A

PROJECT REPORT

ON

Semi-Supervised Machine Learning Approach For DDOS Detection

Submitted in partial fulfilment of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

M. Sai Darshan Balaji	**[18BH1A05F8]**
M. Jaya Krishna Sai	**[18BH1A05F9]**
T. Amarnath Goud	**[18BH1A05E2]**
Md Ali Ahmad Khurshid	**[18BH1A05J7]**

Under the guidance of

Dr. S. Joy Kumar, PhD

CSE DEPARTMENT

ST. MARY'S ENGINEERING COLLEGE

(AFFILIATED TO JNTUH, APPROVED BY AICTE, ACCREDITED BY NAAC)

NEAR RAMOJIFILM CITY, DESHMUKHI (V), YADADRI, BHUVANGIRI DIST-508284

[2018-2022]

DECLARATION

A project titled, **“SEMI-SUPERVISED MACHINE LEARNING APPROACH FOR DDOS DETECTION”** submitted to the Department of **Computer Science and Engineering**, **St. Mary’s Engineering College** in fulfilment of degree for the award of Bachelor of technology, is a bonafide work done by us. No part of this report is copied from Internet and wherever the portion is taken; the same has been duly referred in the text. The reported results are based on the project work entirely done by us and not copied from any other sources. Also, we declare that the matter embedded in this report has not been submitted by us in full or partially therefore the award of any degree of any other institution or university previously.

By

M. Sai Darshan Balaji	**[18BH1A05F8]**
M. Jaya Krishna Sai	**[18BH1A05F9]**
T. Amarnath Goud	**[18BH1A05E2]**
Md. Ali Ahmad Khurshid	**[18BH1A05J7]**

ST. MARY'S ENGINEERING COLLEGE

(AFFILIATED TO JNTUH, APPROVED BY AICTE, ACCREDITED BY NAAC)

NEAR RAMOJIFILM CITY, DESHMUKHI(V), YADADRI, BHUVANGIRI DIST-508284

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report as "**SEMI-SUPERVISED MACHINE LEARNING APPROACH FOR DDOS DETECTION** " has carried out and being submitted by **M. Sai Darshan Balaji (18BH1A05F8) M. Jaya Krishna Sai (18BH1A05F9) T. Amarnath Goud (18BH1A05E2) Md. Ali Ahmad Khurshid (18BH1A05J7)** have done this project in our institution for the partial fulfilment of award of Degree of in "**BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING"** in the department of **"COMPUTER SCIENCE AND ENGINEERING"** to the **Jawaharlal Nehru Technological University**, Hyderabad is a record of bonafide work carried out under our guidance and supervision.

Internal Guide	**HEAD OF THE DEPATMENT**
Dr. S. Joy Kumar, **Ph.D**	**K. HARISH KUMAR** M.TECH.,[Ph.D]
Asst. Professor	Asst. Professor
Department of CSE	Department of CSE

External Examiner

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible. We consider it as a privilege to express our gratitude and respect to all those who guided us in the completion of the project.

We are thankful to our internal guide**, Dr. S. JOY KUMAR., Assistant professor, Department of Computer Science and Engineering, St. Mary's Engineering College** for having been of a source encouragement and for insisting us to do this project work.

We are obliged to **Mr. K. Harish Kumar, Assistant professor, Head of the Department of Computer Science and Engineering, St. Mary's Engineering College** for his guidance and suggestion throughout project work.

We take this opportunity to express a deep sense of gratitude to **Sri Dr. T G ARUL, Principal of St. Mary's Engineering College** for allowing us to do this seminar and for this affectionate encouragement in presenting this project work.

We convey our sincere thanks to **Sri Dr. Rev. K.V.K RAO, Chairman** of **St. Mary's Engineering College** for giving us learning environment to grow out self personally as well as professionally.

We would like to express our thanks to all staff members who have helped us directly and indirectly in accomplishing this project work. We also extended our sincere thanks to our parents and friends for their moral support throughout the project work. Above all we thank God almighty for his manifold mercies in carrying out this project work successfully.

M. Sai Darshan Balaji	**[18BH1A05F8]**
M. Jaya Krishna Sai	**[18BH1A05F9]**
T. Amarnath Goud	**[18BH1A05E2]**
Md. Ali Ahmad Khurshid	**[18BH1A05J7]**

ABSTRACT

The appearance of malicious apps is a serious threat to the Android platform. Most types of network interfaces based on the integrated functions, steal users' personal information and start the attack operations. In this paper, we propose an effective and automatic malware detection method using the text semantics of network traffic. In particular, we consider each HTTP flow generated by mobile apps as a text document, which can be processed by natural language processing to extract text-level features. Later, the use of network traffic is used to create a useful malware detection model. We examine the traffic flow header using N-gram method from the natural language processing (NLP). Then, we propose an automatic feature selection algorithm based on chi-square test to identify meaningful features. It is used to determine whether there is a significant association between the two variables. We propose a novel solution to perform malware detection using NLP methods by treating mobile traffic as documents. We apply an automatic feature selection algorithm based on N-gram sequence to obtain meaningful features from the semantics of traffic flows. Our methods reveal some malware that can prevent detection of antiviral scanners. In addition, we design a detection system to drive traffic to your own-institutional enterprise network, home network, and 3G / 4G mobile network. Integrating the system connected to the computer to find suspicious network behaviours.

Index Terms—Malware detection, HTTP flow analysis, text semantics, machine learning.

INDEX

FIGURE INDEX

LIST OF PLATES

1. INTRODUCTION

Despite the important evolution of the information security technologies in recent years, the DDoS attack remains a major threat of Internet. The attack aims mainly to deprive legitimate users from Internet resources. The impact of the attack relies on the speed and the amount of the network traffic sent to the victim. Generally, there exist two categories of the DDoS attack namely Direct DDoS attack and Reflection-based DDoS directly the victim host with a large number of network packets. Whereas, in the Reflection based DDoS attack the attacker uses the zombie hosts to take control over a set of compromised hosts called Reflectors. The latter are used to forward a massive amount of attack traffic to the victim host. Recently, destructive DDoS attacks have brought down more than 70 vital services of Internet including Github, Twitter, Amazon, Paypal, etc . Attackers have taken advantages of Cloud Computing and Internet of Things technologies to generate a huge amount of attack traffic; more than 665 Gb/s. Analyzing this amount of network traffic at once is inefficient, computationally costly and often leads the intrusion detection systems to fall.

Data mining techniques have been used to develop sophisticated intrusion detection systems for the last two decades. Artificial Intelligence, Machine Learning (ML), Pattern Recognition, Statistics, Information Theory are the most used data mining techniques for intrusion detection . Application process of data mining techniques in general and ML techniques more specifically requires five typical steps selection, pre-processing, transformation, mining, and interpretation. Despite that pre-processing and transformation steps may be trivial for intrusion detection applications, selection, mining and interpretation steps are crucial for selecting relevant data, filtering noisy data and detecting intrusions . These three crucial steps are the most challenging of the existing data mining based intrusion detection approaches.

The existing Machine Learning based DDoS detection approaches can be divided into three categories. Supervised ML approaches that use generated labelled network traffic datasets to build the detection model. Two major issues are facing the supervised approaches. First, the generation of labelled network traffic datasets is costly in terms of computation and time. Without a continuous update of their detection models, the supervised machine learning approaches are unable to predict the new legitimate and attack behaviours. Second, the presence of large amount of irrelevant normal data in the incoming network traffic is noisy and reduces the performances of supervised ML classifiers. Unlike the first category, in the unsupervised approaches no labelled dataset is needed to build the detection model. The DDoS and the normal traffics are distinguished based on the analysis of their underlying distribution characteristics. However, the main drawback of the unsupervised approaches is the high false positive rates. In the high dimensional network traffic data the distance between points becomes meaningless and tends to homogenize. This problem, known as ‘the curse of dimensionality’, prevents unsupervised approaches to accurately detect attacks . The semi-supervised ML approaches are taking advantages of both supervised and unsupervised approaches by the ability to work on labelled and unlabeled datasets. Also, the combination of supervised and unsupervised approaches allows increasing accuracy and decreases the false positive rates. However, semi-supervised approaches are also challenged by the drawbacks of both approaches. Hence, the semi-supervised approaches require a sophisticated implementation of its components in order to overcome the drawbacks of supervised and unsupervised approaches. In this paper we present an online sequential semi supervised ML approach for DDoS detection. A time based sliding window algorithm is used to estimate the entropy of the network header features of the incoming network traffic. When the entropy exceeds its normal range, the unsupervised co-clustering algorithm splits the incoming network traffic into three clusters. Then an information gain ratio is computed based on the average entropy of the network header features between the network traffic subset of the current time window and each one of the obtained clusters. The network traffic data clusters that produce high information gain ratio are considered as anomalous and they are selected for pre-processing and classification using an ensemble classifiers based on the Extra-Trees algorithm.

Our approach constitutes of two main parts unsupervised and supervised. The unsupervised part includes entropy estimation, co-clustering and information gain ratio. The supervised part is the Extra-Trees ensemble classifiers. The unsupervised part of our approach allows to reduce their relevant and noisy normal traffic data, hence reducing false positive rates and increasing accuracy of the supervised part. Whereas, the supervised part issued to reduce the false positive rates of the unsupervised part and to accurately classify the DDoS traffic. To better evaluate the performance of the proposed approach three public network traffic datasets are used in the experiment, namely the NSL-KDD, theUNBISCXIDS2012dataset[13]andtheUNSW-NB15. The experimental results are satisfactory when compared with the state-of-the-art DDoS detection methods. The main contributions of this paper can be summarized as follows:

- Presenting an unsupervised and time sliding window algorithm for detecting anomalous traffic data based on co-clustering, entropy estimation and information gain ratio. This algorithm allows reducing drastically the amount of network traffic to pre-process and to classify, resulting in a significant improvement of the performance of the proposed approach.
- Adopting supervised ensemble ML classifiers based on the Extra-Trees algorithm to accurately classify the anomalous traffic and to reduce the false positive rates.
- Combining both previous algorithms in a sophisticated semi-supervised approach for DDoS detection. This allows achieving good DDoS detection performance compared to the state-of-the-art DDoS detection methods.
- The unsupervised part of our approach allows reducing the irrelevant and noisy normal traffic data, hence reducing false positive rates and increasing accuracy of the supervised part. Whereas, the supervised part allows reducing the false positive rates of the unsupervised part and to accurately classify the DDoS traffic. This paper is organized as follows. Section 2 highlights state-of-the-art DdoS detection methods. Section 3 presents a brief of the benchmark datasets used in this paper. The proposed approach is detailed in Section 4. Section 5 describes the conducted experiments and the performance metrics used to evaluate the proposed approach. The obtained results, the results discussion and the conducted comparisons are given in Section6. Finally, Section 7 draws the conclusion and outlines future works.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

The first phase of their approach consists of dividing the incoming network traffic into three types of protocols TCP, UDP or Other. Then classifying it into normal or anomaly traffic. In the second stage a multi-class algorithm classify the anomaly detected in the first phase to identify the attacks class in order to choose the appropriate intervention. Two public datasets are used for experiments in this paper namely the UNSW-NB15 and the NSL-KDD several approaches have been proposed for detecting DDoS attack. Information theory and machine learning are the performances of network intrusion detection approaches, in general, rely on the distribution characteristics of the underlaying network traffic data used for assessment. The DDoS detection approaches in the literature are under two main categories unsupervised approaches and supervised approaches. Depending on the benchmark datasets used, unsupervised approaches often suffer from high false positive rate and supervised approach cannot handle large amount of network traffic data and their performances are often limited by noisy and irrelevant network data. Therefore, the need of combining both, supervised and unsupervised approaches arises to overcome DDoS detection issues.

DISADVANTAGES:

- The datasets above are split into train subsets and test subsets using a configuration of 60% and 40% respectively. The train subsets are used to fit the Extra-Trees ensemble classifiers and the test subsets are used to test the entire proposed approach. Before fitting the classifiers the train subsets are normalized using the MinMa*x* method

- This section presents the details of the proposed approach and the methodology followed for detecting the DDoS attack. The proposed approach consists of five major steps: Datasets pre-processing, estimation of network traffic Entropy, online co-clustering, information gain ratio

- The aim of splitting the anomalous network traffic is to reduce the amount of data to be classified by excluding the normal cluster for the classification. For DDoS detection normal traffic records are irrelevant and noisy as the normal behaviours continue to evolve. Most of the time the new unseen normal traffic instances cause the increase of the false positive rate and the decrease of the classification accuracy. Hence, excluding some noisy normal instances of the network traffic data for classification is beneficial in terms of low false positive rates and classification accuracy. Assuming that after the network traffic clustering one cluster contains only normal traffic, a second one contains only DDoS traffic and a third one contains both DDoS and normal traffic.

2.2 PROPOSED SYSTEM

This section introduces our methodology to detect the DDoS attack. The five-fold steps application process of data mining techniques in network systems discussed in characterizes the followed methodology. The main aim of combining algorithms used in the proposed approach is to reduce noisy and irrelevant network traffic data before pre-processing and classification stages for DDoS detection while maintaining high performance in terms of accuracy, false positive rate and running time, and low resources usage. Our approach starts with estimating the entropy of the FSD features over a time-based sliding window. When the average entropy of a time window exceeds its lower or upper thresholds the co-clustering algorithm split the received network traffic into three clusters. Entropy estimation over time sliding windows allows
To detect abrupt changes in the incoming network traffic distribution which are often caused by DDoS attacks. Incoming network traffic within the time windows having abnormal entropy values is suspected to contain DDoS traffic. The focus only on the suspected time windows allows to filter important amount of network traffic data, therefore only relevant data is selected for the remaining steps of the proposed approach. Also, important resources are saved when no abnormal entropy occurs. In order to determine the normal cluster, we estimate the information gain ratio based on the average entropy of the FSD features between the received network traffic data during the current time window and each one of the obtained clusters. As discussed in the previous section during a DDoS period the generated amount of attack traffic is largely bigger than the normal traffic. Hence, estimating the information gain ratio based on the FSD features allows identifying the two cluster that preserve more information about the DDoS attack and the cluster that contains only normal traffic. Therefore, the cluster that produce lower information gain ratio is considered as normal and the remaining clusters are considered as anomalous. The information gain ratio is computed for each cluster as follows:

ADVANTAGE:

- Where subsetw represents the received subset of network data during the time window w, Ci (i = 1, 2, 3) are the obtained clusters from subsetw and |Ci | is the size of the ith cluster. avgH(subset) is the average entropy of the FSD features of the input subset and |subset | represents the size
- The clustering of the incoming network traffic data allows reducing important amount of normal and noisy data before the pre-processing and classification steps. More than 6% of a whole traffic dataset can be filtered.

2.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ♦ **ECONOMICAL FEASIBILITY**
- ♦ **TECHNICAL FEASIBILITY**
- ♦ **SOCIAL FEASIBILITY**

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3. SYSTEM DESIGN

3.1 DEFINITION

System design is the process of defining the components, modules, interfaces, and data for a system to satisfy specified requirements. System development is the process of creating or altering systems, along with the processes, practices, models, and methodologies used to develop them.

Architectural design

The architectural design of a system emphasizes the design of the system architecture that describes the structure, behaviour and more views of that system and analysis.

Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems, designs are included. Logical design includes ER diagrams.

Physical design

The physical design relates to the actual input and output processes of the system. This is explained in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed. In physical design, the following requirements about the system are decided.

- Input requirement,
- Output requirements,
- Storage requirements,
- Processing requirements,
- System control and backup or recovery

Processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

3.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

- Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- Select methods for presenting information.
- Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

3.3 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

3.4 SYSTEM ARCHITECTURE

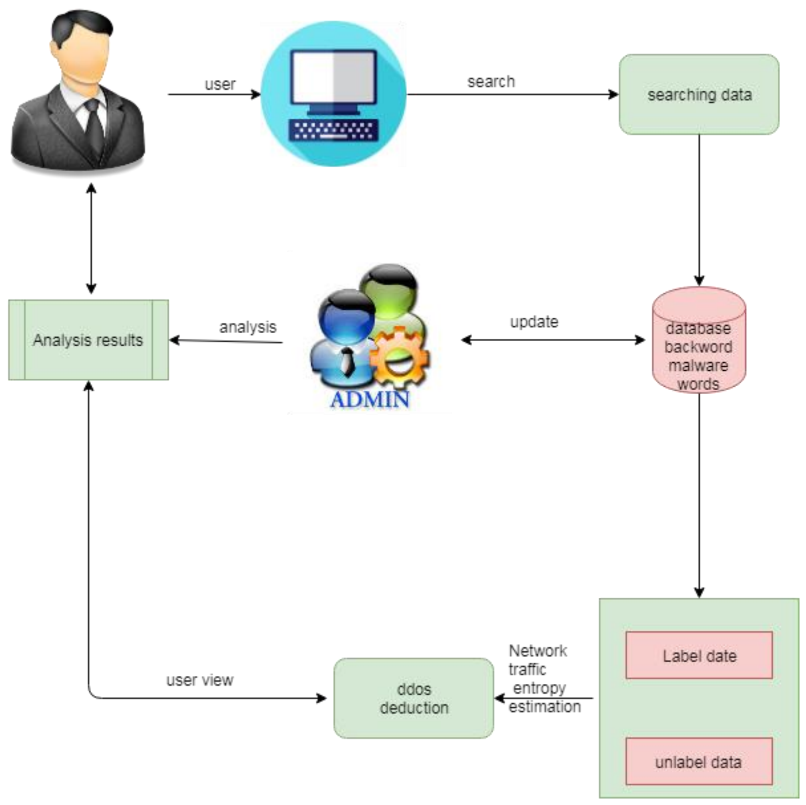


Fig.3.1 System Architecture

3.5 DATA FLOW DIAGRAM

The DFD is also called as bubble chart. The data flow diagram (DFD) is one of the most important modelling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system

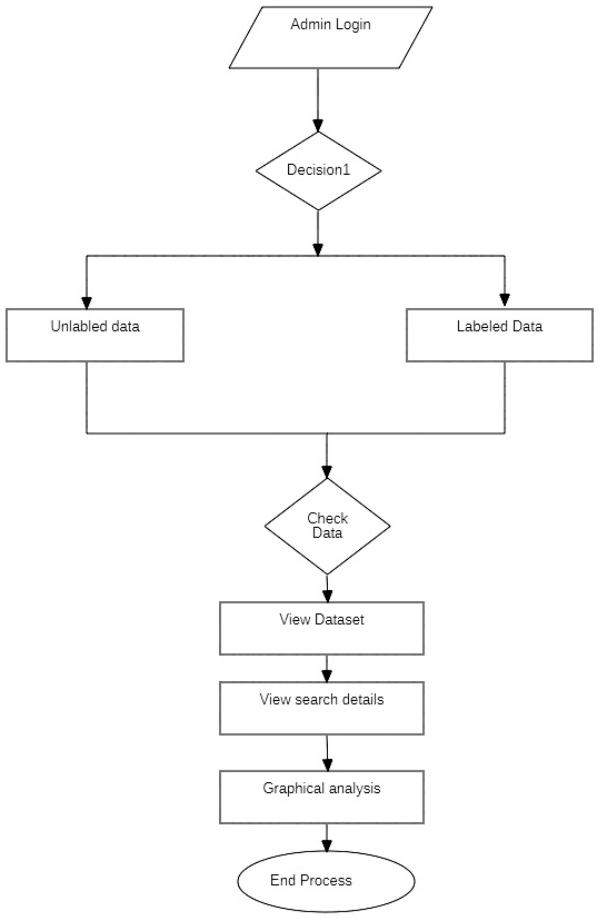


Fig:3.2 Data flow diagram

3.6 UML DIAGRAMS

- UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.
- The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML has two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.
- The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.
- The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

- UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

3.6.1 USE CASE DIAGRAM

- A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and 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, their goals (represented as use cases), and any dependencies between those use cases.
- The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

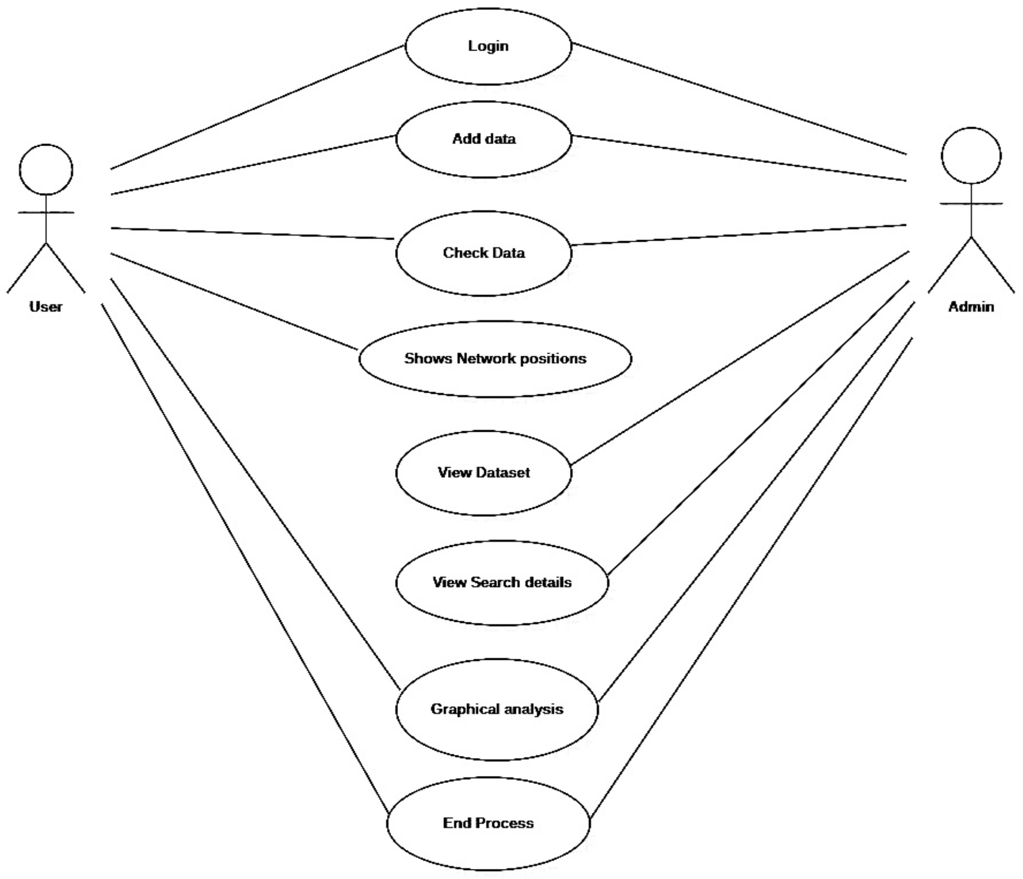


Figure 3.3 Use case diagram

3.6.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

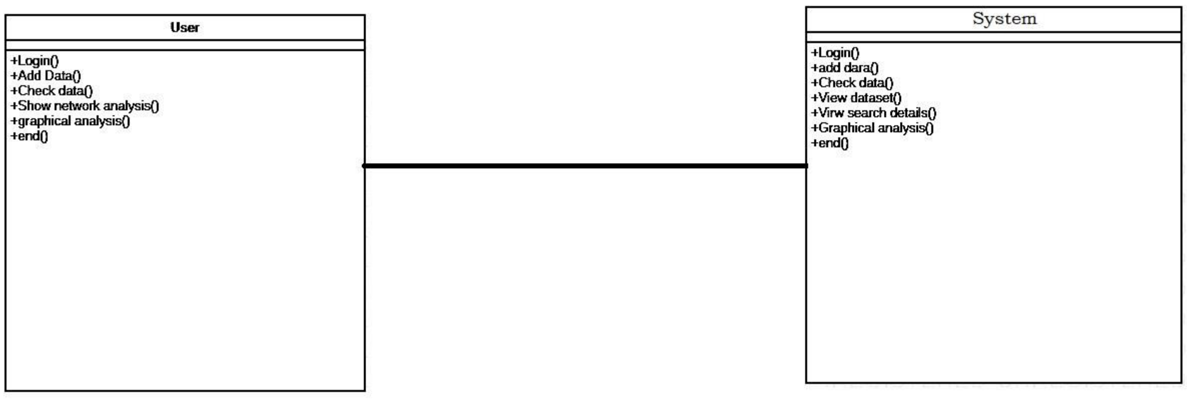


Figure 3.4 Class diagram

3.7 DATA DICTIONARY

ADMIN:

FIELD NAME	**DATA TYPE**	**DATA SIZE**	**DESCRIPTION**
Id	Int	10	Admin Id, Auto Generated
Password	Varchar	10	Login Password For Admin

DDOS DATASET:

FIELD NAME	**DATA TYPE**	**DATA SIZE**	**DESCRIPTION**
Ddos id	Int	10	Ddos Id, Auto Generator
Ddos data	Varchar	45	data
Attack Result	Varchar	10	Attack result

4. SYSTEM SPECIFICATIONS

HARDWARE REQUIREMENTS:

❖ **System**	**:**	Pentium IV 2.4 GHz.
❖ **Hard Disk**	**:**	40 GB.
❖ **RAM**	**:**	512 Mb.

SOFTWARE REQUIREMENTS:

❖ **Operating system**	**:**	Windows 7 Ultimate.
❖ **Coding Language**	**:**	Python.
❖ **Front-End**	**:**	Python.
❖ **Designing**	**:**	Html, css, JavaScript.
❖ **Data Base**	**:**	MySQL.

5. OVER VIEW OF LANGUAGE

5.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the nonprofit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active. Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file − Live Demo print "Hello, Python!"

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows − $ python test.py

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/pytho
n   print   "Hello,
Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable

$./test.py
```

This produces the following result −

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter. Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only. and exec not assert finally or break for pass class from print continue global raise def if return del import try elif in while else is with except lambda yield **Lines and Indentation**

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example − if True:

  print
"True"
else:

```
print
"False"
```

However, the following block generates an
error − if True:

```
print
"Answer"
print
"True"
else:
print
"Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python

import
sys try:

  # open file stream
file = open(file_name,
"w") except IOError:

  print "There was an error writing to",
file_name       sys.exit() print "Enter '",
file_finish, print "' When finished" while
file_text != file_finish:
  file_text = raw_input("Enter
text: ")        if file_text ==
file_finish:
    # close the file          file.close
break                file.write(file_text)
file.write("\n") file.close() file_name
= raw_input("Enter filename: ") if
len(file_name) == 0:
  print "Next time please enter
something"   sys.exit() try:
  file   =   open(file_name,   "r")
except IOError:      print "There
was   an   error   reading   file"
sys.exit()  file_text  =  file.read()
file.close() print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example − total = item_one + \   item_two + \

        item_three

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example − days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']

Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal − word = 'word' sentence = "This is a sentence." paragraph = """This is a paragraph. It is made up of multiple lines and sentences."""

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them. Live Demo

#!/usr/bin/pyt
hon # First
comment

print "Hello, Python!" # second comment

This produces the following result −
Hello, Python!

You can type a comment on the same line after a statement or expression −
name = "Madisetti" # This is again comment

You can comment multiple lines as
follows − # This is a comment.

# This is a comment, too.

# This is a comment, too.

# I said that already.

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

'''

This is a multiline

comment.

'''

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement. Waiting for the User
The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

#!/usr/bin/python

raw_input("\n\nPress the enter key to exit.")

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application. Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example − if expression :

```
  suite
elif
expressio
n :


suite
else  :
suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h usage: python [option] ... [-c cmd | -m mod | file |
-] [arg] ... Options and arguments (and corresponding
environment variables):
-c     cmd : program passed in as string (terminates option list)

-d     : debug output from parser (also PYTHONDEBUG=x)
```

-E   : ignore environment variables (such as PYTHONPATH)

-h   : print this help message and exit

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example − list1 = ['physics', 'chemistry', 1997, 2000]; list2 = [1, 2, 3, 4, 5]; list3 = ["a", "b", "c", "d"]

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on. A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example − tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5); tup3 = "a", "b", "c", "d";
The empty tuple is written as two parentheses containing nothing
− tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value − tup1 = (50,);
Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

#!/usr/bin/python tup1 = ('physics',
'chemistry', 1997, 2000); tup2 = (1,
2, 3, 4, 5, 6, 7); print "tup1[0]: ",
tup1[0]; print "tup2[1:5]: ",
tup2[1:5];

When the above code is executed, it produces the following
result − tup1[0]: physics tup2[1:5]: [2, 3, 4, 5]

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

#!/usr/bin/python dict = {'Name': 'Zara',
'Age': 7, 'Class': 'First'} print
"dict['Name']: ", dict['Name'] print
"dict['Age']: ", dict['Age']

When the above code is executed, it produces the following
result − dict['Name']: Zara dict['Age']: 7

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo #!/usr/bin/python dict =
{'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']

When the above code is executed, it produces the following result −
dict['Alice']:

Traceback (most recent call last):

File "test.py", line 4, in
<module> print "dict['Alice']: ",
dict['Alice'];

KeyError: 'Alice'

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python dict = {'Name': 'Zara',
'Age': 7, 'Class': 'First'} dict['Age'] = 8; #
update existing entry dict['School'] =
"DPS School"; # Add new entry print
"dict['Age']: ", dict['Age'] print
"dict['School']: ", dict['School']
```

When the above code is executed, it produces the following
result − dict['Age']: 8 dict['School']: DPS School Delete
Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example − Live Demo

```
#!/usr/bin/python dict = {'Name': 'Zara',
'Age': 7, 'Class': 'First'} del dict['Name']; #
remove entry with key 'Name' dict.clear();
# remove all entries in dict del dict ; #
delete entire dictionary print "dict['Age']: ",
dict['Age'] print "dict['School']: ",
dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more − dict['Age']:

```
Traceback (most recent call
last): File "test.py", line 8, in
<module> print "dict['Age']:
", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note − del() method is discussed in subsequent section.

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python  dict  =  {'Name':  'Zara',
'Age':     7,     'Name':     'Manni'}     print
"dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python     dict     =
{['Name']:  'Zara',  'Age':  7}
print      "dict['Name']:      ",
dict['Name']
```

When the above code is executed, it produces the following result − Traceback (most recent call last):   File "test.py", line 3, in <module>   dict = {['Name']: 'Zara', 'Age': 7};

TypeError: unhashable type: 'list'

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates − Live Demo #!/usr/bin/python tup1 = (12, 34.56); tup2 = ('abc', 'xyz');

# Following action is not valid for tuples

# tup1[0] = 100;

# So let's create a new tuple as
follows tup3 = tup1 + tup2; print
tup3;

When the above code is executed, it produces the following result −

(12, 34.56, 'abc', 'xyz')

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

#!/usr/bin/python tup = ('physics',
'chemistry', 1997, 2000); print tup;
del tup; print "After deleting tup :
"; print tup;

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more − ('physics', 'chemistry', 1997, 2000) After deleting tup :

Traceback (most recent call
last): File "test.py", line 9, in
<module> print tup;
NameError: name 'tup' is not defined

5.2 DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

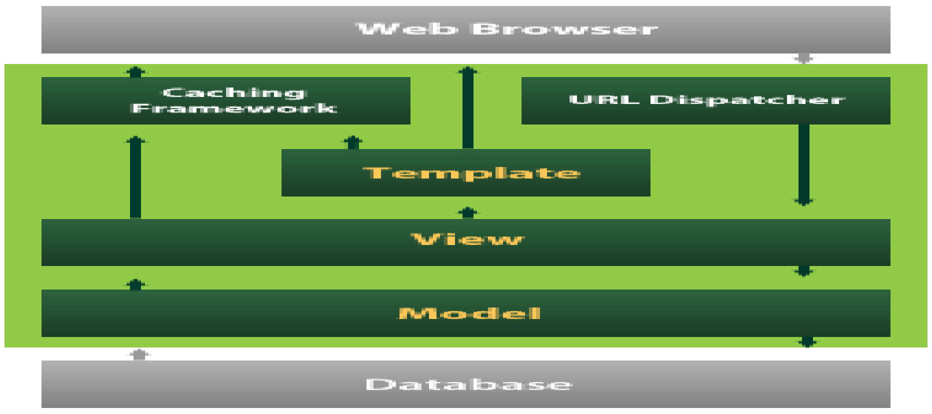


Figure 5. 2: Django Flowchart

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models

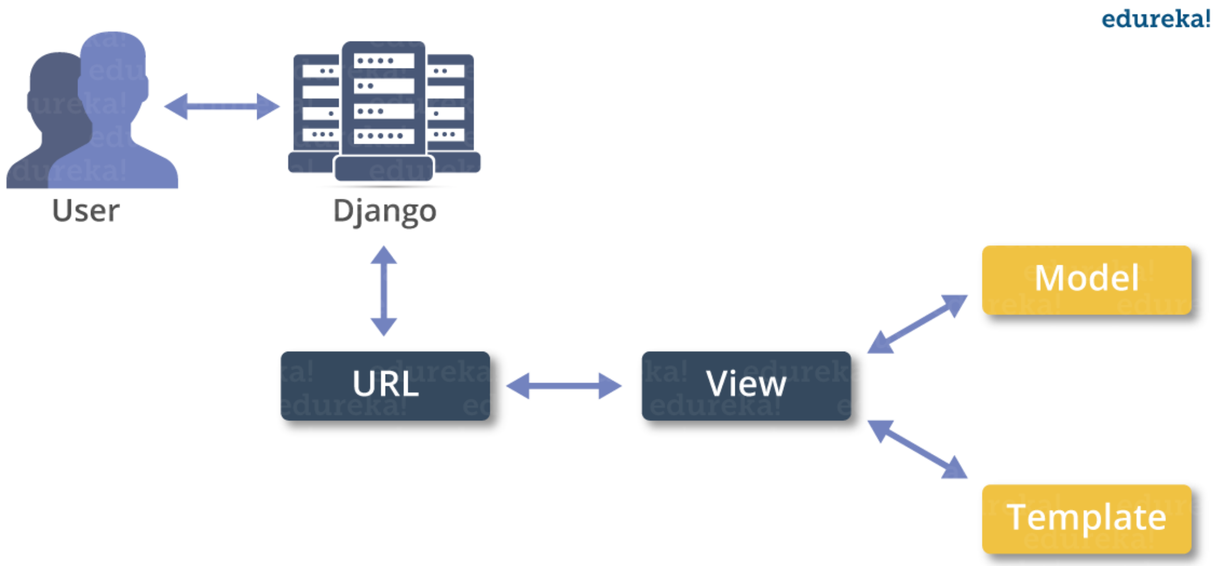


Figure 5. 2.1: Django Architecture

6. IMPLEMENTATION

6.1 System Modules

MODULES:

There are three modules can be divided here for this project they are listed as below

- User Apps
- DDOS Attack Deduction
- Classifications of DDOS attack
- Graphical analysis

From the above four modules, project is implemented. Bag of discriminative words are achieved

1. User Apps

User handling for some various times of smart phones, desktops, laptops and tablets .If any kind of devices attacks for some unauthorized Malware software's .In this Malware on threats for user personal dates includes for personal contact, bank account numbers and any kind of personal documents are hacking in possible.

2. DDOS Attack Deduction

User searches the any link Notably, not all network traffic data generated by malicious apps correspond to malicious traffic. Many malware take the form of repackaged benign apps; thus, Malware can also contain the basic functions of a benign app. Subsequently, the network traffic they generate can be characterized by mixed benign and malicious network traffic. We examine the traffic flow header using Co-clustering algorithm from the natural language processing (NLP).

3. Classifications of DDOS Attack:

Here, we compare the classification performance of Co-clustering algorithm with other popular machine learning algorithms. We have selected several popular classification algorithms. For all algorithms, we attempt to use multiple sets of parameters to maximize the performance of each algorithm. Using Co-clustering algorithm algorithms classification for malware bag-of-words weight age.

4. Graphical analysis

The graph analysis is done by the values taken from the result analysis part and it can be analyzed by the graphical representations. Such as pie chart, pyramid chart and funnel chart here in this project.

6.2 PROPOSED ALGORITHM

Co-clustering is rather a recent paradigm for unsupervised data analysis, but it has become increasingly popular because of its potential to discover latent local patterns, otherwise unapparent by usual unsupervised algorithms such as k-means. Wide deployment of co-clustering, however, requires addressing a number of practical challenges such as data transformation, cluster initialization, scalability, and so on. Therefore, this thesis focuses on developing sophisticated co-clustering methodologies to maturity and its ultimate goal is to promote co-clustering as an invaluable and indispensable unsupervised analysis tool for varied practical applications. To achieve this goal, we explore the three specific tasks: (1) development of co-clustering algorithms to be functional, adaptable, and scalable (co-clustering algorithms); (2) extension of co-clustering algorithms to incorporate application-specific requirements (extensions); and (3) application of co-clustering algorithms broadly to existing and emerging problems in practical application domains (applications). As for co-clustering algorithms, we develop two fast Minimum Sum-Squared Residue Co-clustering (MSSRCC) algorithms [CDGS04], which simultaneously cluster data points and features via an alternating minimization scheme and generate co-clusters in a “checkerboard” structure. The first captures co-clusters with constant values, while the other discovers co-clusters with coherent “trends” as well as constant values. We note that the proposed algorithms are two special cases (bases 2 and 6 with Euclidean distance, respectively) of the general co-clustering framework, Bregman Co-clustering (BCC) [BDG+07], which contains six Euclidean BCC and six I-divergence BCC algorithms. Then, we substantially enhance the performance of the two MSSRCC algorithms by escaping from poor local minima and resolving the degeneracy problem of generating empty clusters in partitional clustering algorithms through the three specific strategies: (1) data transformation; (2) deterministic spectral initialization; and (3) local search strategy. Concerning co-clustering extensions, we investigate general algorithmic strategies for the general BCC framework, since it is applicable to a large class of distance measures and data types. We first formalize various data transformations for datasets with varied scaling and shifting factors, mathematically justify their effects on the six Euclidean BCC algorithms, and empirically validate the analysis results. We also adapt the local search strategy, initially developed for the two MSSRCC algorithms, to all the twelve BCC algorithms. Moreover, we consider variations of cluster assignments and cluster updates, including greedy vs. non-greedy cluster assignment, online vs. batch cluster update, and so on. Furthermore, in order to provide better scalability and usability, we parallelize all the twelve BCC algorithms, which are capable of co-clustering large-scaled datasets over multiple processors. Regarding co-clustering applications, we extend the functionality of BCC to incorporate application-specific requirements: (1) discovery of inverted patterns, whose goal is to find anti-correlation; (2) discovery of coherent co-clusters from noisy data, whose purpose is to do dimensional reduction and feature selection; and (3) discovery of patterns from time-series data, whose motive is to guarantee critical time-locality. Furthermore, we employ co-clustering to pervasive computing for mobile devices, where the task is to extract latent patterns from usage logs as well as to recognize specific situations of mobile-device users. Finally, we demonstrate the applicability of our proposed algorithms for aforementioned applications through empirical results on various synthetic and real-world datasets. In summary, we present co-clustering algorithms to discover latent local patterns, propose their algorithmic extensions to incorporate specific requirements, and provide their applications to a wide range of practical domains.

6.3 SOURCE CODE

Manage.py

```
#!/usr/bin/env python
import os
import sys


if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "DDOS_ATTACK.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

Settings.py

```
"""
Django settings for DDOS_ATTACK project.

Generated by 'django-admin startproject' using Django 1.11.5.

For more information on this file, see
https://docs.djangoproject.com/en/1.11/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.11/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '=wkl9m18)fl45s@u@dek*l6^j^vur((k58i-51cg$^+c6(nw7i'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'data_admins',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'DDOS_ATTACK.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [(os.path.join(BASE_DIR,'assets/templates'))],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'DDOS_ATTACK.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'ddos_attack',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': '127.0.0.1',
        'PORT': '3306',
    }
}
```

```
# Password validation
# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.11/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.11/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS=[os.path.join(BASE_DIR,'assets/static')]
```

url.py

```
"""DDOS_ATTACK URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.11/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.conf.urls import url, include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import url
from django.contrib import admin
from data_admins import views as admins

urlpatterns = [
    url(r'^admin/', admin.site.urls),

    url('^$',admins.index,name="index"),
    url('user/register', admins.register, name="register"),
    url('user/add_data',admins.add_data,name="add_data"),
    url('user/userpage',admins.userpage,name="userpage"),
    url('user/labeled_data',admins.labeled_data,name="labeled_data"),
    url('user/unlabeled_data',admins.unlabeled_data,name="unlabeled_data"),
    url('user/ddos_analysis',admins.ddos_analysis,name="ddos_analysis"),
    url('user/chart_page/(?P<chart_type>\w+)',admins.chart_page,name="chart_page"),
]
```

7. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

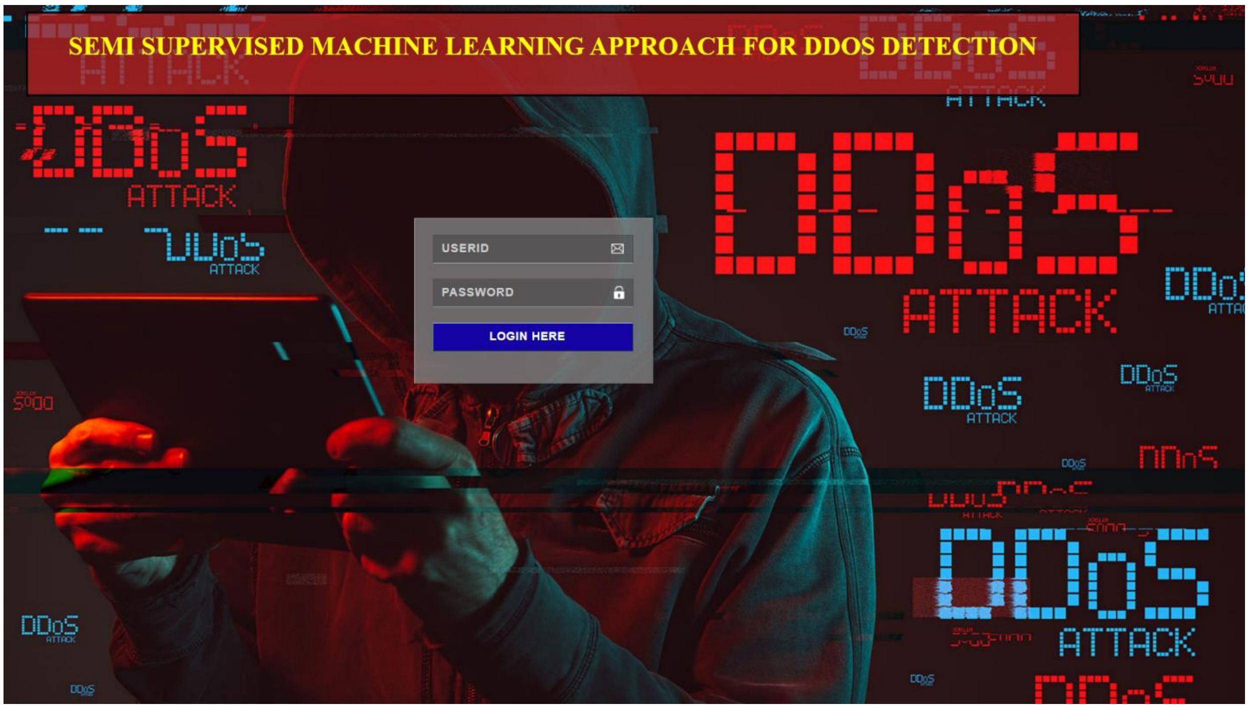
TEST CASES:

S.NO	Test Cases	Expected Result	Result
1	Login	If Username and Password is correct then it will getting valid page	Pass
2	Login	If Username and Password is Incorrect then it will getting Invalid page	Pass
3	Add Data	Data added will be clustered into labelled data if it is a valid attack	Pass
4	Add Data	Data added will be clustered into Unlabelled data if it is not a valid attack	Pass

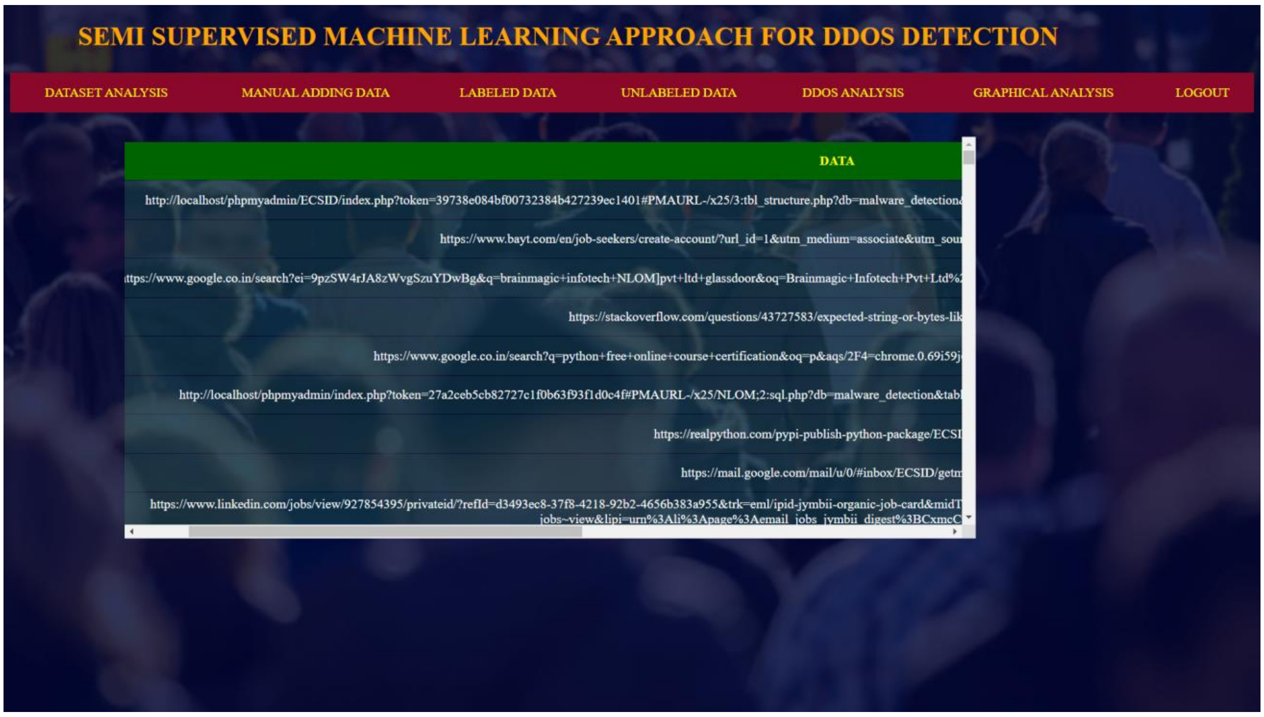
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8. OUTPUT SCREENS

8.1 Login

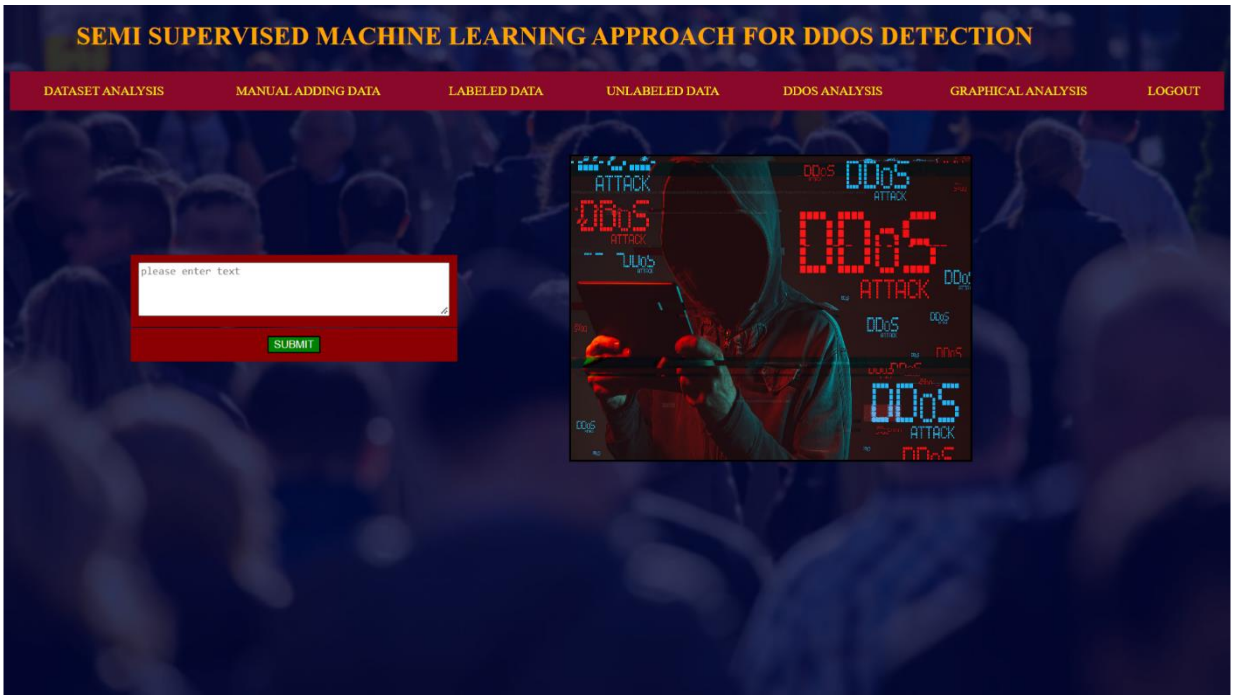

SEMI SUPERVISED MACHINE LEARNING APPROACH FOR DDOS DETECTION
USERID
PASSWORD
LOGIN HERE

8.2 User Page

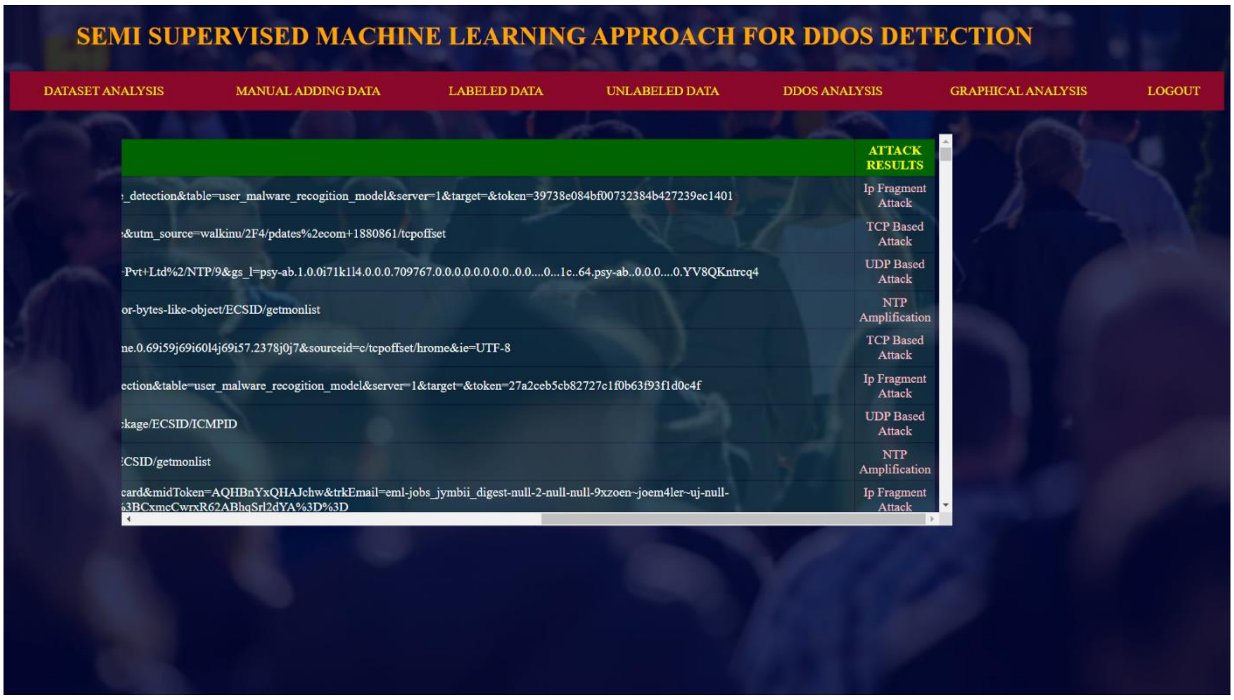


DATA
http://localhost/phpmyadmin/ECSID/index.php?token=39738e084bf00732384b427239ec1401#PMAURL-/x25/3:tbl_structure.php?db=malware_detection
https://www.bayt.com/en/job-seekers/create-account/?url_id=1&utm_medium=associate&utm_sou
ttps://www.google.co.in/search?ei=9pzSW4rJA8zWvgSzuYDwBg&q=brainmagic+infotech+NLOM]pvt+ltd+glassdoor&oq=Brainmagic+Infotech+Pvt+Ltd%
https://stackoverflow.com/questions/43727583/expected-string-or-bytes-lik
https://www.google.co.in/search?q=python+free+online+course+certification&oq=p&aqs/2F4=chrome.0.69i59j
http://localhost/phpmyadmin/index.php?token=27a2ceb5cb82727c1f0b63f93f1d0c4f#PMAURL-/x25/NLOM;2:sql.php?db=malware_detection&tab
https://realpython.com/pypi-publish-python-package/ECSI
https://mail.google.com/mail/u/0/#inbox/ECSID/getm
https://www.linkedin.com/jobs/view/927854395/privateid/?refId=d3493ec8-37f8-4218-92b2-4656b383a955&trk=eml/ipid-jymbii-organic-job-card&midT jobs~view&lipi=urn%3Ali%3Apage%3Aemail_jobs_jymbii_digest%3BCxmcC

8.3 Add Data

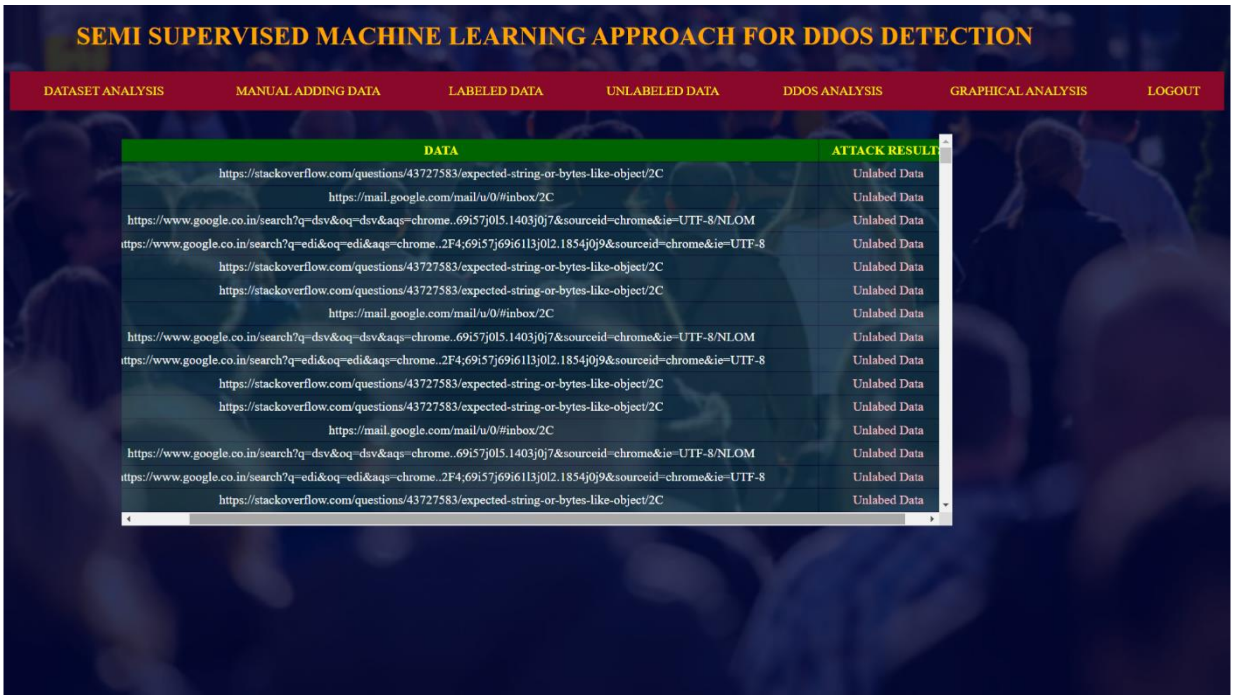


8.4 Labeled Data



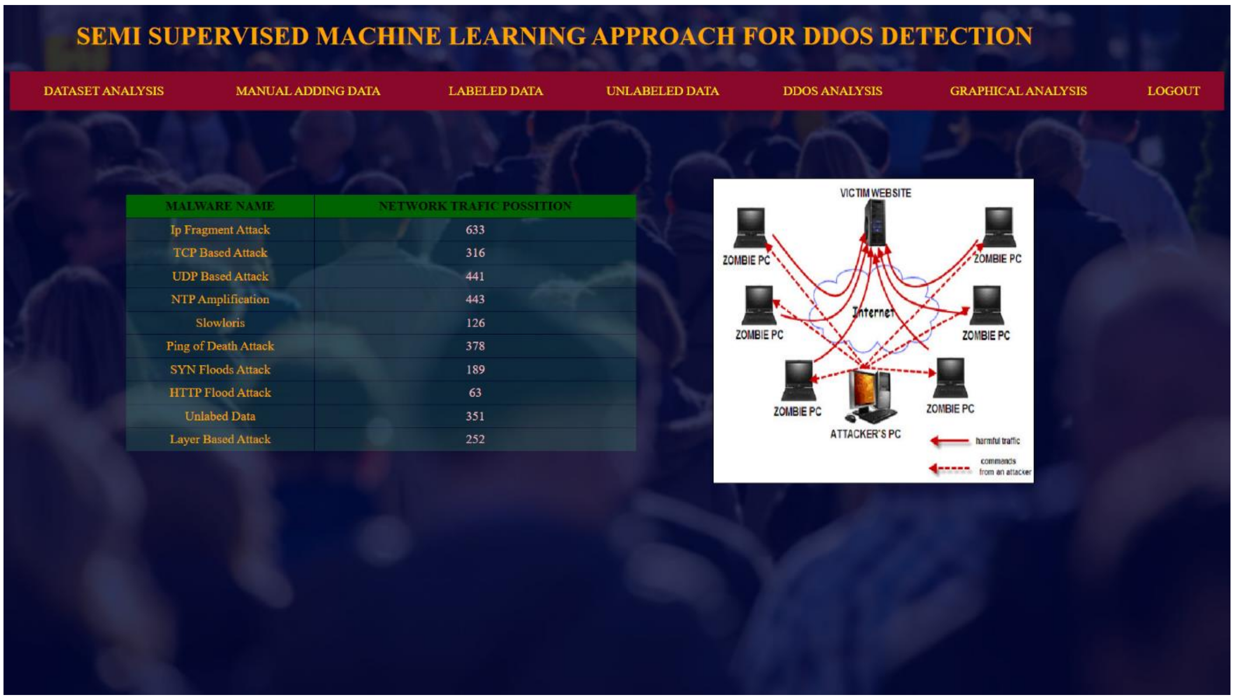
	ATTACK RESULTS
_detection&table=user_malware_recogition_model&server=1&target=&token=39738e084bf00732384b427239ec1401	Ip Fragment Attack
&utm_source=walkinu/2F4/pdates%2ecom+1880861/tcpoffset	TCP Based Attack
-Pvt+Ltd%2/NTP/9&gs_l=psy-ab.1.0.0i71k1l4.0.0.0.709767.0.0.0.0.0.0.0.0..0.0....0...1c..64.psy-ab..0.0.0....0.YV8QKntrcq4	UDP Based Attack
or-bytes-like-object/ECSID/getmonlist	NTP Amplification
ne.0.69i59j69i60l4j69i57.2378j0j7&sourceid=c/tcpoffset/hrome&ie=UTF-8	TCP Based Attack
ection&table=user_malware_recogition_model&server=1&target=&token=27a2ceb5cb82727c1f0b63f93f1d0c4f	Ip Fragment Attack
kage/ECSID/ICMPID	UDP Based Attack
CSID/getmonlist	NTP Amplification
card&midToken=AQHBnYxQHAJchw&trkEmail=eml-jobs_jymbii_digest-null-2-null-null-9xzoen~joem4ler~uj-null-3BCxmcCwrxR62ABhqSrl2dYA%3D%3D	Ip Fragment Attack

8.5 Unlabeled Data



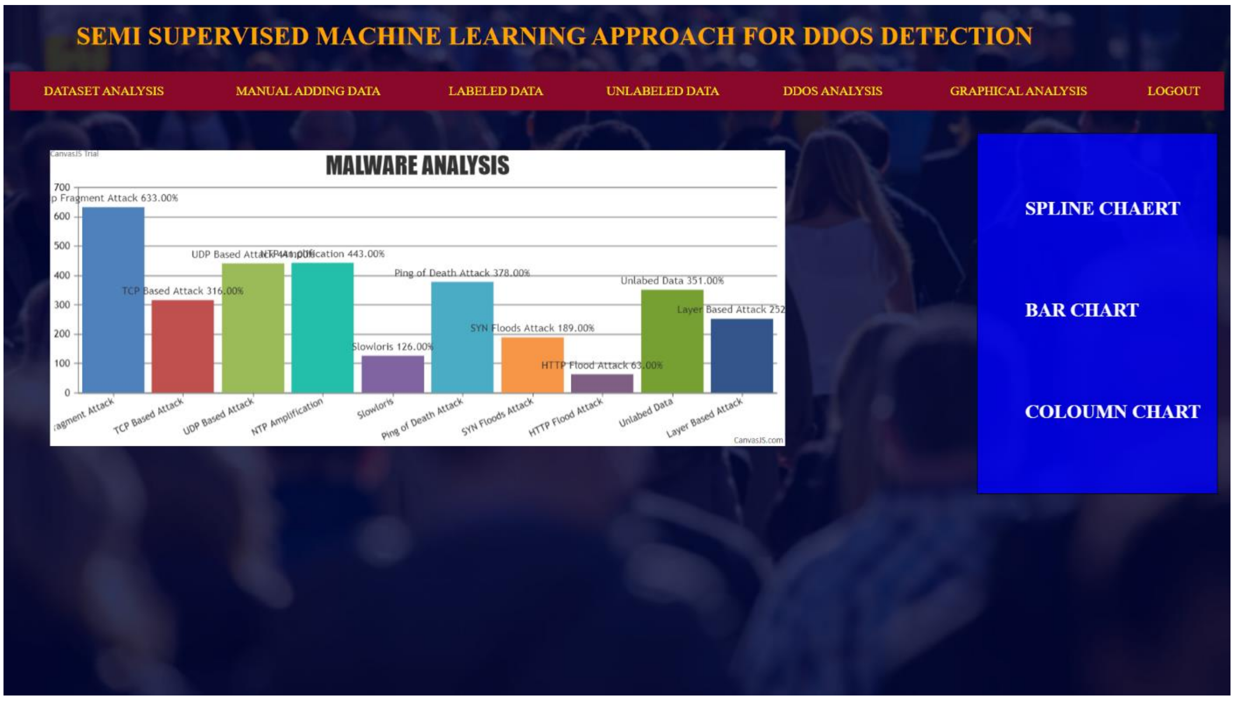
DATA	ATTACK RESULT
https://stackoverflow.com/questions/43727583/expected-string-or-bytes-like-object/2C	Unlabed Data
https://mail.google.com/mail/u/0/#inbox/2C	Unlabed Data
https://www.google.co.in/search?q=dsv&oq=dsv&aqs=chrome..69i57j0l5.1403j0j7&sourceid=chrome&ie=UTF-8/NLOM	Unlabed Data
ttps://www.google.co.in/search?q=edi&oq=edi&aqs=chrome..2F4;69i57j69i61l3j0l2.1854j0j9&sourceid=chrome&ie=UTF-8	Unlabed Data
https://stackoverflow.com/questions/43727583/expected-string-or-bytes-like-object/2C	Unlabed Data
https://stackoverflow.com/questions/43727583/expected-string-or-bytes-like-object/2C	Unlabed Data
https://mail.google.com/mail/u/0/#inbox/2C	Unlabed Data
https://www.google.co.in/search?q=dsv&oq=dsv&aqs=chrome..69i57j0l5.1403j0j7&sourceid=chrome&ie=UTF-8/NLOM	Unlabed Data
ttps://www.google.co.in/search?q=edi&oq=edi&aqs=chrome..2F4;69i57j69i61l3j0l2.1854j0j9&sourceid=chrome&ie=UTF-8	Unlabed Data
https://stackoverflow.com/questions/43727583/expected-string-or-bytes-like-object/2C	Unlabed Data
https://stackoverflow.com/questions/43727583/expected-string-or-bytes-like-object/2C	Unlabed Data
https://mail.google.com/mail/u/0/#inbox/2C	Unlabed Data
https://www.google.co.in/search?q=dsv&oq=dsv&aqs=chrome..69i57j0l5.1403j0j7&sourceid=chrome&ie=UTF-8/NLOM	Unlabed Data
ttps://www.google.co.in/search?q=edi&oq=edi&aqs=chrome..2F4;69i57j69i61l3j0l2.1854j0j9&sourceid=chrome&ie=UTF-8	Unlabed Data
https://stackoverflow.com/questions/43727583/expected-string-or-bytes-like-object/2C	Unlabed Data

8.6 DDOS Analysis

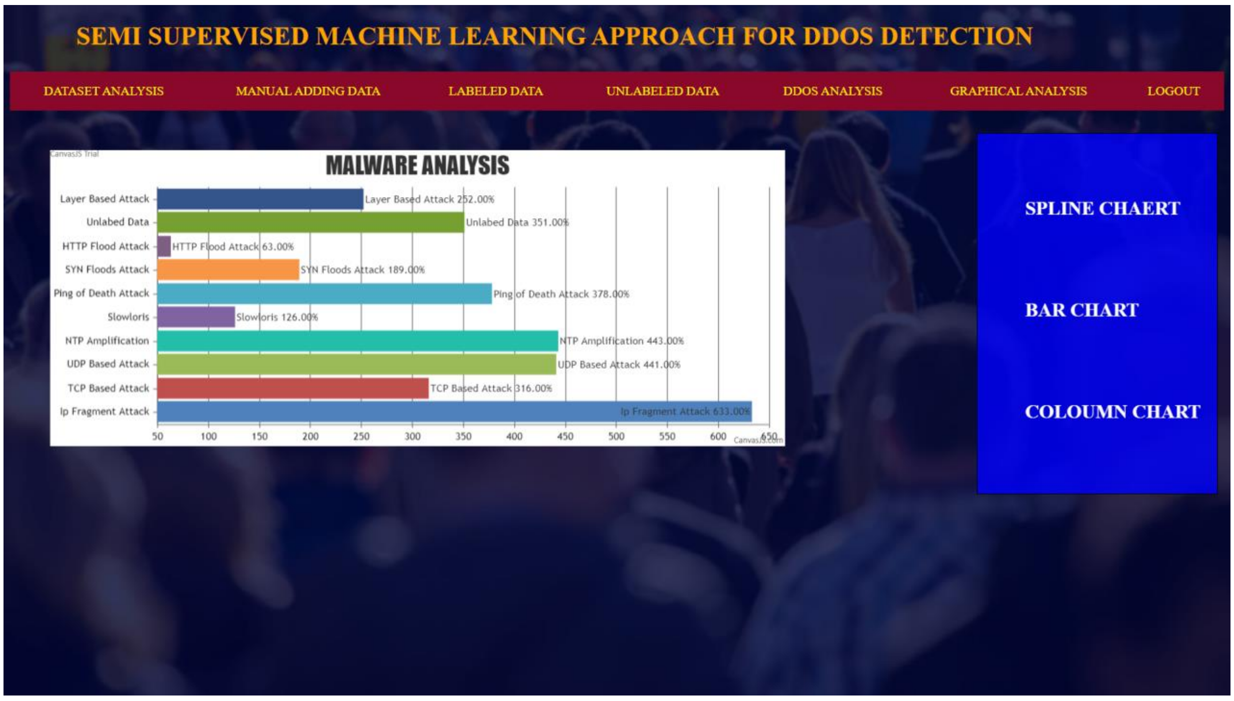


MALWARE NAME	NETWORK TRAFIC POSSITION
Ip Fragment Attack	633
TCP Based Attack	316
UDP Based Attack	441
NTP Amplification	443
Slowloris	126
Ping of Death Attack	378
SYN Floods Attack	189
HTTP Flood Attack	63
Unlabed Data	351
Layer Based Attack	252

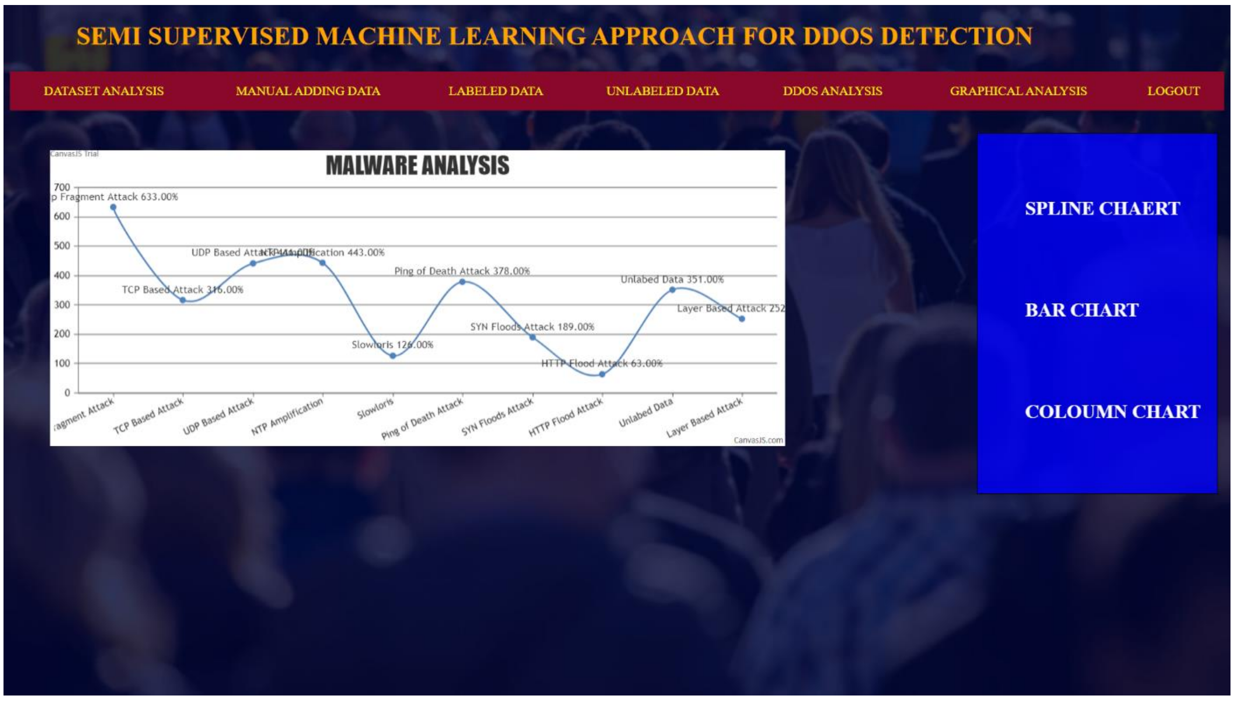
8.7 Coloum Graph



8.8 Bar Graph



8.9 Spline Graph



9. CONCLUSION

Android has a new and fastest growing threat to malware. Currently, many research methods and antivirus scanners are not hazardous to the growing size and diversity of mobile malware. As a solution, we introduce a solution for mobile malware detection using network traffic flows, which assumes that each HTTP flow is a document and analyses HTTP flow requests using NLP string analysis. The N-Gram line generation, feature selection algorithm, and SVM algorithm are used to create a useful malware detection model. Our evaluation demonstrates the efficiency of this solution, and our trained model greatly improves existing approaches and identifies malicious leaks with some false warnings. The harmful detection rate is 99.15%, but the wrong rate for harmful traffic is 0.45%. Using the newly discovered malware further verifies the performance of the proposed system. When used in real environments, the sample can detect 54.81% of harmful applications, which is better than other popular anti-virus scanners. As a result of the test, we show that malware models can detect our model, which does not prevent detecting other virus scanners. Obtaining basically new malicious models Virus Total detection reports are also possible. Added, Once new tablets are added to training samples, we will please re-train and refresh and update the new malware.

BIBLIOGRAPHY

1. Bhuyan MH, Bhattacharyya DK, Kalita JK (2015) An empirical evaluation of information metrics for low-rate and high- rate ddos attack detection. Pattern Recogn Lett 51:1–7
2. Lin S-C, Tseng S-S (2004) Constructing detection knowledge for ddos intrusion tolerance. Exp Syst Appl 27(3):379–390
3. Chang RKC (2002) Defending against flooding-based distributed denial-of-service attacks: a tutorial. IEEE Commun Mag 40(10):42–51
4. Yu S (2014) Distributed denial of service attack and defence. Springer, Berlin
5. Wikipedia (2016) 2016 dyn cyberattack. https://en.wikipedia.org/ wiki/2016 Dyn cyberattack. (Online; accessed 10 Apr 2017)
6. theguardian (2016) Ddos attack that disrupted internet was largest of its kind in history, experts say. https://www.theguardian.com/ technology/2016/oct/26/ddos-attack-dyn-mirai-botnet. (Online; accessed 10 Apr 2017)
7. Kalegele K, Sasai K, Takahashi H, Kitagata G, Kinoshita T (2015) Four decades of data mining in network and systems management. IEEE Trans Knowl Data Eng 27(10):2700–2716
8. Han J, Pei J, Kamber M (2006) What is data mining. Data mining: concepts and techniques. Morgan Kaufinann
9. Berkhin P (2006) A survey of clustering data mining techniques. In: Grouping multidimensional data. Springer, pp 25–71
10. Mori T (2002) Information gain ratio as term weight: the case of summarization of ir results. In: Proceedings of the 19th international conference on computational linguistics, vol 1. Association for Computational Linguistics, pp 1–7.
11. Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. Mach Learn 63(1):3–42
12. Tavallaee M, Bagheri E, Lu W, Ghorbani A-A (2009) A detailed analysis of the kdd cup 99 data set. In: Proceedings of the second IEEE symposium on computational intelligence for security and defence applications 2009
13. Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA (2012) Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput Secur 31:357–374

14. Moustafa N, Slay J (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: Military communications and information systems conference (MilCIS), 2015. IEEE, pp 1–6
15. Moustafa N, Slay J (2016) The evaluation of network anomaly detection systems: statistical analysis of the unsw- nb15 data set and the comparison with the kdd99 data set. Inf Secur J: Glob Perspect 25:18– 31s.