data into two categories which are normal and Denial of Service (DoS)

7.2 Future Scope

We plan to implement a combination technique for misuse detection and anomaly detection with the proposed 12 relevant features in order to better detect unknown attack types as and on different Probe attack.

CHAPTER 8

REFERENCES

REFERENCES

- [1] C. Jirapummin, N. Wattanapongsakorn, J. Kanthamanon, Hybrid neural networks for intrusion detection system, in: The International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Thailand, 2002, pp. 928–931. C.
- [2] W. Lee, S. Stolfo, K. Mok, Mining in a data-flow environment: experience in network intrusion detection, in: The 5th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '99), San Diego, 1999.
- [3] Z. Pan, S. Chen, G. Hu, D. Zhang, Hybrid neural network and C4.5 for misuse detection, in: The 2nd International Conference on Machine Learning and Cybernetics, China, 2003, pp.2463–2467.
- [4] M. Moradi, M. Zulkernine, A neural network based system for intrusion detection and classification of attacks, in: The IEEE International Conference on Advances in Intelligent Systems Theory and Applications, Luxembourg, 2004, pp. 148–153BM.
- [5] N. Ngamwitthayanon, N. Wattanapongsakorn, C. Charnsripinyo, D.W. Coit, Multi-stage network-based intrusion detection system using back propagation neural networks, in: Asian International Workshop on Advanced Reliability Modeling (AIWARM), Taiwan, 2008, pp. 609–619.
- [6]] K. Labib, R. Vemuri, NSOM: a real-time network-based intrusion detection system using self-organizing maps, Networks and Security (2002).
- [7] R.S. Puttini, Z. Marrakchi, L. Me, A Bayesian classification model for real-time intrusion detection, in: The 22nd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering. AIP Conference Proceedings, vol. 659, 2003, pp. 150–162
- [8] M. Amini, A. Jalili, H. Reza Shahriari, RT-UNNID: a practical solution to realtime network-based intrusion detection using unsupervised neural networks Computer & Security 25(2005)459–468.

- [9] M-Y. Su, G-J. Yu, C-Y. Lin, A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach, *Computers and Security* 28 (2009) 301–309
- [10] Z. Li, Y. Gao, Y. Chen, HiFIND: a high-speed flow-level intrusion detection approach with DoS resiliency, *Computer Networks* 54 (2010) 1282–1299
- [11] S. Chakrabarti, M. Chakraborty, I. Mukhopadhyay, Study of snort-based IDS, in: *International Conference and Workshop on Emerging Trends in Technology (ICWET)*, Mumbai, India, 2010, pp. 43–47.

PAPER PUBLISHED

CERTIFICATES



**International Conference on
Convergence of Smart Technologies
IC2ST-2021**

9th and 10th January 2021, Pune, India

**CERTIFICATE
OF PARTICIPATION**

This is to certify that Mr. Hrithik Gaikwad from *VIT, Mumbai University*
has attended and presented paper titled *SVM and NN
Machine Learning model on KDDCUP99 Dataset*

in the International Conference on Convergence of Smart
Technologies (IC2ST-2021).

A handwritten signature in blue ink, appearing to read "V. Kharat".

General Chair

Dr. Vilas Kharat
SPPU, Pune, India.

A handwritten signature in blue ink, appearing to read "B. Mareschal".

General Co-Chair

Dr. Bertrand Mareschal
ULB, Belgium



**International Conference on
Convergence of Smart Technologies
IC2ST-2021**

9th and 10th January 2021, Pune, India

**CERTIFICATE
OF PARTICIPATION**

This is to certify that *Mr. Suraj Balvanshi from VIT, Mumbai University*
has attended and presented paper titled *SVM and NN*

Machine Learning model on KDDCUP99 Dataset

in the International Conference on Convergence of Smart
Technologies (IC2ST-2021).

A handwritten signature in blue ink, appearing to read "V. Kharat".

General Chair

Dr. Vilas Kharat
SPPU, Pune, India.

A handwritten signature in blue ink, appearing to read "B. Mareschal".

General Co-Chair

Dr. Bertrand Mareschal
ULB, Belgium



**International Conference on
Convergence of Smart Technologies
IC2ST-2021**

9th and 10th January 2021, Pune, India

**CERTIFICATE
OF PARTICIPATION**

This is to certify that *Mr. Ashutosh Bist from VIT, Mumbai University*
has attended and presented paper titled *SVM and NN*

Machine Learning model on KDDCUP99 Dataset

in the International Conference on Convergence of Smart
Technologies (IC2ST-2021).

A handwritten signature in blue ink, appearing to read "V. Kharat".

General Chair

Dr. Vilas Kharat
SPPU, Pune, India.

A handwritten signature in blue ink, appearing to read "B. Mareschal".

General Co-Chair

Dr. Bertrand Mareschal
ULB, Belgium.



**International Conference on
Convergence of Smart Technologies
IC2ST-2021**

9th and 10th January 2021, Pune, India

**CERTIFICATE
OF PARTICIPATION**

This is to certify that *Prof Yash Shah from VIT, Mumbai University*

has attended and presented paper titled *SVM and NN*

Machine Learning model on KDDCUP99 Dataset

in the International Conference on Convergence of Smart
Technologies (IC2ST-2021).

A handwritten signature in blue ink, appearing to read "V. Kharat".

General Chair

Dr. Vilas Kharat
SPPU, Pune, India.

A handwritten signature in blue ink, appearing to read "B. Mareschal".

General Co-Chair

Dr. Bertrand Mareschal
ULB, Belgium