AN INTELLIGENT SOFTWARE DEFINED NETWORK CONTROLLER FOR DETECTING DISTRIBUTED DENIAL OF SERVICE ATTACK

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE AWARD OF DEGREE OF BACHELORS OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

N.SREE LAKSHMI	**2210315531**
CH.HARIDEEP	**2210315505**
P.SESHA SAI	**2210315534**
V.AKSHITH	**2210315558**

UNDER THE GUIDANCE OF

Mrs. G. RATHNAMMA,

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF TECHNOLOGY

GITAM (Deemed to be university)

HYDERABAD

APRIL-2019

SCHOOL OF TECHNOLOGY
GITAM (Deemed to be university)
HYDERABAD .

Rudraram Village,PatancheruMandal,
Sangareddy Dist – 502329,TS,INDIA.
Ph: 08455-220556/57; Fax :08455-20046
Website :www.gitam.edu

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **AN INTELLIGENT SOFTWARE DEFINED NETWORK CONTROLLER FOR DETECTING DISTRIBUTED DENIAL OF SERVICE ATTACK** was presented satisfactorily to Department of Computer Science and Engineering, SCHOOL OF TECHNOLOGY, GITAM (DEEMED TO BE UNIVERSITY), HYDERABAD by **N.SREE LAKSHMI, CH.HARIDEEP, P.SESHA SAI,V.AKSHITH**

Bearing H.T-NO's **2210315531, 2210315505, 2210315534, 2210315558** in partial fulfilment of requirement for their mini project work carried out under my guidance and supervision.

Mrs.G.Rathnamma
Assistant Professor
(Project Guide)

B.RajendraPrasad Babu
ProjectCoordinator

Dr.S.Phani Kumar
Head of the Department
Dept ofCSE

Name and Signature of the Project PanelMembers:

1.
2.
3.

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Mrs.G.RATHNAMMA** for her vital suggestions, meticulous guidance and constant motivation which is taking us a long way in the successful completion of this project.

We extend our sincere thanks to **Mr.B.Rajendra Pasad Babu** for providing us with the opportunity to achieve our goal.

We cannot move on without thanking **Mr.Dr.S.Phani Kumar**(CSE-HOD), for creating the required academic environment which made our task possible.

Onamoralpersonalnote,ourdeepestappreciationandgratitudetoourbelovedparents, and faculties who have been an inspiration and have provided us with unrelenting encouragement andsupport.

DECLARATION

We, **N.SREE LAKSHMI, CH.HARIDEEP, P.SESHA SAI, V.AKSHITH**, bearing roll numbers, **2210315531, 2210315505, 2210315534, 2210315558**, hereby declare that the project report entitled **"AN INTELLIGENT SOFTWARE DEFINED NETWORK CONTROLLER FOR DETECTING DISTRIBUTED DENIAL OF SERVICE ATTACK",** under the guidance of **Mrs.G.RATHNAMMA, ASSISTANT PROFESSOR,** Department of Computer Science and Engineering, School of Technology, GITAM (Deemed to be University), Hyderabad, has submitted the project evaluation.

This is a record of bonafide work carried out by us and the content embodied inthis projecthavenotbeenreproduced/copiedfromanysource.Thecontentembodiedin this project report have not been submitted to any other university or institute for the award of any other degree ordiploma.

NAME	PIN NO	SIGNATURE
1. N.SREE LAKSHMI	**2210315531**	
2. CH.HARIDEEP	**2210315505**	
3. P.SESHA SAI	**2210315534**	
4. V.AKSHITH	**2210315558**	

CONTENTS

ABSTRACT…..... i

LISTOFFIGURES ii

LIST OFSCREENSHOTS iii

ABBREVIATIONS…..... iv

1. INTRODUCTION 1

1.1 SOFTWAREDEFINEDNETWORKS..... 1

1.2 EXISTINGSYSTEM… 2

1.3 PROPOSEDSYSTEM… 3

1.4 OBJECTIVE OFTHEPROJECT… 3

2. LITERATURESURVEY…..... 5

2.1 APPLY SDN ARCHITECTURE TO5GSLICING 5

2.2 IMPLEMENTATION CHALLENGESOFSDN 5

2.3 A SURVEY AND A LAYERED TAXONOMYOFSDN..... 6

2.4 A SURVEY OF SDN:PAST, PRESENT&FUTURE…..... 7

2.5 DDOS ATTACK DETECTION IN SDN WITHCLOUDCOMPUTING…..... 7

3. ANALYSIS 9

3.1 PROBLEMDEFINITION..... 9

3.2 SOFTWAREREQUIREMENTSPECIFICATION…..... 9

3.2.1EXTERNALINTERFACEREQURIMENTS……11

3.3 PROCESS MODEL USEDWITHJUSTIFICATION..... 12

4. DESGN……14

4.1 SYSTEM ARCHITECTURE……14

4.2 SYSTEM FLOW……16

5. IMPLEMENTATION……18

5.1 FLOW CHART……18
5.1.1 NETWORK SETUP……18
5.2 EXECUTION CODE……20
6. TESTING……..30
6.1 SYSTEM TESTING……..30
6.2 MODULE TESTING……31
6.3 INTEGRATION TESTING……31
6.4 ACCEPTANCE TESTING……31
7. RESULT ANALYSIS……33
7.1 SCREEN SHOTS……33
8. CONCLUSION……45
9. REFERENCES……46

ABSTRACT

Software Defined Network (SDN) architecture is a new and novel way of network management mechanism. In SDN, switches do not process the incoming packets like conventional network computing environment. They match for the incoming packets in the forwarding tables and if there is none it will be sent to the controller for processing which is the operating system of the SDN. A Distributed Denial of Service (DDoS) attack is a biggest threat to cyber security in SDN network. The attack will occur at the network layer or the application layer of the compromised systems that are connected to the network. In this paper a machine learning based intelligent method is proposed which can detect the incoming packets as infected or not. The different machine learning algorithms adopted for accomplishing the task are Naive Bayes, K-Nearest neighbor (KNN) and Support vector machine (SVM) to detect the anomalous behavior of the data traffic. These three algorithms are compared according to their performances and KNN is found to be the suitable one over other two. The performance measure is taken here is the detection rate of infectedpackets.

LIST OF FIGURES

Fig 1UMBERLLAMODEL.....……12

Fig 2 GRAPHS OF DIFFERENT ALGORITHMS……13

Fig 3SDNARCHITECTURE.....….14

Fig 4OPENFLOWSEARCH .....…17

Fig 5PROPOSEDMODELFLOWCHART .....…18

Fig 6NETWORKTOPOLGY .....….19

LIST OF SCREEN SHOTS

SESSIONCHARTS 41

ABBREVIATION

SDN :- SOFTWARE DEFINED NETWORK

DDOS :- DISTRIBUTED DENIAL OF SERVICE

DOS :- DENIAL OF SERVICE

An Intelligent Software defined Network Controller for preventing Distributed Denial of Service Attack

1. INTRODUCTION

1.1 Software DefinedNetwork

SDN is presently is one of the most focused area among contemporary emerging network computing paradigms and gaining wide acceptance from both fraternity of academia and industry. The Open Networking Foundation (ONF) is a non profitable conglomerate whose work deals with task of development, standardization, and tradability of SDN. ONF has given a most appropriate and precise definition of SDN which is quoted in the subsequent lines as follows: “In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized ,and the underlying network infrastructure is abstracted from the applications. ONF has also given a high-level architecture for SDN which is represented as a three layers present vertically stacked together. The layers are infrastructure layer, control layer and application layer. The infrastructure layer is otherwise known as the data plane consisting of forwarding Elements (FEs) which may include physical switches or virtual switches. The physical switches could be like Juniper, Junos, MX-series, and the virtual switches could be Open v switch. These switches can be accessed through an open interface by which packet forwarding could be done. The control layer is otherwise known as control plane that comprises of a set of software-based SDN controllers which works as a controller. It contains some open APIs which are responsible for supervising the network forwarding features. The three main interfaces present in controller are: southbound, northbound and east/westbound interfaces. The topmost layer is endowed with end user applications like network virtualization, mobility management, security application and so on. The high level architecture of software defined network given by open flow consortium is depicted in figure. To provide on demand service to a customer SDN technology can play a major role. Seamlessly, making the availability of resources and services to customers is a requirement which is always challenged by Denial of Service (DOS) and Distributed Denial of Services (DDOS). The main aim of DoS and DDOS attacks are to put together a machine or network resource unavailable to its target users. DDOS attacks are caused by more than one persons, or bots, and DoS attacks are created by one person or a system. So a DOSattack

can be considered as a form of DDOS attack. DDOS attack occurs when an unexpectedly huge amount of data traffic is seen to be generated and forwarded from a particular server to an intended target computer to make that a victim of the attack. In this case there is chance of a sudden drop in performance in the network [9, 10]. In this paper we have proposed an intrusion detection model which works in network layer and can identify an incoming packet coming from a particular source machine is a malicious one or not. If it is so, instead of forwarding the same packet to further it can be dropped. By dropping the malicious packet, theDDOSattackcanbepreventedtoacertainextent.Themaincontributionsofthispaper can be summarized as the following points given below. Using a SoftwareDefined Network controller to detect DDOS attack in the network layer. Designing an intelligent intrusion detection model which can identify the malicious and non-maliciousincoming packets from the packet characteristics. Using machine learning algorithms to design a classifier which can classify the intruded and nonintruded packets. The rest of the paper is organized as follows. Section 2 represents the related work. Section 3 illustrates the proposed model. Section 4 represents the experimental setup and results of the simulation work and the inferences are present in section 5. Section 6 concludes the paper with also provides the future workdescription.

1.2 Existing System

SDN is presently is one of the most focused area among contemporary emerging network computing paradigms and gaining wide acceptance from both fraternity of academia and industry. The Open Networking Foundation (ONF) is a non profitable conglomerate whose work deals with task of development, standardization, and tradability of SDN. In SDN, switches do not process the incoming packets like conventional network computing environment. They match for the incoming packets in the forwarding tables and if there is none it will be sent to the controller for processing which is the operating system of the SDN. A Distributed Denial of Service (DDoS) attack is a biggest threat to cyber security in SDN network. The attack will occur at the network layer or the application layer of the compromised systems that are connected to the network. The main aim of DoS and DDOS attacks are to put together a machine or network resource unavailable to its target users. DDOS attacks are caused by more than one persons, or bots, and DoS attacks are created by one person or a system.

An intelligent software defined network controller for preventing distributed denial of service attack

Disadvantages of Existing System

1. Performance measure islow
2. Detecting packets are not fullysatisfied
3. Lowaccuracy

1.3 ProposedSystem

In this paper a machine learning based intelligent method is proposed which can detect the incoming packets as infected or not. The different machine learning algorithms adopted for accomplishing the task are Naive Bayes, K-Nearest neighbor (KNN) and Support vector machine (SVM) to detect the anomalous behavior of the data traffic. These three algorithms are compared according to their performances and KNN is found to be the suitable one over other two. The performance measure is taken here is the detection rate of infected packets. The data are collected in two groups. From which one set is taken from normal case and the other set is for the infected one. The features of data which are considered as inputs to the learning models are number of Packets, Protocol, Delay, Bandwidth, Source IP and Destination IP.

Advantages of Proposed System

1. High accuracy
2. It‟s correctly identified to an infected class to total number of infectedsamples.

Here the generated data are interpreted and understood.

1.4 Objective of theProject

Software Defined Network (SDN) architecture is a new and novel way of network management mechanism. In SDN, switches do not process the incoming packets like conventional network computing environment. They match for the incoming packets in the forwarding tables and if there is none it will be sent to the controller for processing which is the operating system of the SDN. A Distributed Denial of Service (DDoS) attack is a biggest threat to cyber security in SDN network. The attack will occur at the network layer or the application layer of the compromised systems that are connected to the network. In this paper a machinelearning basedintelligentmethodisproposedwhichcandetecttheincoming

packets as infected or not. The different machine learning algorithms adopted for accomplishing the task are Naive Bayes, K-Nearest neighbor (KNN) and Support vector machine (SVM) to detect the anomalous behavior of the data traffic. These three algorithms are compared according to their performances and KNN is found to be the suitable one over other two. The performance measure is taken here is the detection rate of infected packets.

2. LITERATURESURVEY

2.1 Applying SDN Architecture to 5GSlicing

The architecture of software defined networking (SDN) is defined in ONF TR-502 [1] and TR521 [2]. By providing a complete view of all resources required to serve a business purpose, the SDN architecture supports the key principles of Slicing in a network. As a conceptual framework for a standardized platform supporting Network Slicing, it can serve as one of the technical building blocks to fulfill the business requirements for the fifth generation of mobile technology (5G). In the 5G White Paper [4], NGMN has laid out business demands and introduced the service deployment concept of NetworkSlicing.

The SDN architecture provides a foundation for a standardized platform supporting slicing, including a unified control plane for an adaptive data plane, as an enabler to support the 5G business requirements. While the SDN architecture and its resource abstractions apply to wire line, wireless and mobile networking contexts, it is applicable for 5G, particularly as the technical concept of SDN enables realizing the business purpose of slicing in a network system. The vision for 5G and motivation for SDN match. For future business opportunities system flexibility is a key requirement, i.e. to not only support concurrent and/or currently emerging use cases, but even those not yet envisioned.

2.2 Implementation Challenges for Software-DefinedNetworks

Cloud services are exploding and organizations are converging their data centres in order to take advantage of the predictability, continuity, and quality of service delivered by virtualization technologies. In parallel, energy-efficient and high-security networking is of increasing importance. Network operators, service and product providers require a new network solution to efficiently tackle the increasing demands of this changing network landscape. Software-Defined Networking has emerged as an efficient network technology capable of supporting the dynamic nature of future network functions and intelligent

applications while lowering operating costs through simplified hardware, software, and management.

SDN has emerged as a means to improve programmability within the network to support the dynamic nature of future network functions. As bandwidth demand escalates, the provision of additional capabilities and processing power with support for multiple 100GE channels will be seamless through an SDN-based update and/or upgrade. SDN promises flexibility, centralized control and open interfaces between nodes enabling an efficient, adaptive network. In order to achieve this goal, a number of outstanding challenges must be resolved. In this article we have presented a discussion of a number of challenges in the area of performance, scalability, security and interoperability. Existing research and industry solutions could resolve some of these problems and a number of working groups are also discussing potential solutions. In addition to these, the hybrid programmable architecture could be a means to counter performance and scalability issues introduced by SDN. The objective of the model is to optimize flow processing in thenetwork.

2.3 A Survey and a Layered Taxonomy of Software-DefinedNetworking

Software-defined networking (SDN) has recently gained unprecedented attention from industry and research communities, and it seems unlikely that this will be attenuated in the near future. The ideas brought by SDN, although often described as a “revolutionary paradigm shift” in networking, are not completely new since they have their foundations in programmable networks and control–data plane separation projects. SDN promises simplified network management by enabling network automation, fostering innovation through programmability, and decreasing CAPEX and OPEX by reducing costs and power consumption. In this paper, we aim at analyzing and categorizing a number of relevant research works toward realizing SDN promises. We first provide an overview on SDN roots and then describe the architecture underlying SDN and its main components. Thereafter, we present existing SDN-related taxonomies and propose a taxonomy that classifies the reviewed research works and brings relevant research directions intofocus.

SDN has recently gained an unprecedented attention from academia and industry. SDN was born in academia. Several important organizations such as Google and VMware are running SDN networks and several experimental testbed are running and testing Open Flow networks worldwide, including NDDI , OFELIA , FIBRE, JGN-X and GENI . Thus, a survey on SDN

that studies various aspects of this novel networking paradigm was needed. In this paper, we elaborated a thorough survey and tutorial on SDN to investigate the potential of SDN in revolutionizing networks as we know them today. First, we went back to the roots from where SDN and Open Flow haveemerged.

2.4 A Survey of Software-Defined Networking: Past, Present, and Future of ProgrammableNetworks

The idea of programmable networks has recently re-gained considerable momentum due to the emergence of the Software-Defined Networking (SDN) paradigm. SDN, often referred to as a "radical new idea in networking", promises to dramatically simplify network management and enable innovation through network programmability. This paper surveys the state-of-the-art in programmable networks with an emphasis on SDN. We provide a historic perspective of programmable networks from early ideas to recent developments. Then we present the SDN architecture and the Open Flow standard in particular, discuss current alternatives for implementation and testing of SDN-based protocols and services, examine current and future SDN applications, and explore promising research directions based on the SDN paradigm.

In this paper, we provided an overview of programmable networks and, in this context, examined the emerging field of Software-Defined Networking (SDN). We look at the history of programmable networks, from early ideas until recent developments. In particular we described the SDN architecture in detail as well as the Open Flow standard. We presented current SDN implementations and testing platforms and examined network services and applications that have been developed based on the SDN paradigm. We concluded with a discussion of future directions enabled by SDN ranging from support for heterogeneous networks to Information Centric Networking (ICN).

2.5 Distributed Denial of Service (DDOS) Attack Detection in Software Defined Networking with CloudComputing

Cloud computing has almost swept the IT industry replacing the enterprises‟ traditional computing. With the recent developments in cloud computing has also grown the threats to the security of cloud services and data. Amongst the many security issues, DDoS has its way in threatening the availability of the services provided by the cloud to the intended users. To

detect and mitigate these attacks, a classifier algorithm and a forecasting algorithm are used respectively. With these algorithms deployed in SDN architecture, we can greatly reduce the DDoS attacks in the cloud thereby providing the cloud users uninterrupted services.

In this paper, we first discussed the reasons why DDoS attacks are growing in cloud computing environments. Then we summarized the difficulty in defeating DDoS attacks in cloud computing environments. In addition, we presented some good features of SDN-based cloud in defeating DDoS attacks and discussed some challenges of SDN-based cloud. Since SDN-based cloud is still in its concept phase, we provided a comprehensive survey on some of the works that have already been done to defend DDoS attacks using SDN. We categorized the existing methods in three different class and presented a thorough comparison.

3. ANALYSIS

3.1 ProblemDefinition

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering,information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of softwaredevelopment methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process. **3.2.**

3.2 Software RequirementSpecification

A Software Requirements Specification (SRS) – a requirements specificationfor a softwaresystem is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctionalrequirements are requirements which impose constraints on the design or implementation (such as performance engineeringrequirements, qualitystandards, or designconstraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- Business requirements describe in business terms *what* must be delivered or accomplished to provide value.

- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of businessrequirements.)

- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminaryinvestigation:

- **ECONOMICFEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility forcertain.

- **OPERATIONALFEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization‟s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performancestatus.

- **TECHNICALFEASIBILITY**

Earlier no system existed to cater to the needs of „Secure Infrastructure Implementation System‟. The current system developed is technically feasible. It is a web based userinterface

for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database‟s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability andsecurity.

3.2.1 External InterfaceRequirements

User Interface

The user interface of this system is a user friendly Java Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through Java capabilities.

SoftwareInterfaces

The required software is JAVA1.6.

Operating Environment

Windows XP,Linux.

HARDWARE REQUIREMENTS:

- Processor - Pentium–IV
- RAM - 256MB(min)
- HardDisk - 20 GB
- KeyBoard - Standard Windows Keyboard
- Mouse - Two or Three ButtonMouse
- Monitor - SVGA

SOFTWARE REQUIREMENTS:

- OperatingSystem : Windows XP
- ProgrammingLanguage : Java

3.3 PROCESS MODEL USED WITHJUSTIFICATION

SDLC (UmbrellaModel):

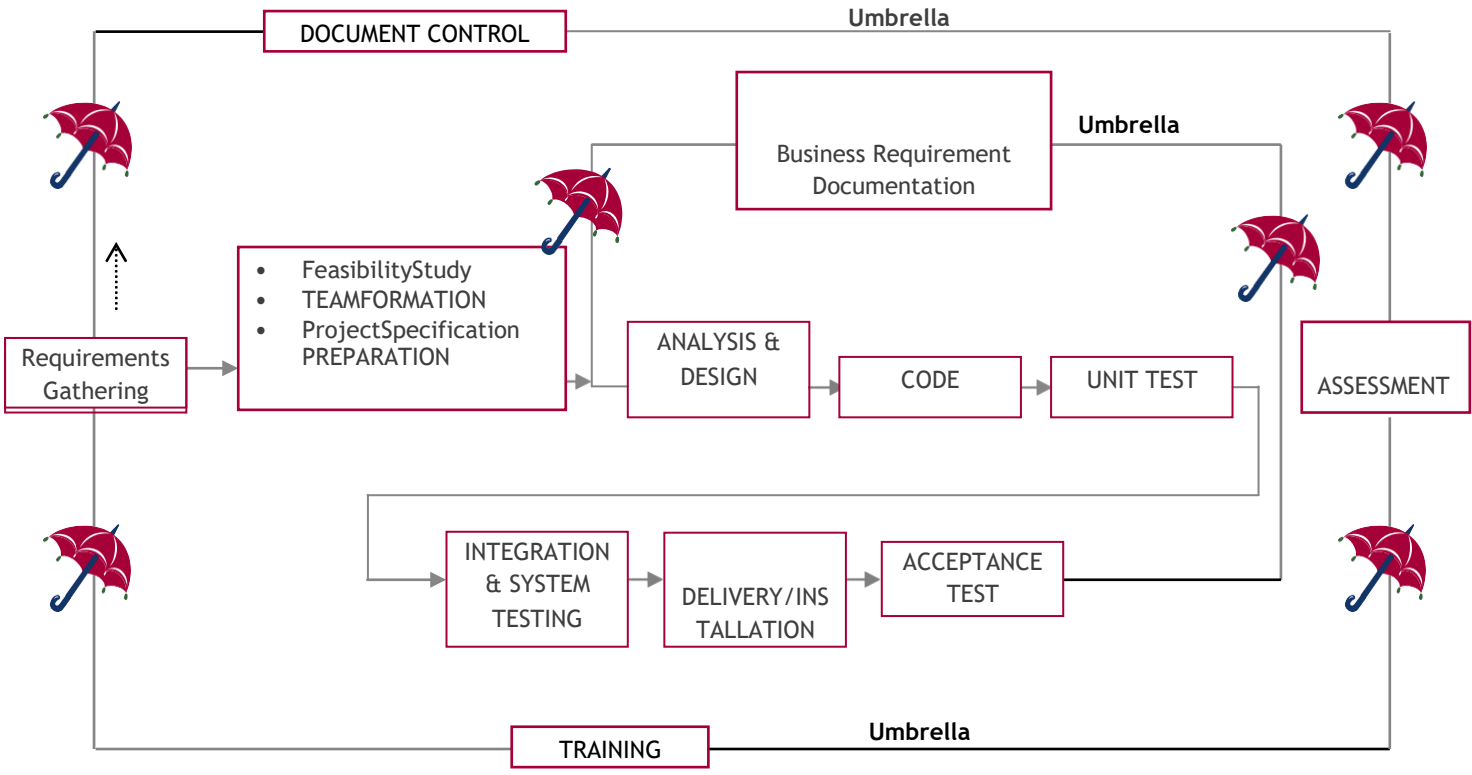


Fig 1: Umberlla Model

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

Stages in SDLC:

- RequirementGathering
- Analysis
- Designing
- Coding
- Testing
- Maintenance

Graph Diagram for Different algorithms:

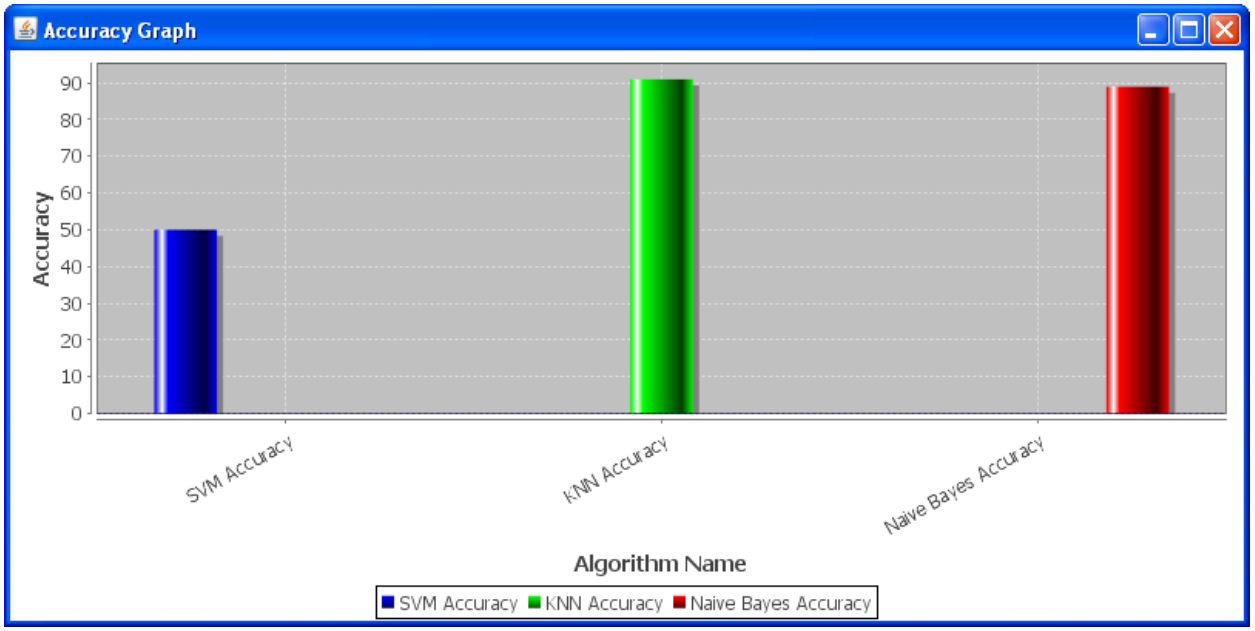


Fig 2.result of different graph

4. DESIGN

The aim of SDN is to provide open interfaces that enable the development of software that can control the connectivity provided by a set of network resources and the flow of network traffic though them, along with possible inspection and modification of traffic that may beperformed in the network. These primitive functions may be abstracted into arbitrary network services, some of which may not be presently apparent.

4.1 SystemArchitecture

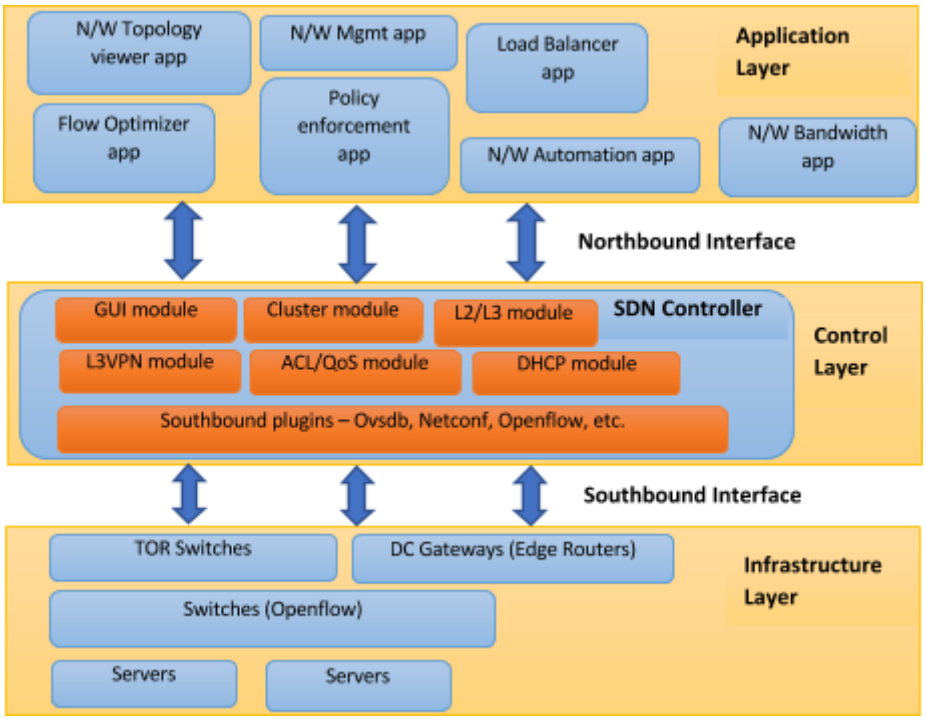


Fig 3:SDN Architecture

SDN broadly consists of 3 layers

- ApplicationLayer
- ControlLayer
- InfrastructureLayer

1.Application layer is open area to develop as much innovative application as possible by leveraging all the network information about network topology, network state, network statistics, etc. There can be several types of applications which can be developed like those related to network automation, network configuration and management, network monitoring, network troubleshooting, network policies and security. Such SDN applications can provide

various end-to-end solutions for real world enterprise and data centrenetworks.

2. **Control Layer** is the land of control plane where intelligent logic in SDN controllers would reside to control network infrastructure. This is the area where every network vendor isworkingtocomeupwiththeirownproductsforSDNcontrollerandframework.Hereinthis layer, a lot of business logic is being written in controller to fetch and maintain different types of network information, state details, topology details, statistics details, and more.

Since SDN controller is for managing networks, so it must have control logic for real world network use-cases like switching, routing, L2 VPN, L3 VPN, firewall security rules, DNS, DHCP, and clustering. Several networking vendors and even open source communities are working on the implementation of these use-cases in their SDN controllers. Once they get implemented, these services expose their APIs (typically REST based) to the upper layer (Application layer), something which makes life easy for network administrators who then use apps on top of SDN controllers to configure, manage and monitor underlying network. Control layer lies in middle and it exposes two types of interfaces – Northbound and Southbound.

Northbound interface: is meant for communication with upper, Application layer and would be in general realized through REST APIs of SDN controllers.

Southbound interface: is meant for communication with lower, Infrastructure layer of network elements and would be in general realized through southbound protocols – Openflow, Netconf, Ovsdb,etc.

The controller communicates with switches by means of OF protocol using the so called southbound interface. Due to security reasons, OF is provided with an optional security attribute that grants the use of TLS on an OF control channel.

This method provides authentication for the switch and the controller (if certificates are appropriately checked on both sides) to stop intruders from impersonating a switch or a controller. Additionally, an encryption of the control channel to hinder eavesdropping is provided. This differs with every implementation because it is not defined.

The controller takes over of all the essential functions like topology management and service. It is able to be improved with additional features; moreover it provides information to other external applications. Currently there is no unique standard for northbound communication. Some efforts have been put to enhance the abstraction level by means of designing network

programming languages on application layer of the SDN.Finally, through the east and westbound interfaces the controller can connect to other controllers .

3. **Infrastructure layer** is composed of various networking equipment which forms underlying network to forward network traffic. It could be a set of network switches and routers in the data centre. This layer would be the physical one over which network virtualization would be laid down through the control layer (where SDN controllers would sit and manage underlying physicalnetwork).

The data plane (sometimes known as the user plane, forwarding plane, carrier plane or bearer plane) is the part of a network that carries user traffic.

The data plane, the control plane and the management plane are the three basic components of a telecommunications architecture.The control plane and management plane serve the data plane, which bears the traffic that the network exists to carry. The data plane enables data transfer to and from clients, handling multiple conversations through multiple protocols, and manages conversations with remote peers. Data plane traffic travels through routers, rather than to or from them.

In conventional networking, all three planes are implemented in the firmware of routers and switches. Software-defined networking (SDN) decouples the data and control planes and implements the control plane in software instead, which enables programmatic access to make network administration much moreflexible.

Moving the control plane to software allows dynamic access and administration. A network administrator can shape traffic from a centralized control console without having to touch individual switches. The administrator can change any network switch's rules when necessary prioritizing, de-prioritizing or even blocking specific types of packets with a very granular level of control.

4.2 SYSTEM FLOW

OpenFlow is a communication protocol that interconnects the forwarding instructions with the data plane. The foundation is based on the SDN paradigm. It checks flow tables are updated on switches and also checks that the controller has a secure protocol for effective communication with its switches.

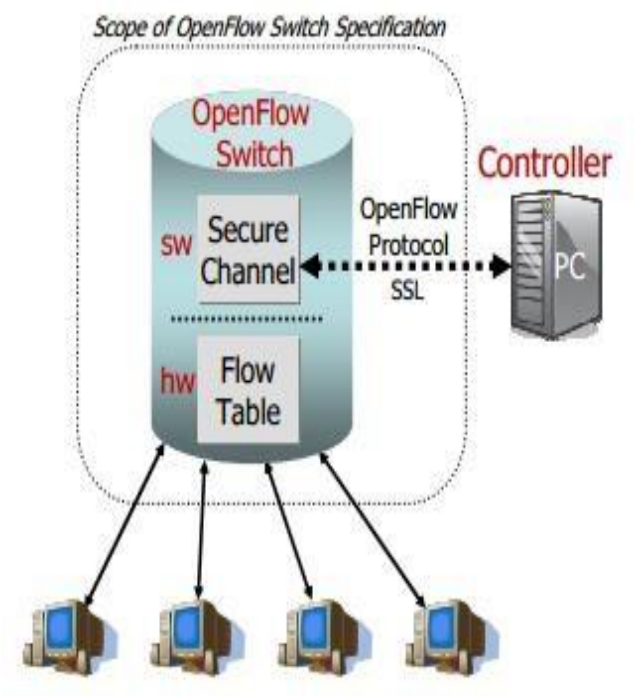


Fig 4 :OpenFlow Switch

The goal of the OF enabled switch is to manage the switch flow tables by receiving controller rules to perform the required updates via the OpenFlow protocol. This protocol gives the ability to the controller to update, delete and insert rules in flow tables. In the following section each of these components are described separately namely the data plane, the control plan and their interaction.

The flow table of each switch includes a set of match fields, counters and instructions. The Ingress port along with the header of each packet that the switch receives is checked against the flow table. If the packet header‟s important parameters such as the Ethernet port number, the source IP port, VLAN tag, destination Ethernet or IP port are found inside the flow table then there is an occurrence of a match.

If there is a match, the switch then runs the specified actions (e.g. drop the packet, block the packet or forward the packet) . If there is no match the controller makes a decision and inserts a new flow table entry. The switch sends a “flow request” message to the controller that includes the new packet header. The controller must verify the received flow request and decide what action should be performed on the packet. After a decision is made it must send the appropriate instructions back to the switch. This new rule indicates the actions that need to be chosen for the packet.

5. IMPLEMENTATION

This contains the algorithm,implementation, and flowchart.

5.1 FlowChart

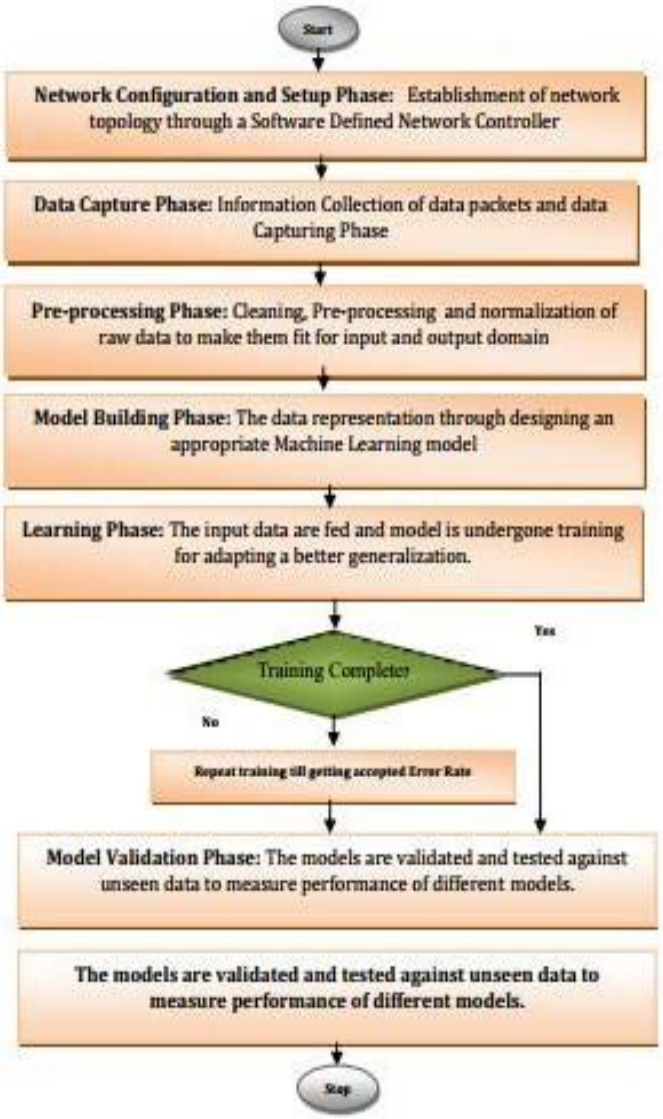


Fig 5 :Proposed Model Flow Chart

5.1.1 NetworkSetup

The network simulation environment for the proposed model is built using Floodlight controller. For building the model the network environment is simulated through a Floodlight Controller. It is a Java based simulation software which also supports eclipse IDE.An

„OpenFlow"networkisneededtobeattachedtothisfordoingcustomrouting;forthis „Mininet"isused.Itisanemulatorwhichcreatesvirtualhosts,switches,controllersand links. For building the model the network environment is simulated through a Floodlight Controller. It is a Java based simulation software which also supports eclipse IDE.An „OpenFlow"networkisneededtobeattachedtothisfordoingcustomrouting;forthis „Mininet"isused. Itisanemulatorwhichcreatesvirtualhosts,switches,controllersandlinks.

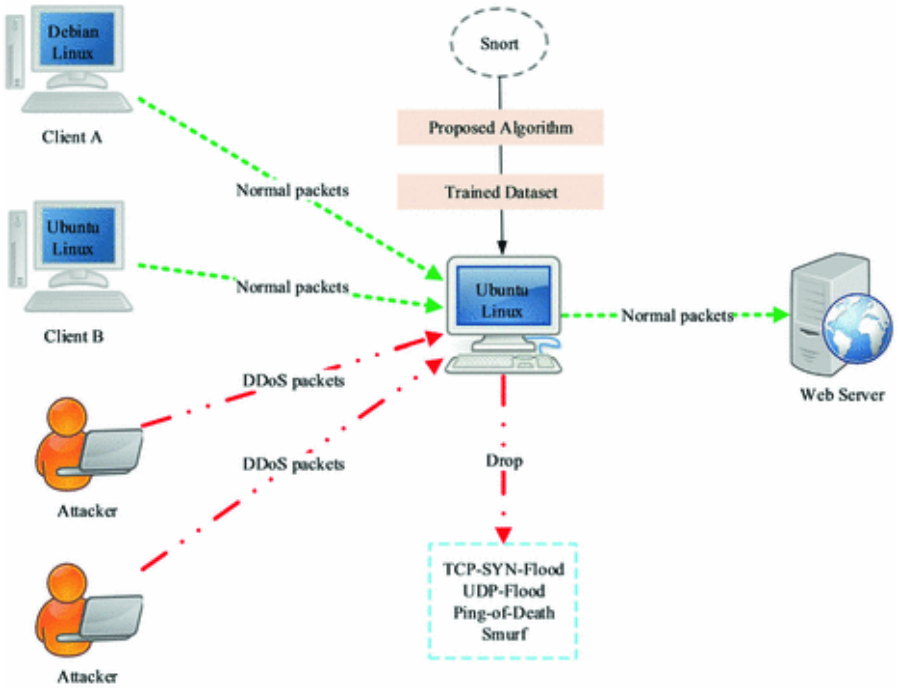


Fig : 6 The network topology of DDOS attack

After doing the initial network setup, the data are extracted using TCP Dump. TCP Dump is an automated tool to monitor network statistics. To simulate the DDOS attack a tool known as Hping3 is being used by which both malicious and non malicious data are collected separately. These raw data areused for building the prediction model using ML algorithms. Three of the supervised algorithms like Support vector machine (SVM) K-Nearest Neighbour (KNN) and Naive Bayes are being used for building a classification model which can classify both type of data. Machine Learning algorithm has been implemented in the controller as a part of Intrusion Detection System (IDS).Weka, a collection of tools for data analysis and predictive modelling is used to compare the performances of different ML algorithms. After doing the training on Weka, the code for testing part is deployed on the

controlleritself.

5.2 ExecutionCode

Main.java

```
package com;

    import javax.swing.JFrame;

    import javax.swing.JPanel;

    import javax.swing.JLabel;

    import javax.swing.JButton;

    import javax.swing.UIManager;

    import java.awt.Font;

    import java.awt.BorderLayout;

    import java.awt.Color;

    import java.awt.Dimension;

    import java.awt.event.ActionListener;

    import java.awt.event.ActionEvent;

    import javax.swing.SwingUtilities;

    import javax.swing.JOptionPane;

    import javax.swing.JTable;

    import javax.swing.JTextArea;

    import javax.swing.JScrollPane;

    import java.io.File;

    import java.util.ArrayList;

    import java.awt.Cursor;

    import javax.swing.JFileChooser;

    import org.jvnet.substance.SubstanceLookAndFeel;

    import java.awt.Cursor;
```

An intelligent software defined network controller for preventing distributed denial of service attack

```
import org.jfree.ui.RefineryUtilities;

public class Main extends JFrame implements Runnable{

        static JTextArea area;

        JScrollPane jsp;

        JPanel p1,p2,p3;

        JLabel l1;

        Font f1,f2;

        JButton b1,b2,b3,b4,b5,b6;

        Thread thread;

        String title = "An Intelligent Software Defined Network Controller For Preventing
Distributed<br/>Denial Of Service Attack";

        File train,test;

        JFileChooser chooser,chooser1;

public Main(){

        super("DDOS Detection Detection");

        p1 = new JPanel();

        p1.setBackground(new Color(255,255,170));

        p1.setPreferredSize(new Dimension(600,80));

        l1                                  =                                  new
JLabel("<HTML><BODY><CENTER>"+title.toUpperCase()+"</center></body></html>");

        l1.setForeground(Color.black);

        l1.setFont(new Font("Times New Roman",Font.BOLD,18));

        p1.add(l1);

chooser = new JFileChooser(new File("dataset"));

        p2 = new JPanel();

        p2.setPreferredSize(new Dimension(420,150));

        f2 = new Font("Times New Roman",Font.BOLD,16);

        b1 = new JButton("Upload DDOS Train Dataset");
```

```
b1.setFont(f2);
p2.add(b1);
b1.setPreferredSize(new Dimension(400,40));
b1.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent
ae){ area.setText("");
                int option = chooser.showOpenDialog(Main.this);
                if(option == chooser.APPROVE_OPTION){
                        train = chooser.getSelectedFile();
                        Cursor hourglassCursor = new Cursor(Cursor.WAIT_CURSOR);
                        setCursor(hourglassCursor);
                        Convert.convert(train.getPath(),"com/train.arff");
                        area.append(train.getName()+" DDOS train dataset loaded\n");
                        Cursor normalCursor = new Cursor(Cursor.DEFAULT_CURSOR);
                        setCursor(normalCursor);
                }
        }
});

b2 = new JButton("Run SVM Algorithm");
b2.setFont(f2);
b2.setPreferredSize(new Dimension(400,40));
p2.add(b2);
b2.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent ae){
                int option = chooser.showOpenDialog(Main.this);
```

```
                    if(option == chooser.APPROVE_OPTION){
                        test = chooser.getSelectedFile();
                        Convert.convert(test.getPath(),"com/test.arff");
                        area.append(test.getName()+" test dataset loaded\n");
                        Cursor hourglassCursor = new Cursor(Cursor.WAIT_CURSOR);
                        setCursor(hourglassCursor);
                        try{
                            SVM.train("com/train.arff");
                            SVM.test("com/test.arff");
                        }catch(Exception e){
                            e.printStackTrace();
                        }
                        Cursor normalCursor = new Cursor(Cursor.DEFAULT_CURSOR);
                        setCursor(normalCursor);
                    }
                }
            });
            b3 = new JButton("Run KNN Algorithm");
            b3.setFont(f2);
            b3.setPreferredSize(new Dimension(400,40));
            p2.add(b3);
            b3.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent ae){
                    int option = chooser.showOpenDialog(Main.this);
                    if(option == chooser.APPROVE_OPTION){
                        test = chooser.getSelectedFile();
```

```
try{
                                        KNN.train("com/train.arff");
                                        KNN.test("com/test.arff");
                                    }catch(Exception e){
                                        e.printStackTrace();
                                    }
                                    Cursor normalCursor = new Cursor(Cursor.DEFAULT_CURSOR);
                                    setCursor(normalCursor);
                                }
                            }
                        });
                        b4 = new JButton("Run Naive Bayes Algorithm");
                        b4.setFont(f2);
                        b4.setPreferredSize(new Dimension(400,40));
                        p2.add(b4);
                        b4.addActionListener(new ActionListener(){
                            public void actionPerformed(ActionEvent ae){
                                int option = chooser.showOpenDialog(Main.this);
                                if(option == chooser.APPROVE_OPTION){
                                    test = chooser.getSelectedFile();
                                    area.append(test.getName()+" test dataset loaded\n");
                                    Cursor hourglassCursor = new Cursor(Cursor.WAIT_CURSOR);
                                    setCursor(hourglassCursor);
                                    try{
                                        NaiveBayesAlg.train("com/train.arff");
                                        NaiveBayesAlg.test("com/test.arff");
```

```
                    }catch(Exception e){
                        e.printStackTrace();
                    }
                    Cursor normalCursor = new Cursor(Cursor.DEFAULT_CURSOR);
                    setCursor(normalCursor);
                }
            }
        });
        b5 = new JButton("Accuracy Graph");
        b5.setFont(f2);
        b5.setPreferredSize(new Dimension(400,40));
        p2.add(b5);
        b5.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ae){
                Chart chart1 = new Chart("Accuracy Graph");
                chart1.pack();
                RefineryUtilities.centerFrameOnScreen(chart1);
                chart1.setVisible(true);
            }
        });
        b6 = new JButton("Exit");
        b6.setFont(f2);
        b6.setPreferredSize(new Dimension(400,40));
        p2.add(b6);
        b6.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ae){
```

```
                    System.exit(0);
                }
        });
        p3 = new JPanel();
        p3.setLayout(new BorderLayout());
        area = new JTextArea();
        area.setFont(f2);
        area.setEditable(false);
        area.setLineWrap(true);
        jsp = new JScrollPane(area);
        jsp.getViewport().setBackground(Color.white);
        p3.add(jsp,BorderLayout.CENTER);
        //p3.setPreferredSize(new Dimension(600,600));
       getContentPane().add(p1,BorderLayout.NORTH);
       getContentPane().add(p2,BorderLayout.SOUTH);
       getContentPane().add(p3,BorderLayout.CENTER);
       thread = new Thread(this);
        thread.start();
}
public void run(){
        while(true){
                try{
                        l1.setVerticalAlignment(JLabel.TOP);
                        thread.sleep(500);
                        l1.setHorizontalAlignment(JLabel.RIGHT);
                        thread.sleep(500);
                        l1.setHorizontalAlignment(JLabel.LEFT);
```

```
                    thread.sleep(500);

                    l1.setHorizontalAlignment(JLabel.CENTER);

                    thread.sleep(500);

                    l1.setVerticalAlignment(JLabel.BOTTOM);

                    thread.sleep(500);

                }catch(Exception e){

                    e.printStackTrace();

                }

        }

}

public static void main(String a[])throws Exception{

        SubstanceLookAndFeel.setCurrentWatermark("org.jvnet.substance.watermark.SubstanceBinaryWatermark");

        SubstanceLookAndFeel.setCurrentTheme("org.jvnet.substance.theme.SubstanceInvertedTheme");

        SubstanceLookAndFeel.setCurrentGradientPainter("org.jvnet.substance.painter.SpecularGradientPainter");

        SubstanceLookAndFeel.setCurrentButtonShaper("org.jvnet.substance.button.ClassicButtonShaper");

        UIManager.setLookAndFeel(new SubstanceLookAndFeel());

        Main main = new Main();

        main.setVisible(true);

        main.setExtendedState(JFrame.MAXIMIZED_BOTH);

}

}
```

NaiveBayes.java

```
package com;

import java.io.FileReader;

import weka.core.Instances;
```

```
import weka.classifiers.bayes.NaiveBayes;
import weka.core.Attribute;
import weka.classifiers.Evaluation;
import javax.swing.JTextArea;
import java.awt.BorderLayout;
import javax.swing.JScrollPane;
import java.io.File;
import java.util.Random;
public class NaiveBayesAlg {
        static double accuracy;
        static NaiveBayes nb;
        static Evaluation eval;
        static int lastIndex;
        static Instances train;
        static double acc;
public static void train(String input) throws Exception {
        nb = new NaiveBayes();
        FileReader reader = new FileReader(input);
   train = new Instances(reader);
   train.setClassIndex(train.numAttributes() - 1);
        lastIndex = train.numAttributes() - 1;
   nb.buildClassifier(train);
        eval = new Evaluation(train);
        eval.crossValidateModel(nb, train,10,new Random(10));
        Main.area.append(eval.toSummaryString("\nResults\n======\n", true)+"\n");
        Main.area.append(eval.toClassDetailsString()+"\n");
        Main.area.append("\n"+eval.toMatrixString()+"\n");
```

```
        String results = nb.toString().trim();
        Main.area.append(results+"\n");
        acc =eval.pctCorrect();
        Main.area.append("Naive Bayes Accuracy : "+acc+"\n\n");
}

public static void test(String input) throws Exception {
        FileReader reader = newFileReader(input);
   Instances test = new Instances(reader);
        test.setClassIndex(test.numAttributes() -1);
        ViewDetection vp = new ViewDetection("Naive Bayes Algorithm");
        for(int i=0; i<test.numInstances(); i++) {
                double index = nb.classifyInstance(test.instance(i));
      String className = train.attribute(lastIndex).value((int)index);
      Object row[] = {test.instance(i).toString(),className};
                vp.dtm.addRow(row);
        }
        vp.setVisible(true);
        vp.setSize(800,600);
}
}
```

6. TESTING

Implementation and Testing:

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testingperiod.

6.1 SystemTesting

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to

move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were alsochecked.

6.2 ModuleTesting

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waitingtime.

6.3 Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

6.4 AcceptanceTesting

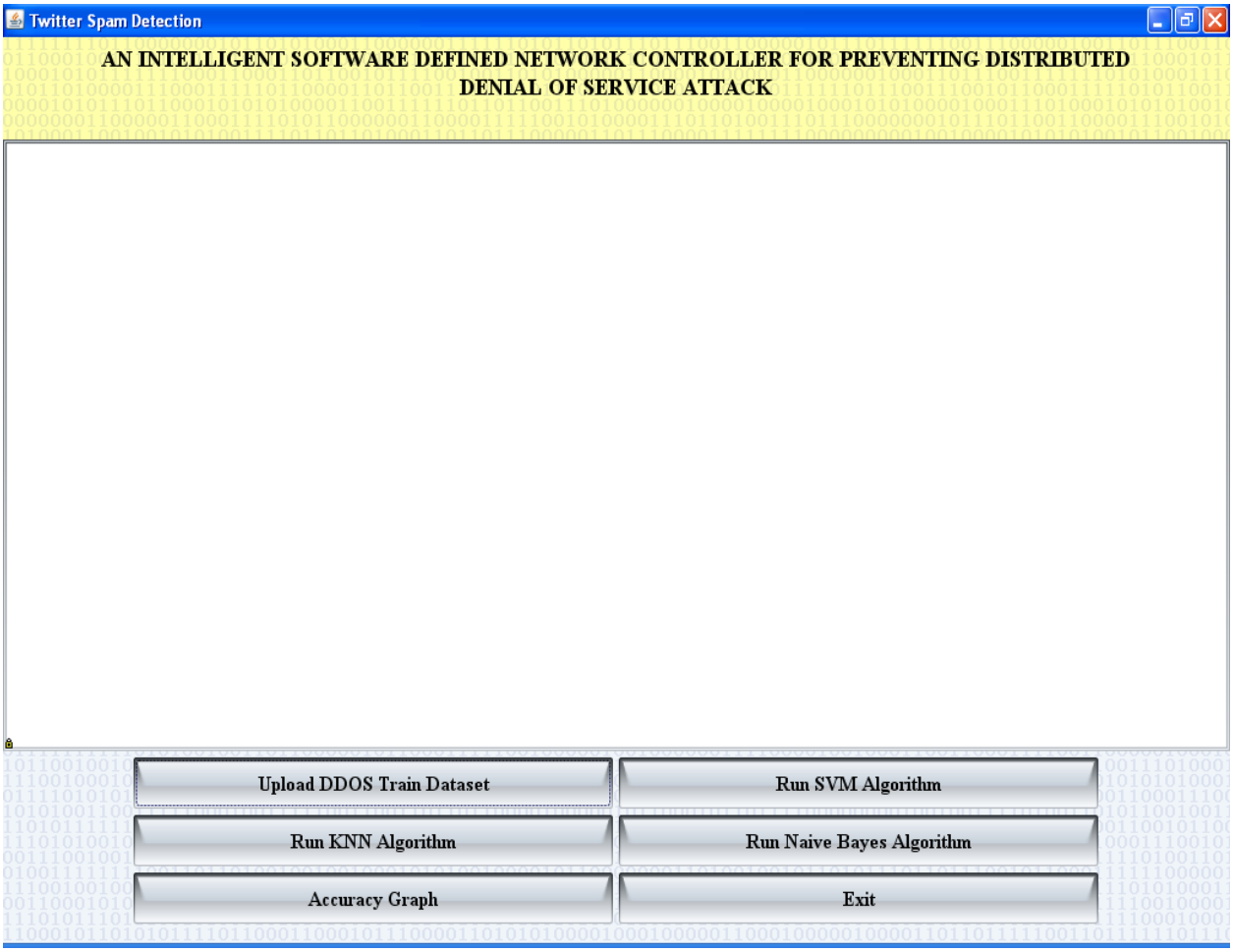
When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation

TEST CASES:

Test Case Id	Test Case Name	Test Case Disc.	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Upload DDOS Train Dataset	Verify the whether dataset is Available or not	If it is not Available	We cannot get details of Train dataset	Train dataset loaded	High	High
02	Run SVM Algorithm	Verify whether the dataset is available or not	If it is not available	We cannot get SVM Algorithm dataset	SVM Algorithm prediction was display	High	High
03	Run KNN Algorithm	Verify whether the dataset is available or not	If it‟s not available	We cannot get KNN Algorithm of dataset	KNN Algorithm was displayed	High	High
04	Run Naives Bayes Algorithm	Verify whether the dataset is available or not	If it is not available	We cannot get Naives Bayes Algorithm dataset	Naives Bayes Algorithm was displayed	High	High
05	Accuracy Graph	Verify Accuracy Graph displaying or not	If it‟s not displaying	We cannot get graph screen	Accuracy Graph was displayed	High	High

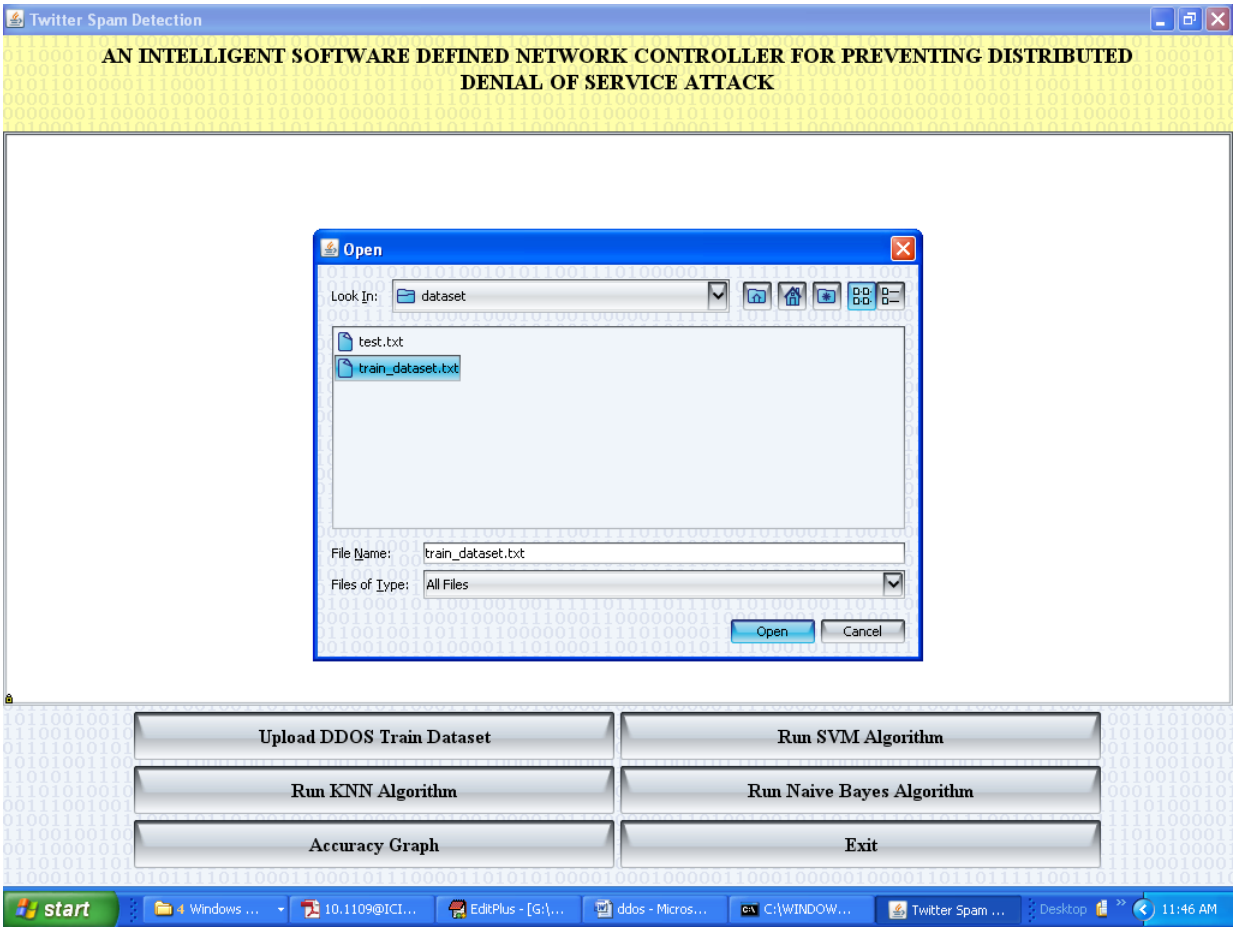
7.RESULTANALYSIS

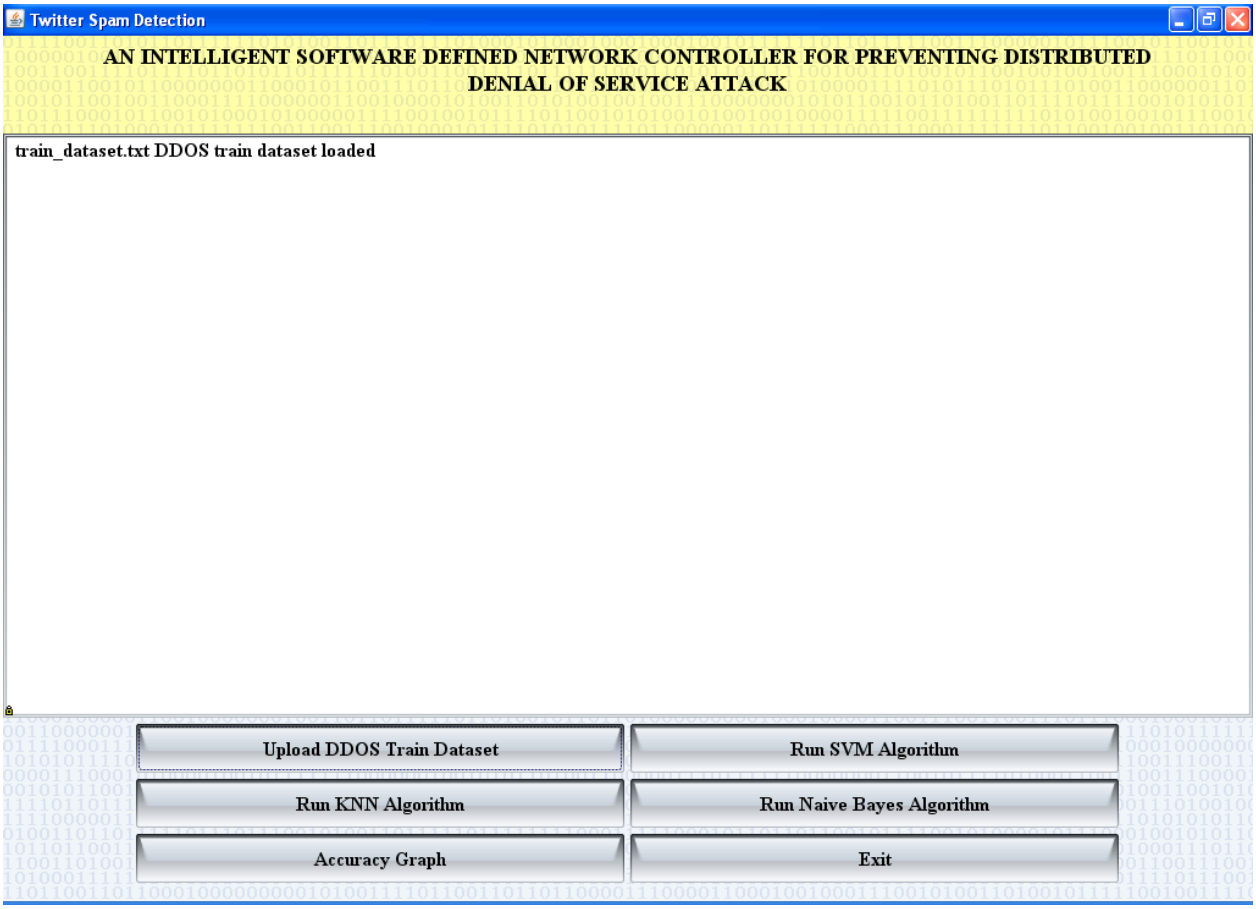
7.1 Screen Shots



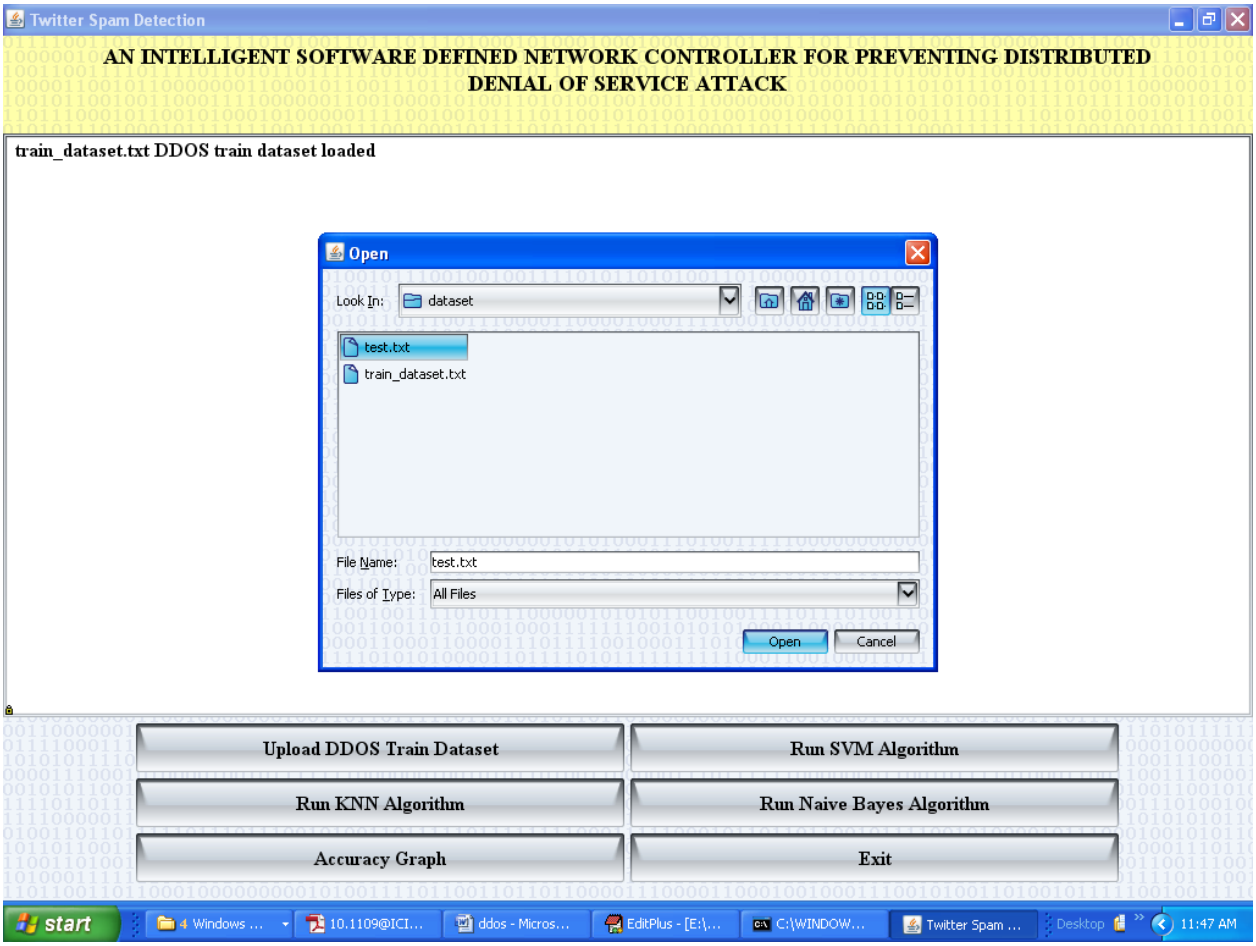
Click on „Upload DDOS Train Dataset“ button and upload train dataset

An intelligent software defined network controller for preventing distributed denial of service attack

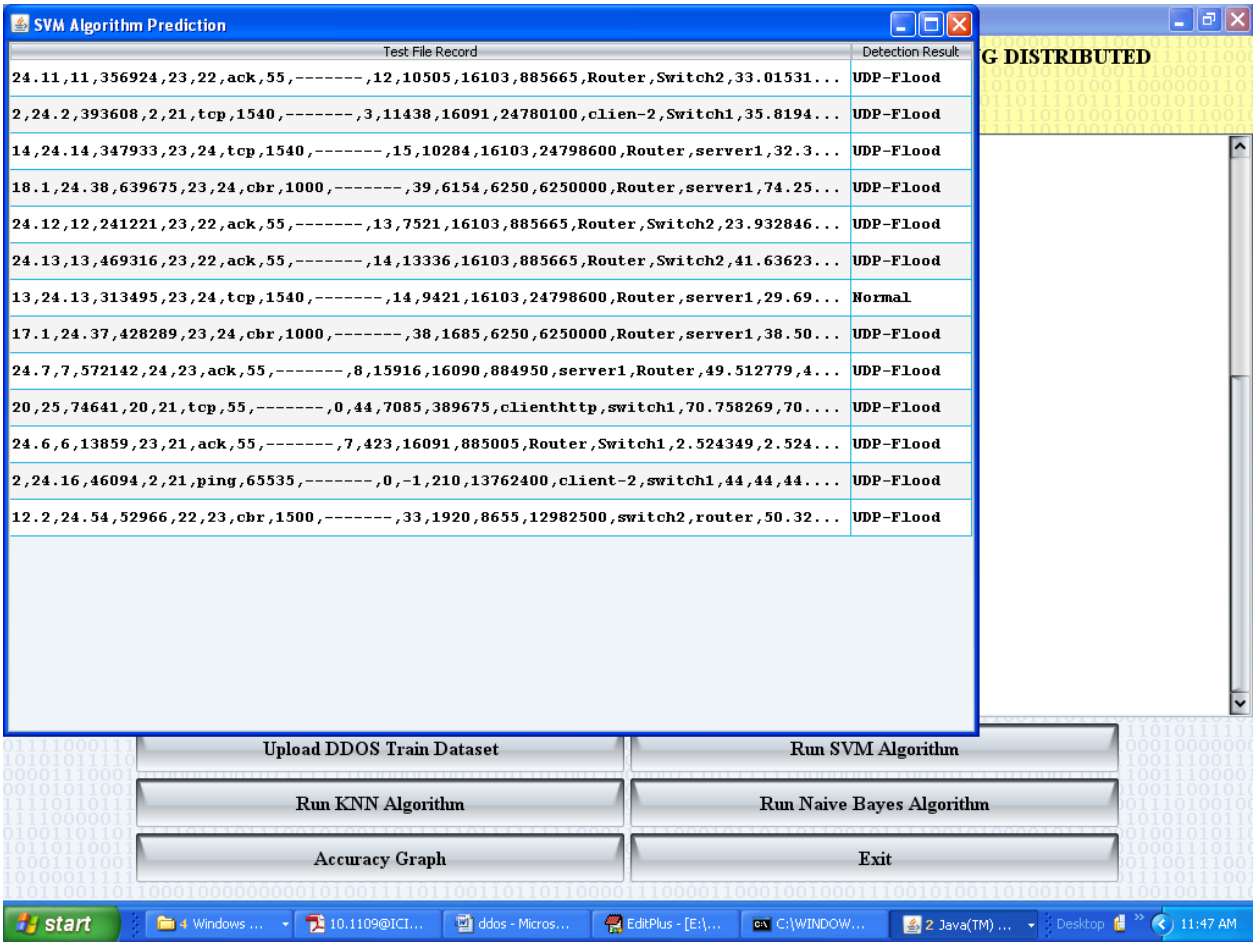




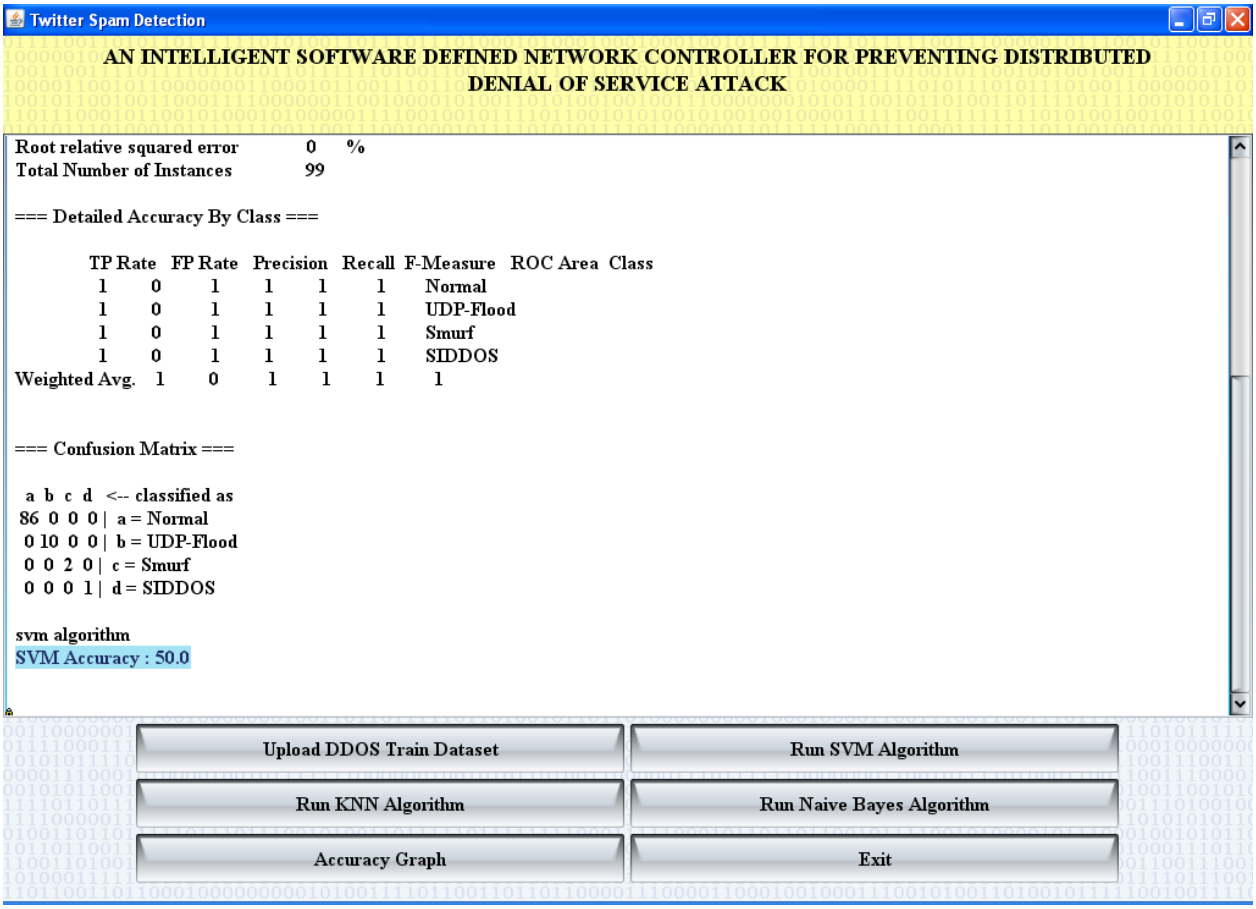
Now click on „Run SVM Algorithm“ button and upload test file



Below is the prediction for test dataset

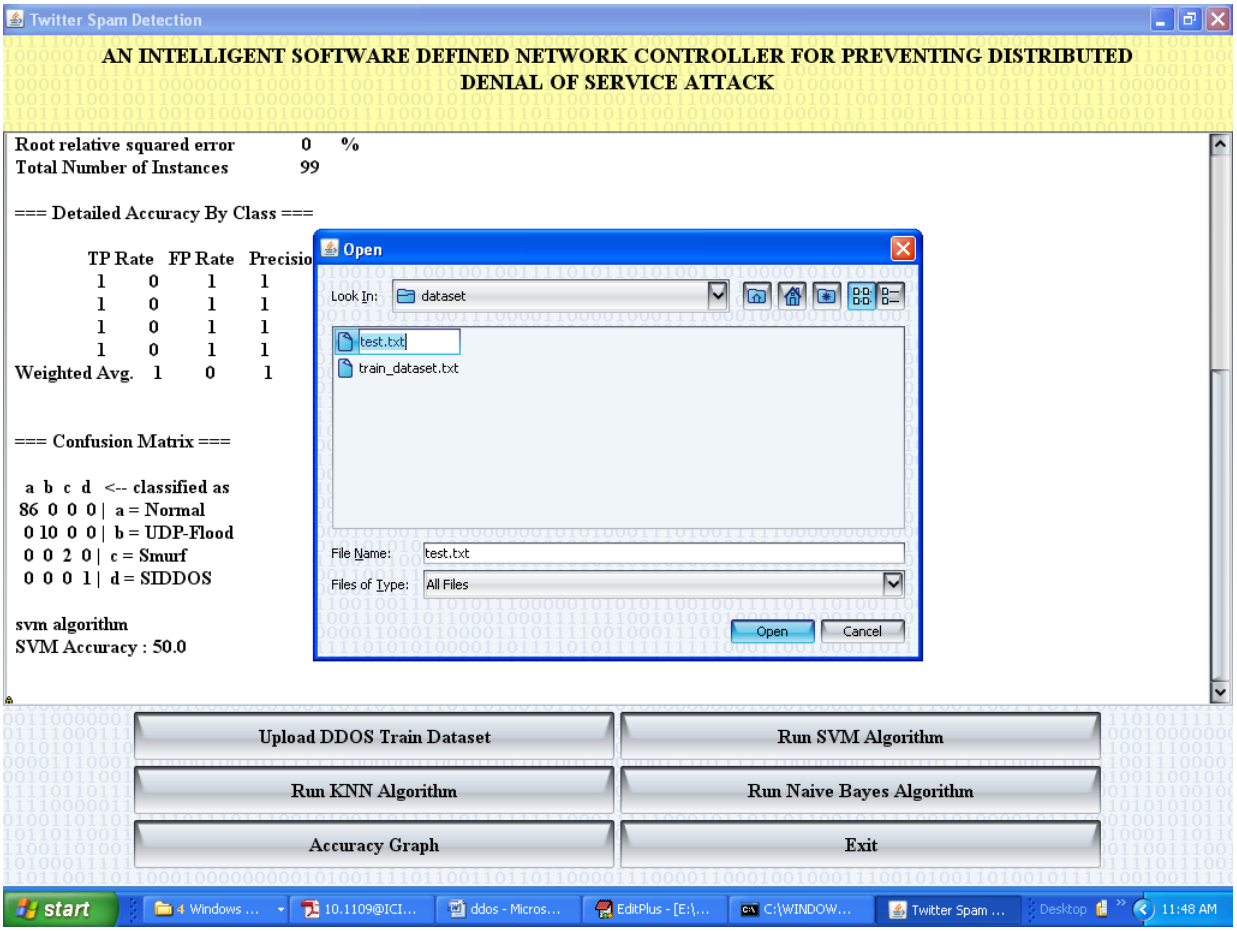


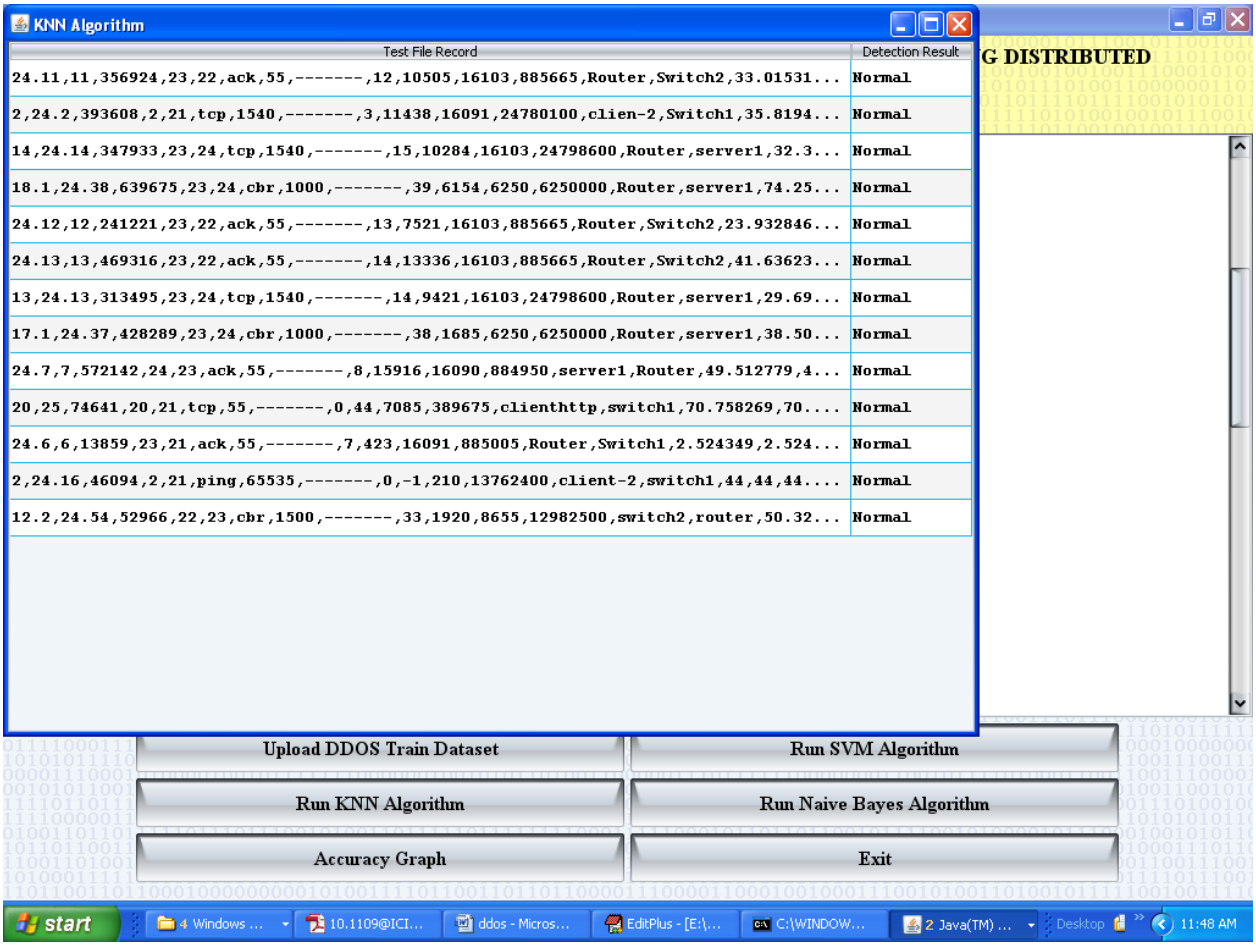
Below is SVM accuracy


Twitter Spam Detection
AN INTELLIGENT SOFTWARE DEFINED NETWORK CONTROLLER FOR PREVENTING DISTRIBUTED DENIAL OF SERVICE ATTACK
Root relative squared error 0 %
Total Number of Instances 99
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 Normal
1 0 1 1 1 1 UDP-Flood
1 0 1 1 1 1 Smurf
1 0 1 1 1 1 SIDDOS
Weighted Avg. 1 0 1 1 1 1
=== Confusion Matrix ===
a b c d <-- classified as
86 0 0 0 | a = Normal
0 10 0 0 | b = UDP-Flood
0 0 2 0 | c = Smurf
0 0 0 1 | d = SIDDOS
svm algorithm
SVM Accuracy : 50.0
Upload DDOS Train Dataset
Run SVM Algorithm
Run KNN Algorithm
Run Naive Bayes Algorithm
Accuracy Graph
Exit

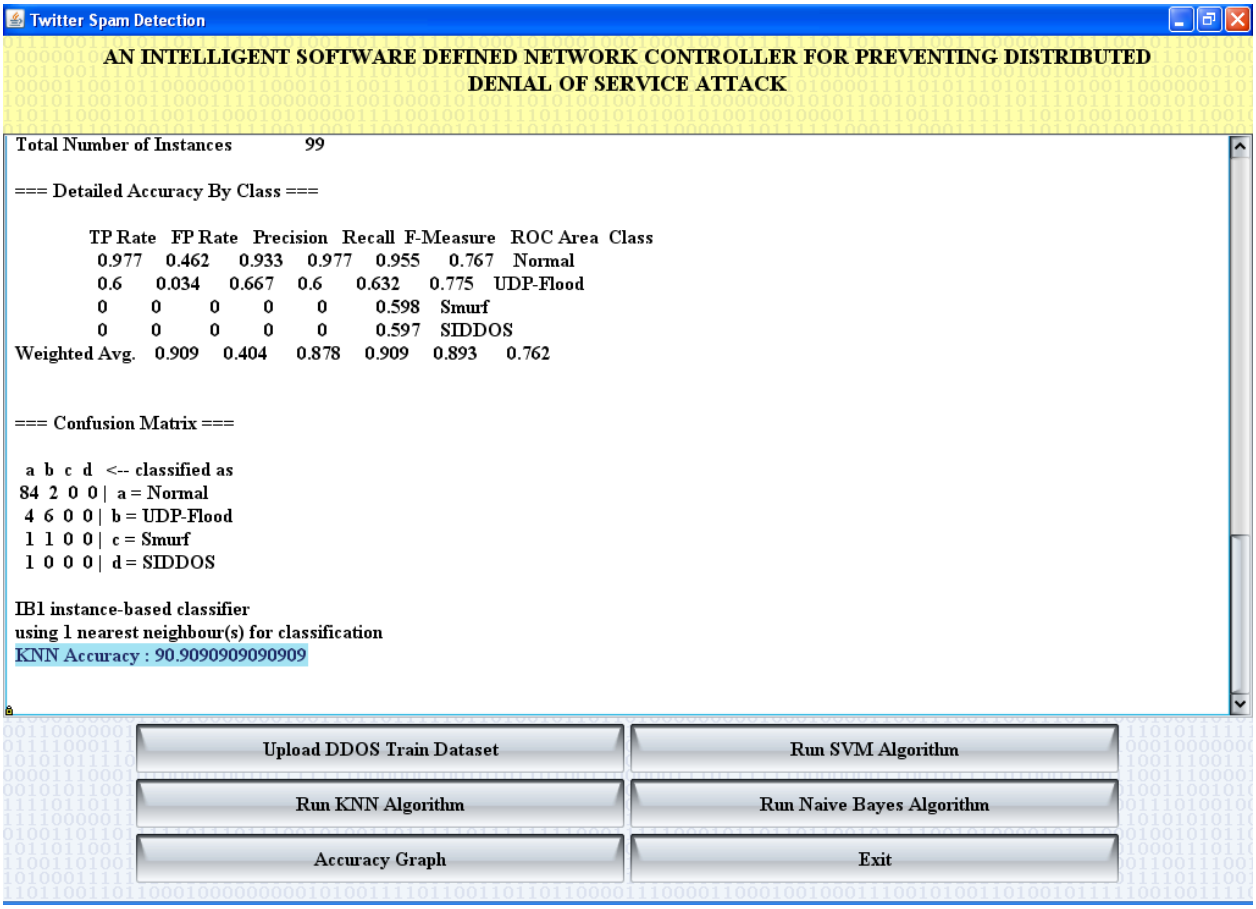
Now run KNN

An intelligent software defined network controller for preventing distributed denial of service attack

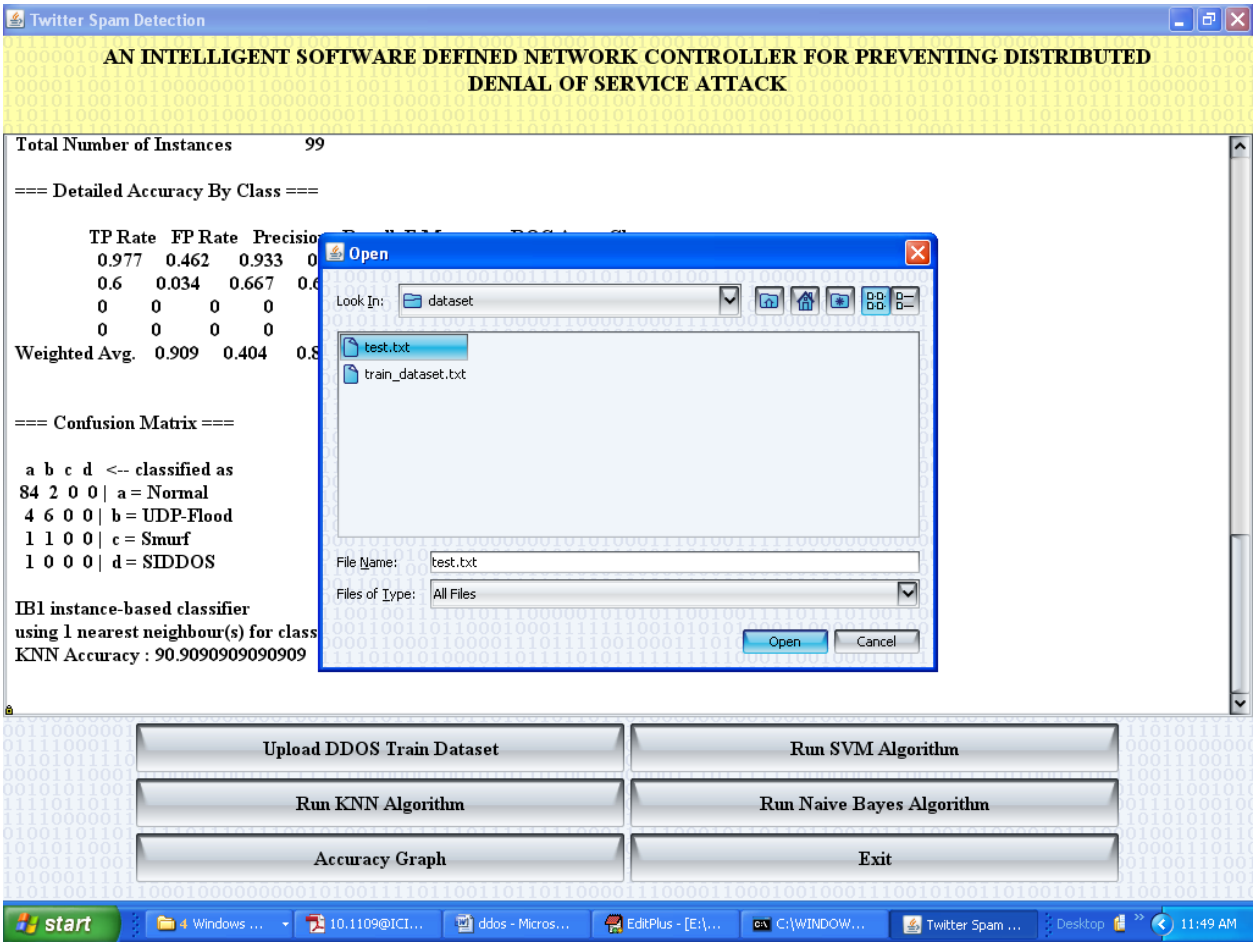




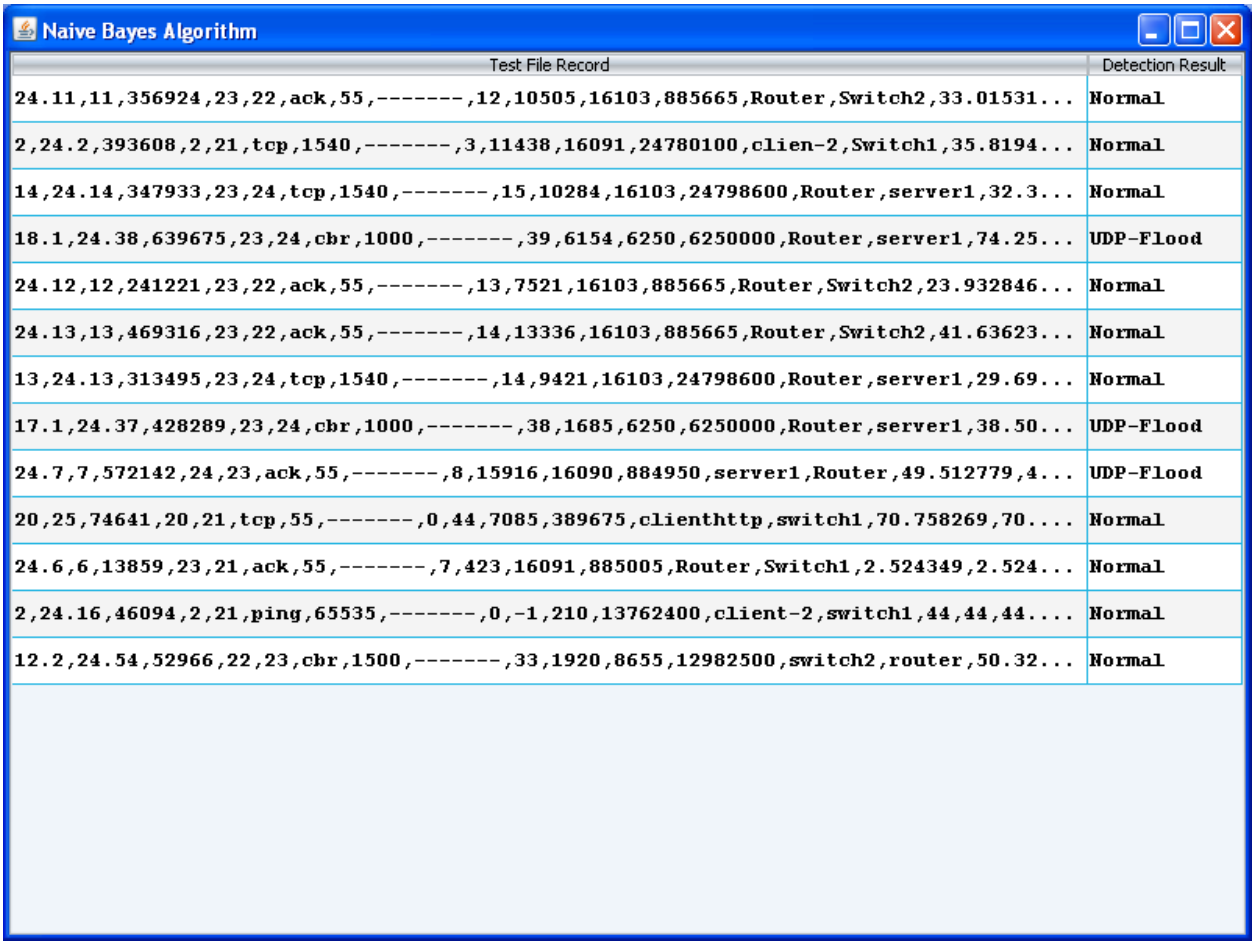
Below is KNN Accuracy



Now run naiye bayes

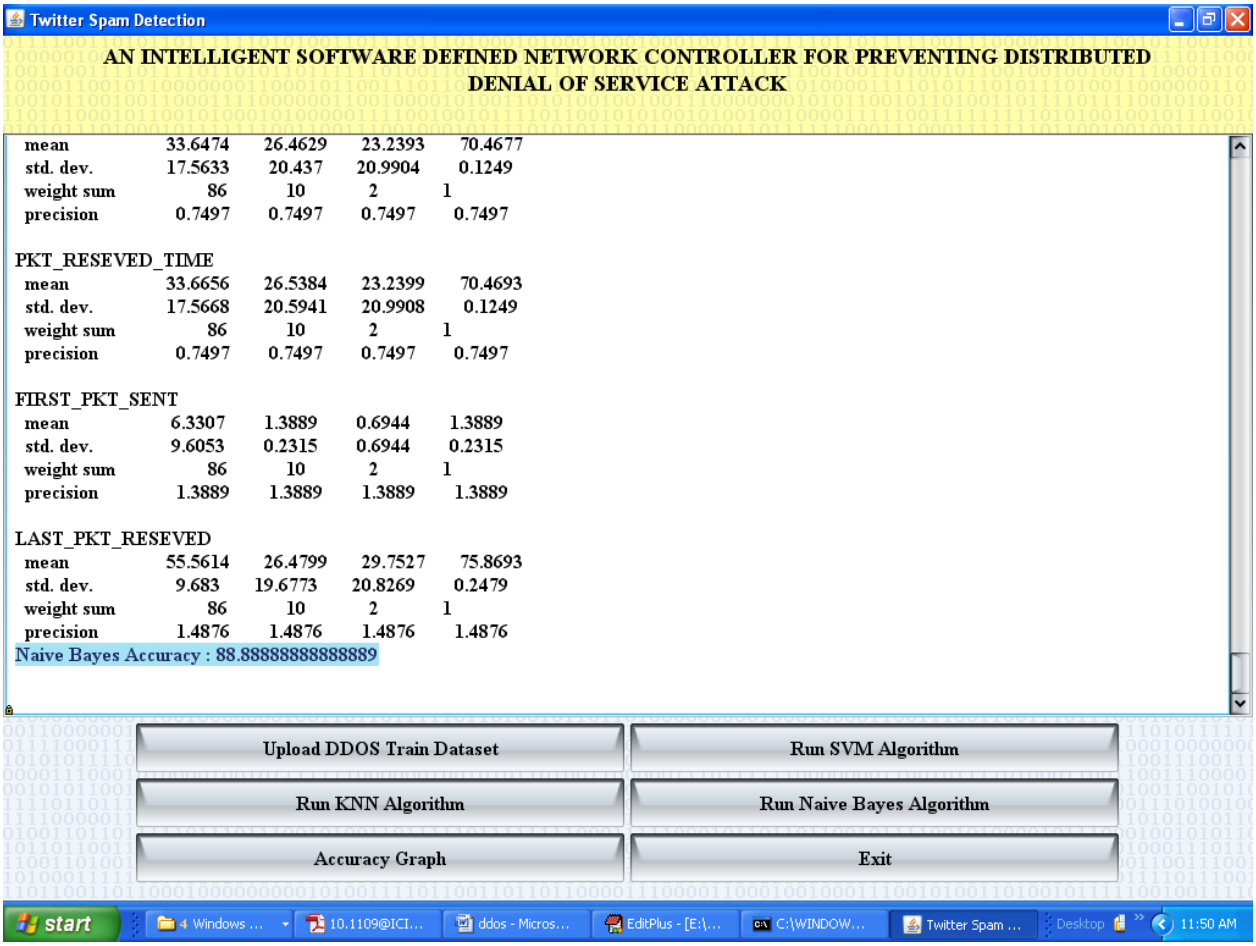


Below is naiye bayes prediction

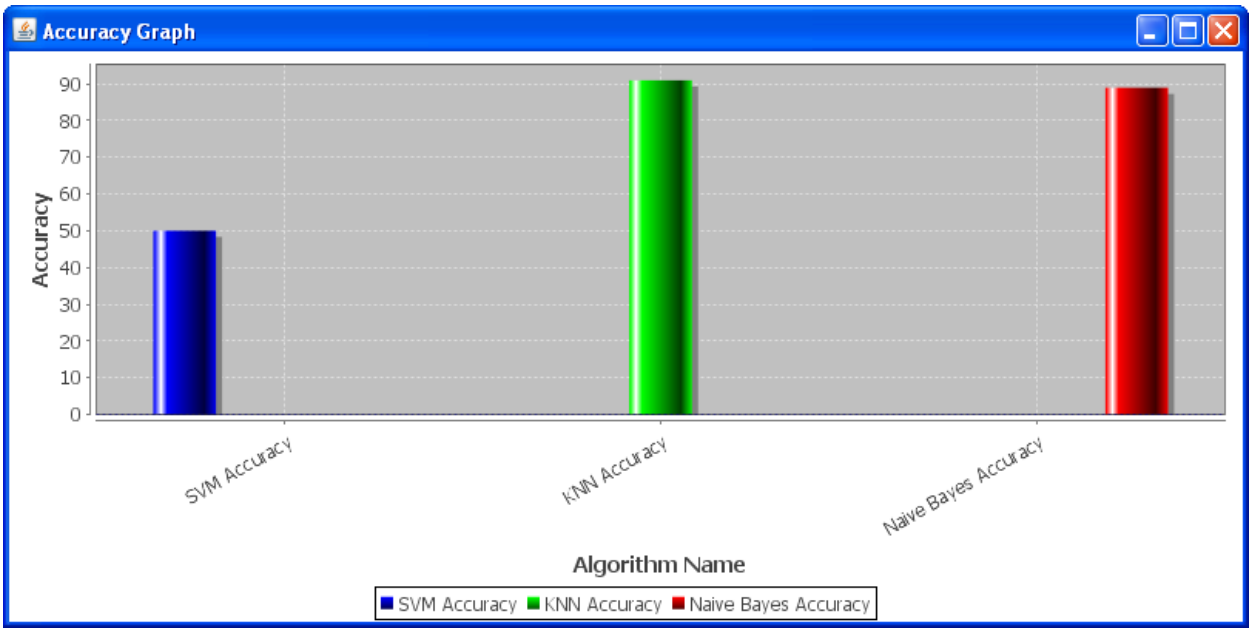


Below is naïve bayes accuracy

An intelligent software defined network controller for preventing distributed denial of service attack



Click on Accuracy graph button



In above graph x-axis represents algorithm name and y-axis represents accuracy

8.CONCLUSION

The current work is evolving around using a SDN controller to detect DDOS attack. After simulating the network, a classification model is built by using three supervised ML algorithms. The supervised algorithms which are used for this work are KNN, Naive Bayes and SVM. A tool is being used to do the DDOS attack in SDN environment. The data are collected in two groups. From which one set is taken from normal case and the other set is for the infected one. The features of data which are considered as inputs to the learning models are number of Packets, Protocol, Delay, Bandwidth, Source IP and Destination IP. Basing on these attributes the learning algorithms are trained on 75% of the total data set and the rest is used as testing. Out of three algorithms, KNN is giving the best performance. According to free lunch theorem of machine learning, no machine learning algorithm works for all setting. Hence, the limitations of this work are that, it has only tested on three algorithms. In future it can be used for testing in some more ML algorithms, which may give better results in terms of training time. Steps could also been taken to make an early notifications to the victim users and machines after identifying the infectedpackets.

9.REFERENCES

[1] Open Networking Foundation, Jun. 2014. [Online]. Available: https://www.opennetworking.org/

[2] Sezer, Sakir, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. "Are we ready for SDN? Implementation challenges for software-defined networks." IEEE Communications Magazine 51, no. 7 (2013):36-43.

[3] Jarraya, Yosr, Taous Madi, and Mourad Debbabi. "A survey and a layered taxonomy of software-defined networking." IEEE Communications Surveys & Tutorials 16, no. 4 (2014): 1955-1980.

[4] Nunes, Bruno Astuto A., Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A survey of softwaredefined networking: Past, present, and future of programmable networks." IEEE Communications Surveys & Tutorials 16, no. 3 (2014): 1617-1634.

[5] List of Openflow Software Projects, Apr. 2013. [Online]. Available:http://yuba.stanford.edu/casado/of-sw.html

[6] Yan, Qiao, F. Richard Yu, Qingxiang Gong, and Jianqiang Li. "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges." IEEE Communications Surveys & Tutorials 18, no. 1 (2016):602-622.

[7] Shin, Seungwon, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. "Avant-guard: Scalable and vigilant switch flow management in software-defined networks." In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 413-424. ACM, 2013.

[8] Wang, Bing, Yao Zheng, Wenjing Lou, and Y. Thomas Hou. "DDoS attack protection in the era of cloud computing and software-defined networking." Computer Networks 81 (2015):308-319.