Malicious Attacks Detection Using Machine Learning

Submitted in partial fulfillment of the requirements for

the award of

Bachelor of Engineering Degree in Computer Science and Engineering

By

YEMIREDDY CHAITANYA

(38110092)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MARCH -2022

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **YEMIREDDY CHAITANYA** (Reg.no:38110092) who carried out the project entitled "**Malicious Attacks Detection Using Machine Learning**" under my supervision from JUNE 2021 to NOVEMBER 2021

Internal Guide

Dr. A. Jesudoss, M.E., Ph.D.,

Head of the Department

(Dr.S Vigneshwari & Dr.L.Lakshmanan)

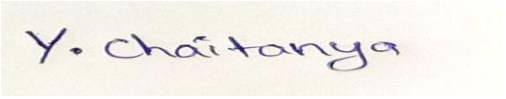
Submitted for Viva voce Examination held on _____

Internal Examiner **External Examiner**

(Ms.Yogitha) (Dr.Mathivanan)

DECLARATION

I **YEMIREDDY CHAITANYA** here by declare that the Project Report entitled **"MALICIOUS ATTACKS DETECTION USING MACHINE LEARNING "** done by me under the guidance of **Dr.A.Jesudoss,ME.,Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering in Computer Science.



DATE:6th March 2022
PLACE:Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my project guide **Dr.A.Jesudoss M.E.,Ph.D** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project.

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

ABSTRACT

Bot detection using machine learning (ML), with network flow-level features, has been extensively studied in the literature. However, existing flow-based approaches typically incur a high computational overhead and do not completely capture the network communication patterns, which can expose additional aspects of malicious hosts. Recently, bot detection systems that leverage communication graph analysis using ML have gained attention to overcome these limitations.

A graph-based approach is rather intuitive, as graphs are true representation of network communications. In this paper, we propose BotChase, a two-phased graph-based bot detection system that leverages both unsupervised and supervised ML. The first phase prunes presumable benign hosts, while the second phase achieves bot detection with high precision. Our prototype implementation of BotChase detects multiple types of bots and exhibits robustness to zero-day attacks. It also accommodates different network topologies and is suitable for large-scale data. Compared to the state-of-the-art, BotChase outperforms an end-to-end system that employs flow-based features and performs particularly well in an online setting.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

MATLAB	Matrix Laboratory
PD	Pandas
NLTK	Natural Language Tool Kit Computing
NX	Networkx
TTK	Tkinter

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade 'A' By NAAC | 12B Status by UGC |Approved By AICTE
www.sathyabama.ac.in

School of Computing

Department of Information Technology

Certificate of Presentation

This is to certify that

YEMIREDDY CHAITANYA

of

Sathyabama Institute of Science and Technology

has presented a paper entitled

Malicious Attacks Detection using Machine Learning

in the AICTE sponsored

International Conference on Artificial Intelligence and Machine Learning (IAIM-2022),

held from **27th January 2022 - 29th January 2022.**

Dr. R. SUBHASHINI	Dr.T. SASIKALA	Dr. T. SASIPRABA
HOD [Information Technology]	*Dean [School of Computing]*	*Vice Chancellor*

CHAPTER 1

INTRODUCTION

Now a days everyone is storing their information in their systems. Here comes a problem in providing security to their systems. On other hand cyber-attacks are also increasing randomly which can hack your personal data like photos, social media and chats. Bot attacks increased worldwide. There are also some servers getting hacked which contains data of some lakhs people, where hacking a server is equal to hacking some lakhs people data.

Botnet is also a type of cyber-attack which is a collection of internet-connected devices, where these devices are called as bot. By using this bots the attacker can also hack a big servers. These bots all together called as bot army. Botnet can make time-consuming tasks easier because of its army. Botnet also perform helpful tasks people are using it for malicious works. It is also a source of many malicious activities. The different models of botnet are Client/Server .There are many types in botnet like centralized, client-server, decentralized and peer-to-peer models and attacks such as DDoS, phishing, cryptojacking, snooping, bricking, Brute force and spambots. Common Botnet actions are Email spam, Financial breach, Targeted intrusions. A bot herder can do a collective of hijacked devices by using remote commands. Once your machine is infected, it becomes a bot, you may not even know. Botnet leads to Financial theft, Informational theft, Sabotage of services, Selling access to other criminals. The 3 main components of botnet are the bots, Botnet attacks has been increased in the recent years at the same time different types of Botnet detection frameworks are also increased.

The hacker can access the device only when his application was in the device. Once his application started running in the device then he can steal, change or destroy information. The hacker can also steal money, username and passwords. The hacker can also change your confidential data. Also install and run any application in your system he want. All the devices which are connected to the internet can be hacked by the hacker. The more targeted devices like desktop and laptops which runs on Windows OS or macOS. Mobiles are next target devices as more people are using by connecting them to the internet. Recent years connecting devices to the internet has increased rapidly botnets also create from connected devices has become more noted.

First the hacker will start by injecting the malware infection to your device. some download links to the target device to hack the device. For example Trojan Horse (Happy New Year! Click here to see magic). If the owner of the device does not know about whether the download link is an attacker link and if he click on the link then the hacker application will get download in the device and sit around wait for command from the main system (hacker system). Now the hacker can access everything from his device. In order not to get attacked by hackers he should know all the malware links, so he can save his device from hacker. To stay away from malware links his device should able to find the malware links or prevent the initial infection or identify

an existing infection. Botnet attacks are hard to detect. Preventing botnet attacks is more difficult. Yet we can still take certain measures to prevent botnet attacks.

1.1 OVERVIEW

Cyberattacks are on the rise these days. Many systems are getting infected by attacks to overcome these attacks, In the past, we used signature-based research. However, as technology developed, attacks became more sophisticated and we used k-means and decision trees to see how many bots were targeted and how many were not. If there is an attack, we will find how many bots were attacked or detected and we will give the number.

1.2 OBJECTIVE

A botnet is a collection of bots, agents in compromised hosts, controlled by botmasters via command and control (C2) channels. A malevolent adversary controls the bots through botmaster, which could be distributed across several agents that reside within or outside the network. Hence, bots can be used for tasks ranging from distributed denial-of-service (DDoS), to massive-scale spamming, to fraud and identify theft. While bots thrive for different sinister purposes, they exhibit a similar behavioral pattern when studied up-close. The intrusion kill-chain dictates the general phases a malicious agent goes through in-order to reach and infest its target.

1.3 SCOPE

For this phase in BotChase, we evaluate four SL techniques, namely DT, LR, SVM and FNN. We use DT with Gini instance split rule algorithm, LR without regularization, and SVM with the Gaussian kernel and a soft margin penalty of 1. Moreover, NN is configured to use cross entropy as an error function and 10 hidden layers of 1000 units each. The DT classifier shows the best performance with the small dataset, as depicted in Table IV. It successfully detects all bots in the test dataset, with only a single FP out of the 366871 benign hosts. In contrast, all other classifiers are lackluster and unable to recall even a single bot from the dataset. We believe this is because all classifiers, except DT, rely on gradient-descent for errorcorrection. This implies that every single node in the dataset will affect the end-hypothesis function. Thus, with a dataset that is unbalanced, the hypothesis will be biased towards the benign hosts, which is the case for LR, SVM and FNN.

CHAPTER 2
LITERATURE SURVEY

2.1 Effective Botnet Detection Through Neural Networks on Convolutional Features(Shao-Chien Chen, Yi-Ruei Chen, Wen-Guey Tzeng)

ABSTRACT: Botnet is one of the major threats on the Internet for committing cybercrimes, such as DDoS attacks, stealing sensitive information, spreading spams, etc. It is a challenging issue to detect modern botnets that are continuously improving for evading detection. In this paper, we propose a machine learning based botnet detection system that is shown to be effective in identifying P2P botnets. Our approach extracts convolutional version of effective flow-based features, and trains a classification model by using a feed-forward artificial neural network. The experimental results show that the accuracy of detection using the convolutional features is better than the ones using the traditional features. It can achieve 94.7% of detection accuracy and 2.2% of false positive rate on the known P2P botnet datasets. Furthermore, our system provides an additional confidence testing for enhancing performance of botnet detection. It further classifies the network traffic of insufficient confidence in the neural network. The experiment shows that this stage can increase the detection accuracy up to 98.6% and decrease the false positive rate up to 0.5%.

2.2 An approach for host based botnet detection system

AUTHORS: Yulia ALEKSIEVA, Hristo VALCHANOV, Veneta ALEKSIEVA.

ABSTRACT: Most serious occurrence of modern malware is Botnet. Botnet is a rapidly evolving problem that is still not well understood and studied. One of the main goals for modern network security is to create adequate techniques for the detection and eventual termination of Botnet threats. The article presents an approach for implementing a host-based Intrusion Detection System for Botnet attack detection. The approach is based on a variation of a genetic algorithm to detect anomalies in a case of attacks. An implementation of the approach and experimental results are presented.

2.3 Towards using transfer learning for Botnet Detection

AUTHORS: Prapa Rattadilok, Basil Alothman

ABSTRACT: Botnet Detection has been an active research area over the last decades. Researchers have been working hard to develop effective techniques to detect Botnets. From reviewing existing approaches it can be noticed that many of them target specific Botnets. Also, many approaches try to identify any Botnet activity by analysing network traffic. They achieve this by concatenating existing Botnet datasets to obtain larger datasets, building predictive models using these datasets and then employing these models to predict whether network traffic is safe or harmful. The problem with the first approaches is that data is usually scarce and costly to obtain. By using small amounts of data, the quality of predictive models will always be questionable. On the other hand, the problem with the second approaches is that it is not always correct to concatenate datasets containing network traffic from different Botnets. Datasets can have different distributions which means they can downgrade the predictive performance of machine learning models. Our idea is

instead of concatenating datasets, we propose using transfer learning approaches tocarefully decide what data to use. Our hypothesis is “Predictive Performance can be improved by using transfer learning techniques across datasets containing network traffic from different Botnets”.

2.4 Development of an Intrusion Detection System Using a Botnet with the R Statistical Computing System

AUTHORS: Takashi Yamanoue, Junya Murakami

ABSTRACT: Development of an intrusion detection system, which tries to detect signs of technology of malware, is discussed. The system can detect signs of technology of malware such as peer to peer (P2P) communication, DDoS attack, Domain Generation Algorithm (DGA), and network scanning. The system consists of beneficial botnet and the R statistical computing system. The beneficial botnet is a group of Wiki servers, agent bots and analyzing bots. The script in a Wiki page of the Wiki server controls an agent bot or an analyzing bot. An agent bot is placed between a LAN and its gateway. It can capture every packet between hosts in the LAN and hosts behind the gateway from the LAN. An analyzing bot can be placed anywhere in the LAN or WAN if it can communicate with the Wiki server for controlling the analyzing bot. The analyzing bot has R statistical computing system and it can analyze data which is collected by agent bots.

2.5 An efficient botnet detection system for P2P botnet

AUTHORS: M. Thangapandiyan, P. M. Rubesh Anand

ABSTRACT: Peer-to-Peer (P2P) botnets are exploited by the botmasters for their resiliency against the take down efforts. As the modern botnets are stealthier, the traditional botnet detection approaches are not suitable for the botnet detection. In this paper, an efficient botnet detection system is proposed for detecting the P2P botnet. The proposed botnet detection system estimates the flow export using NetFlow protocol. The packet flow is analyzed using three main components namely, Exporter, Collector, and Analyzer. The exporter captures the packet and monitors the contents of the packet. The collector captures the flow traffic and the analyzer component initiates an automated analysis of traffic with the captured packet information. The packet flow information is collected by virtual interface and physical probe. The virtual interface is used for collecting the malicious traffic information between the Virtual Machines (VMs) and the physical probe gathers malicious traffic information between the network bridges connecting VMs. The information collected from these techniques are analyzed for detecting the botnets in inter VM and intra VM. Compared to the existing Dendritic Cell Algorithm (DCA), the proposed VM based botnet detection system has minimal time consumption, increased detection speed, and higher attack prevention ratio.

2.6 Overview of Botnet Detection Based on Machine Learning

AUTHORS: Xiaxin Dong, Jianwei Hu ,Yanpeng Cui

ABSTRACT: With the rapid development of the information industry, the applications of Internet of things, cloud computing and artificial intelligence have greatly affected people's life, and the network equipment has increased with a blowout type. At the same time, more complex network environment has also led to a more serious

network security problem. The traditional security solution becomes inefficient in the new situation. Therefore, it is an important task for the security industry to seek technical progress and improve the protection detection and protection ability of the security industry. Botnets have been one of the most important issues in many network security problems, especially in the last one or two years, and China has become one of the most endangered countries by botnets, thus the huge impact of botnets in the world has caused its detection problems to reset people's attention. This paper, based on the topic of botnet detection, focuses on the latest research achievements of botnet detection based on machine learning technology. Firstly, it expounds the application process of machine learning technology in the research of network space security, introduces the structure characteristics of botnet, and then introduces the machine learning in botnet detection. The security features of these solutions and the commonly used machine learning algorithms are emphatically analyzed and summarized. Finally, it summarizes the existing problems in the existing solutions, and the future development direction and challenges of machine learning technology in the research of network space security.

2.7 Botnet and P2P Botnet Detection Strategies: A Review

AUTHORS: Jitender Kumar , Himanshi Dhayal

ABSTRACT: Among various network attacks, botnet led attacks are considered as the most serious threats. A botnet, i.e., the network of compromised computers is able to perform large scale illegal activities such as Distributed Denial of Service attacks, click fraud, bitcoin mining etc. These attacks are considered as the major concern now-a-days. In this paper, we present a comprehensive review of botnets, their lifecycle and types. We also discuss the peer-to-peer botnet detection techniques' behaviors using various latest detection techniques.

2.8 Botnet Detection Using Recurrent Variational Autoencoder

AUTHORS: Jeeyung Kim, Alex Sim, Jinoh Kim, Kesheng Wu

ABSTRACT: Botnet detection is an active research topic as botnets are a source of many malicious activities, including distributed denial-of-service (DDoS), click-fraud, spamming, and crypto-mining attacks. However, it is getting more complicated to identify botnets due to the continuous evolution of botnet software and families that harness new types of devices and attack vectors. Recent studies employing machine learning (ML) showed improved performance to detect botnets to some extent, but they are still limited and ineffective with the lack of sequential pattern analysis, which is a key to detect various classes of botnets. In this paper, we propose a novel botnet detection method, built upon Recurrent Variational Autoencoder (RVAE), that effectively captures sequential characteristics of botnet anomalies. We validate the feasibility of the proposed method with the CTU-13 dataset that have been widely employed for botnet detection studies, and show that our method is at least comparable to existing techniques in terms of detection accuracy. In addition, our experimental results show that the proposed method can detect previously unseen botnets by utilizing sequential patterns of network traffic. We will also show how our method can detect botnets in the streaming mode, which is the essential requirement to perform real-time, on-line detection.

2.9 Sonification of Network Traffic for Detecting and Learning About Botnet Behavior

AUTHORS: Mohamed Debashi, Paul Vickers

ABSTRACT: Today's computer networks are under increasing threat from malicious activity. Botnets (networks of remotely controlled computers, or “bots”) operate in such a way that their activity superficially resembles normal network traffic which makes their behavior hard to detect by current intrusion detection systems (IDS). Therefore, new monitoring techniques are needed to enable network operators to detect botnet activity quickly and in real time. Here, we show a sonification technique using the SoNSTAR system that maps characteristics of network traffic to a real-time soundscape enabling an operator to hear and detect botnet activity. A case study demonstrated how using traffic log files alongside the interactive SoNSTAR system enabled the identification of new traffic patterns characteristic of botnet behavior and subsequently the effective targeting and real-time detection of botnet activity by a human operator. An experiment using the 11.39 GiB ISOT botnet data set, containing labeled botnet traffic data, compared the SoNSTAR system with three leading machine learning-based traffic classifiers in a botnet activity detection test. SoNSTAR demonstrated greater accuracy (99.92%), precision (97.1%), and recall (99.5%) and much lower false positive rates (0.007%) than the other techniques. The knowledge generated about characteristic botnet behaviors could be used in the development of future IDSs.

2.10 Analysis of Machine Learning Algorithms for IoT Botnet

AUTHOR: Umang Garg, Vaibhav Kaushik, Anushka Panwar, Neha Gupta

ABSTRACT: The Internet of Things (IoT) gains a lot of popularity day-by-day due to their everlasting availability and ease. As the popularity of IoT increases, it also attracts hackers which try to take advantage of the vulnerability of IoT devices. An Intrusion Detection System (IDS) is an intelligence-based system that can investigate or detect the intrusion in the IoT botnet and check the state of software and hardware executing in the network. Once the intrusion is detected, it may generate an alarm to alert the administrator or send some alert message to the owner. In the last decade, there are several IDSs available which can detect the intrusion. But the major problems with the existing IDSs like accuracy rate, generation of the false alarm, and fewer chances of detection of unknown attacks. To deal with the above problems, some machine learning techniques have been involved by researchers. These techniques can differentiate between the normal and abnormal behavior of the user's data or network traffic with high accuracy. In this paper, we summarize and classify the machine learning algorithms that can be used in IDS with their metrics, parameters. Then, a case study is implemented with the UNSW-NB15 dataset that has realistic network traffic with frequently used machine learning techniques. After that, a comparison will be done and displayed by using an accuracy percentage table and a bar chart. Finally, some challenges and future scope of the machine learning techniques in the improvement of IDS will be discussed.

2.11 An enhancing framework for botnet detection using generative adversarial networks

AUTHORS: Chuanlong Yin, Yuefei Zhu, Shengli Liu, Jinlong Fei, Hetong Zhang

ABSTRACT: The botnet, as one of the most formidable threats to cyber security, is often used to launch large-scale attack sabotage. How to accurately identify the botnet, especially to improve the performance of the detection model, is a key technical issue. In this paper, we propose a framework based on generative adversarial networks to augment botnet detection models (Bot-GAN). Moreover, we explore the performance of the proposed framework based on flows. The experimental results show that Bot-GAN is suitable for augmenting the original detection model. Compared with the original detection model, the proposed approach improves the detection performance, and decreases the false positive rate, which provides an effective method for improving the detection performance. In addition, it also retains the primary characteristics of the original detection model, which does not care about the network payload information, and has the ability to detect novel botnets and others using encryption or proprietary protocols.

2.12 A Survey on Botnet Detection Techniques

AUTHOR: Shivani Gaonkar, Nandini Fal Dessai, Jenny Costa

ABSTRACT: Due to the increased rate of internet usage, security problems have also increased. One of the serious threats in network security are Botnets. A Botnet is defined as a collection of various bots that Botmaster controls through the Command and Control (C&C) channel. During recent times, different technologies and techniques have been proposed to track the detection of botnets. This paper summarizes different techniques to detect different botnets. General bot detection and IoT-bot detection techniques are separately explained. UNSW-NB15 datasets have been used in training and testing of the proposed model. A real-time IoT-Bot detection using deep learning algorithm is proposed in this paper. Wireshark is used to capture a package from network traffic.

2.13 Analysis of Botnet Domain Names for IoT Cybersecurity

AUTHOR: Wanting Li, Jian Jin, Jong-Hyouk Lee

ABSTRACT: Botnets are widespread nowadays with the expansion of the Internet and commonly occur in many cyber-attacks, resulting in serious threats to network services and users' properties. With the rapid development of the Internet of Things (IoT) applications, the botnet can easily make use of IoT devices for larger-scale attacks. Domain name system (DNS) is widely used by the botnet to establish the connection between bots and their corresponding command-and-control (C&C). In order to avoid the track of the C&C through the DNS information, some sophisticated schemes are used by the botnet and fast-flux is a typical one. In this paper, the activities of Rustock botnet domain names which just use the fast-flux as the connection method between bots and C&C, are deeply analyzed from multiple aspects. Besides, we extract 32 special features of Rustock domain named querying traffic. Then multiple popular classifiers are adopted in order to pick the malicious domain names out from the DNS traffic using those 32 features. The work of this paper aims to provide guidance for future botnet detection based on real statics and experiments.

2.14 Email Shape Analysis for Spam Botnet Detection

AUTHOR: Paul Sroufe, Santi Phithakkitnukoon, Ram Dantu, Joao Cangussu

ABSTRACT: Botnets have become the major sources of spamming, which generates massive unwanted traffic on networks. An effective detection mechanism can greatly mitigate the problem. In this paper, we present a novel botnet detection mechanism based on the email "shape" analysis that relies on neither content nor reputation analysis. Shape is our new way of characterizing an email by mimicking human visual inspection. A set of email shapes are derived and then used to generate a botnet signature. Our preliminary results show greater than 80% classification accuracy (without considering email content or reputation analysis). This work investigates the discriminatory power of email shape, for which we believe will be a significant complement to other existing techniques such as a network behavior analysis.

2.15 Bot Detection via IoT Environment

AUTHOR: Im Y. Jung, Jae J. Jang, Jae-geun Moon

Abstract: Many users do not realize whether their devices become bots or not. There are many security accidents due to malicious bots. To solve this problem, we propose a monitor system composed of IoT devices to detect bots.

2.16 Detection Method of DNS-based Botnet Communication Using Obtained NS Record History

AUTHOR: Katsuyoshi Iida, Yong Jin, Hikaru Ichise

ABSTRACT: To combat with botnet, early detection of the botnet communication and fast identification of the bot-infected PCs is very important for network administrators. However, in DNS protocol, which appears to have been used for botnet communication recently, it is difficult to differentiate the ordinary domain name resolution and suspicious communication. Our key idea is that the most of domain name resolutions first obtain the corresponding NS (Name Server) record from authoritative name servers in the Internet, whereas suspicious communication may omit the procedures to hide their malicious activities. Based on this observation, we propose a detection method of DNS basis botnet communication using obtained NS record history. Our proposed method checks whether the destined name server (IP address) of a DNS query is included in the obtained NS record history to detect the botnet communications

2.17 Botnet detection using software defined networking

AUTHOR: Udaya Wijesinghe, Udaya Tupakula, Vijay Varadharajan

ABSTRACT: Software Defined Networking (SDN) is considered as a new approach promising simplified network management by providing a programmable interface. The idea of SDN is based on the separation of control plane from the data plane in networking devices. This is achieved by having the network intelligence centralised in what is called as SDN controller. In this paper we propose techniques for botnet detection in networks using SDN. The SDN controller makes use of generic templates for capturing the traffic flow information from the OpenFlow switches and makes use of this information for detecting bots. We will show that our model can detect a range of bots including IRC, HTTP and peer-to-peer bots.

2.18 DGA Bot Detection with Time Series Decision Trees

AUTHOR: Anaël Bonneton, Daniel Migault, Stephane Senecal, Nizar Kheir

ABSTRACT: This paper introduces a behavioral model for botnet detection that leverages the Domain Name System (DNS) traffic in large Internet Service Provider (ISP) networks. More particularly, we are interested in botnets that locate and connect to their command and control servers thanks to Domain Generation Algorithms (DGAs). We demonstrate that the DNS traffic generated by hosts belonging to a DGA botnet exhibits discriminative temporal patterns. We show how to build decision tree classifiers to recognize these patterns in very little computation time. The main contribution of this paper is to consider whole time series to represent the temporal behavior of hosts instead of aggregated values computed from the time series. Our experiments are carried out on real world DNS traffic collected from a large ISP.

2.19 An analysis of network traffic classification for botnet detection

AUTHOR: Matija Stevanovic, Jens Myrup Pedersen

ABSTRACT: Botnets represent one of the most serious threats to the Internet security today. This paper explores how network traffic classification can be used for accurate and efficient identification of botnet network activity at local and enterprise networks. The paper examines the effectiveness of detecting botnet network traffic using three methods that target protocols widely considered as the main carriers of botnet Command and Control (C&C) and attack traffic, i.e. TCP, UDP and DNS. We propose three traffic classification methods based on capable Random Forests classifier. The proposed methods have been evaluated through the series of experiments using traffic traces originating from 40 different bot samples and diverse non-malicious applications. The evaluation indicates accurate and time-efficient classification of botnet traffic for all three protocols. The future work will be devoted to the optimization of traffic analysis and the correlation of findings from the three analysis methods in order to identify compromised hosts within the network.

2.20 Botnet Domain Name Detection based on machine learning

AUTHOR: Baoping Yan, Guanggang Geng, Zhiwei Yan, Jian Jin

ABSTRACT: Domain Name System (DNS) is a fundamental component of today's Internet: it provides mappings between domain names used by people and the corresponding IP addresses required by network protocols. However, the open and fundamental characteristics of DNS are recently used by the botnet for the communication between bots and C&C. In this paper, we select six kinds of special features of botnet domain querying traffic based on the deep studies of the DNS log. Then three popular classifiers are adopted in order to pick the malicious domains outfrom the DNS traffic using those features.

CHAPTER 3
METHODOLOGY

3.1 EXISTING SYSTEM

In existing flow-based approaches typically incur a high computational overhead and do not completely capture the network communication patterns, which can expose additional aspects of malicious hosts. Recently, bot detection systems that leverage communication graph analysis using ML have gained attention to overcome these limitations. A graph-based approach is rather intuitive, as graphs are true representation of network communications.

3.1.1 EXISTING SYSTEM DISADVANTAGE

•Do not completely capture the network communication patterns, which can expose additional aspects of malicious hosts.

3.2 PROPOSED WORK

In this paper, we propose BotChase, an anomaly-, graph-based bot detection system, which is protocol agnostic, i.e., it detects bots regardless of the protocol. BotChase employs graph-based features in a two phased ML approach, which is robust to zero-day attacks, spatially stable, and suitable for large datasets.We evaluate the BotChase prototype system in an online setting that recurrently trains and tests the ML models with new data. We also leverage the Hoeffding Adaptive Tree (HAT) classifier for incremental learning. This is crucial to account for changes in network traffic and host behavior.

Cyberattacks are on the rise these days. Many systems are getting infected by attacks to overcome these attacks, In the past, we used signature-based research. However, as technology developed, attacks became more sophisticated and we used k-means and decision trees to see how many bots were targeted and how many were not. If there is an attack, we will find how many bots were attacked or detected and we will give the number.

LIMITATIONS

Botnet detection has been an active area of research that has generated a substantial body of work. Common botnet detection approaches are passive. They assume successful intrusions and focus on identifying infected hosts (bots) or detecting C2 communications, by analyzing system logs and network data, using signature- or anomaly-based techniques. Signature-based techniques have commonly been used to detect pre-computed hashes of existing malware in hosts and/or network traffic. They are also used to isolate IRC-based bots by detecting bot-

like IRC nicknames and to identify C2-related DNS requests by detecting C2-like domain names. Metadata such as regular expressions based on packet content and target IP occurrence tuples is an example of what could be employed in a signature and pattern detection algorithm. More generally, signature-based techniques have been employed to identify C2 by comparison with known C2 communication patterns extracted from observed C2 traffic, and infected hosts by comparison with static profiles and behaviours of known bots.

In the application CTU-13 dataset is used form kaggle Upload ctu-13 dataset button ,it open the files.There we select the dataset click on open. After uploading the dataset on screen it display the path from where we are taking dataset , dataset size Also displays total rows and total columns, showing the Start Time, Duration, Protoc , SrcAddrress,Sport,Dire,DstAddress,Dport,State,sTos,dTos,TotalPackets,TotalBytes, SrcBytes,Label and also the rows and columns in side square braces.

Apply k-means to separate bot and benign data from the data set. It gives us the dataset size before removing benign records, i.e (total rows and columns). gives the dataset size after removing the benign records, i.e (total rows and columns) By using k-means we separated the Bot and Benign data.

When we have a look at the CMD there it show as generated bot graph points On UI it shoes the number of nodes , number of edges, number of graph created , between-Ness centrality for all IP address or node. Here ip address nothing but nodes, Execution time, clustering time calculation, alpha centrality time calculation Alpha Centrality time.

After clicking on it, Normalizing features process completed & below are some sample records out out- degree-weight in-degree-wt outdegree ,indegree bot bc lcc ac. All the values of it which are normalized, Normalized & transformed data saved inside normalize_data. csv file, as well as we can have a look at the CMD there it show as features normalization module 100 percent done and shoes the record in it.

It shows Normalized data loading to decision tree classifier Total dataset size to build model.Model training records size, Model testing records size, Decision Tree Accuracy , Decision Tree Precision , Decision Tree Recall ,True -Pos , False-Pos,True-Neg, False-Neg.we have test 20 % of data, and training 80% of data .
The Accuracy of this model is 99%.

3.3 SUPERVISED LEARNING

A method of teaching machine learning labeled data by hand is called supervised learning. its already know output of the algorithm before it start working on it, example classifying a dataset in CTU-13, here it matches the input to output, here we will train

the data and and tested the data , once algorithm is well trained, it is tested using the new data when it comes to unsupervised learning the training phase is big because the machine is only given the input,it has to figure out the output on its own, so there is no supervisor here or there's is no mentor over here.

3.4 UNSUPERVISED LEARNING

Unsupervised learning involves the machine learning without any guidance in the form of unlabeled data.Here it forms as groups for example in this project like attack and non-attack, the only difference is it Cant add the labels, it understands how the cluster groups separate .Types of problems: Association ,clustering: separating on based Anomaly The detection of unusual activities can be used for detecting suspicious activity and the reinforcement of these activities is what we call reinforcement learning now.In unsupervised learning we must find patterns in data and keep exploring the data until it reaches the output. Observing patterns and extracting insights in unsupervised approaches is all about figuring out how to get the output, since the algorithm is only given input, it must find ways to gain insights from data by finding trends and associations, mapping the known input to known outputs.

3.4.1 K-means

It's a technique most of us do in our daily life, for example like group of people sharing tableClustering is the process of dispersing datasets into groups consisting of similar data points. For example: k-means clustering. Exclusive clustering is hard clustering, where points/items belong only to one cluster.

Descion tree: (supervised)

Descion tree it can be used as both supervised and unsupervised, but in this project we are using decision tree as supervised algorithm. It has a root that grows as a number of different options is increased, similar to decision trees. A decision tree is a visual representation of all possible solutions based on many conditions. and the condition now, here we will split the dataset into different subsets will become the input to child, the goal is produce the purest possible distribution of the labels at each nodes

In this project we are using k-means and desicion tree algorithms for building this projecte.

To execute the project we have to click on run , then the CMD opens which shows the path of project where it located, after that the user interface opens, splits of 2 screens one screen contains buttons Other side it shows the executed functions output.

CHAPTER

A. Upload CTU Dataset

B. Apply KMEANS to separate Bot & Benign Data

C. Run Flow Ingestion & Graph Transformation
D. Features Extraction & Normalization
E. Run Decision Tree Algorithm
F. View Graph
G. Exit

First open the application then run in the command prompt User Interface (UI) is displayed. On UI you will have some buttons like Upload CTU Dataset, Apply KMEANS to separate Bot & Benign Data, Run Flow Ingestion & Graph Transformation, Features Extraction & Normalization, Run Decision Tree Algorithm, View Graph, Exit. Click on the first button i.e Upload CTU Dataset, then some datasets are displayed. Select one among them and click open. It gives the dataset size like total rows, total columns and also dataset samples. Then click on the second button i.e Apply KMEANS to separate Bot & Benign Data. It gives you the information about dataset size before removing the benign records and after removing the benign records. This button will apply k means algorithm to the dataset and separate as two clusters namely bot and benign and will remove the benign records from the set. Then click on the third button i.e Run Flow Ingestion & Graph Transformation. It gives the information like number of nodes, number of edges and betweenness. Then click on the the fourth button Features Extraction & Normalization. It complete the Normalizing features process and display some sample records. Then click on the Run Decision Tree Algorithm. It display information like Decision Tree Accuracy, Decision Tree Precision, Decision Tree Recall, True Positive, False Positive, True Negative, False Negative. Then you can select number of nodes to draw graph. After selecting the nodes you can click on View Graph to display the graph. The graph displays the cluster. Last you will find a exit button to exit from the UI. Click on the exit to close the interface. If you want to find botnet attacks from other datasets, then you can again upload a new dataset in the upload button and repeat the steps like applying k means, then click Run Flow Ingestion & Graph Transformation, then feature extraction and normalization and apply run decision tree algorithm.

3.5 Advantage

•It also accommodates different network topologies and is suitable for large-scale data.
•Compared to the state-of-the-art, BotChase outperforms an end-to-end system that employs flow-based features and performs particularly well in an online setting.

3.6 SOFTWARE AND HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS
The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

•Python idel 3.7 version (or)
•Anaconda 3.7 (or)
•Jupiter (or)
•Google colab

HARDWARE REQUIREMENTS
Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

•Operating system: windows, linux
•Processor: minimum intel i3
•Ram
•Hard disk : minimum 250gb

3.7SYSTEM STUDY

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

3.7.1 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.

3.7.2 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.

3.7.3 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.

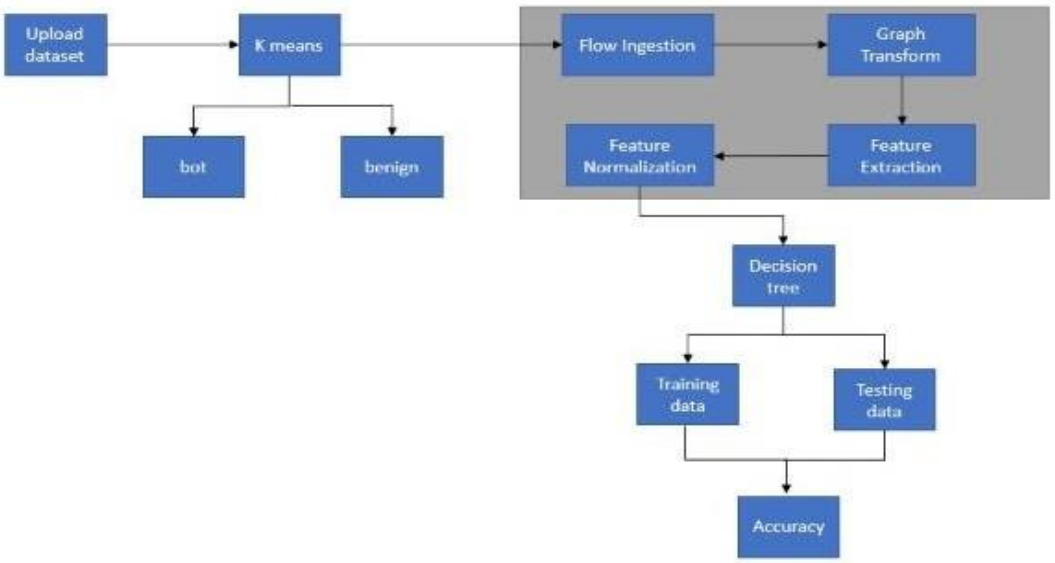


Fig 1 .SYSTEM ARCHITECTURE

3.8 DATA FLOW DIAGRAM:

1.The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2.The data flow diagram (DFD) is one of the most important modeling 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.
3.DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4.DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

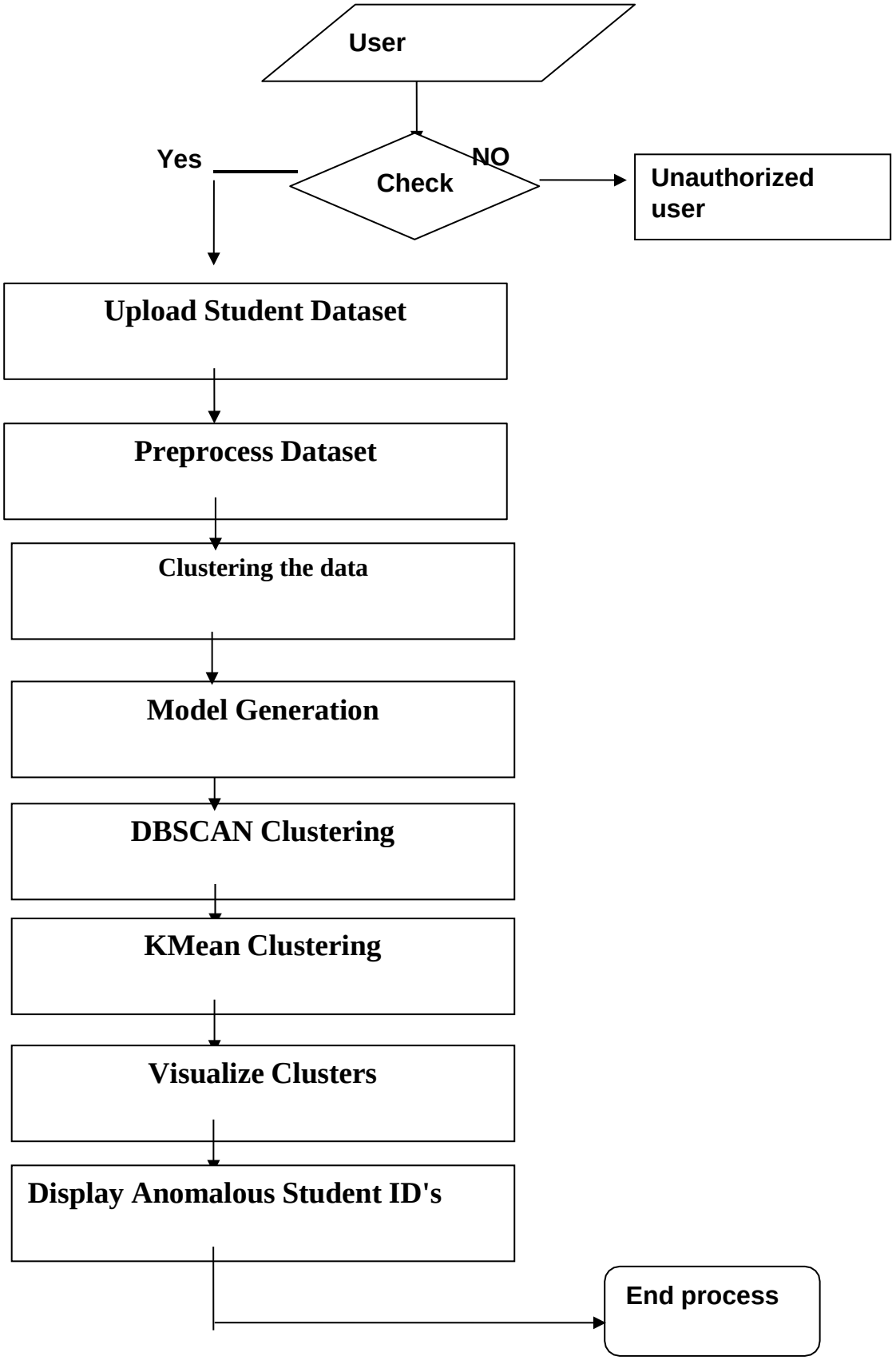


Fig 2 **DATA FLOW DIAGRAM:**

UML DIAGRAMS

3.8.1 Introduction To UML

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, 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. The 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. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

3.8.2 Goals of UML

. The primary goals in the design of the UML were:

•Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.

•Provide extensibility 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.

Why we use UML?

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.

3.9 UML Diagram

The underlying premise of UML is that no one diagram can capture the different elements of a system in its entirety. Hence, UML is made up of nine diagrams that can be used to model a system at different points of time in the software life cycle of a system.

The nine UML diagrams are:

3.9.1 Use case diagram:

The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The use case diagram shows which actors interact with each use case.

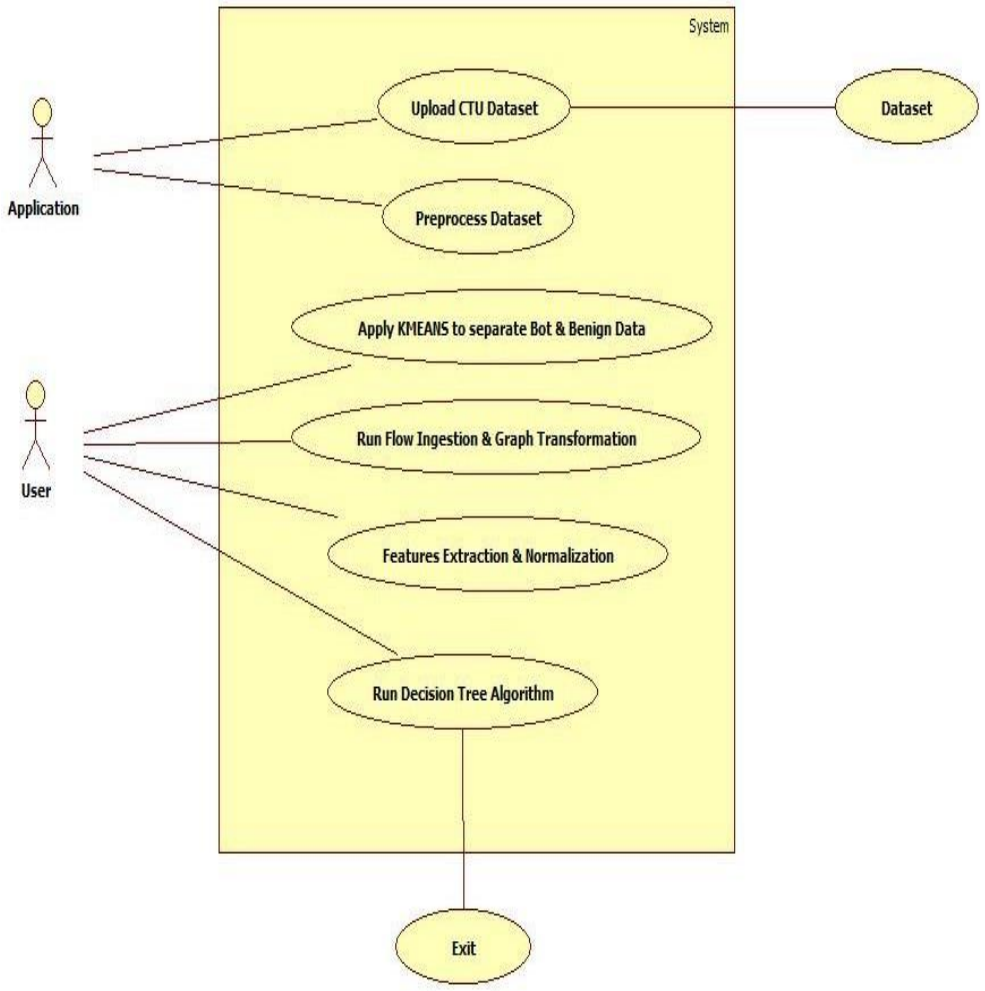


Fig 3: Use case diagram

3.9.2 Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

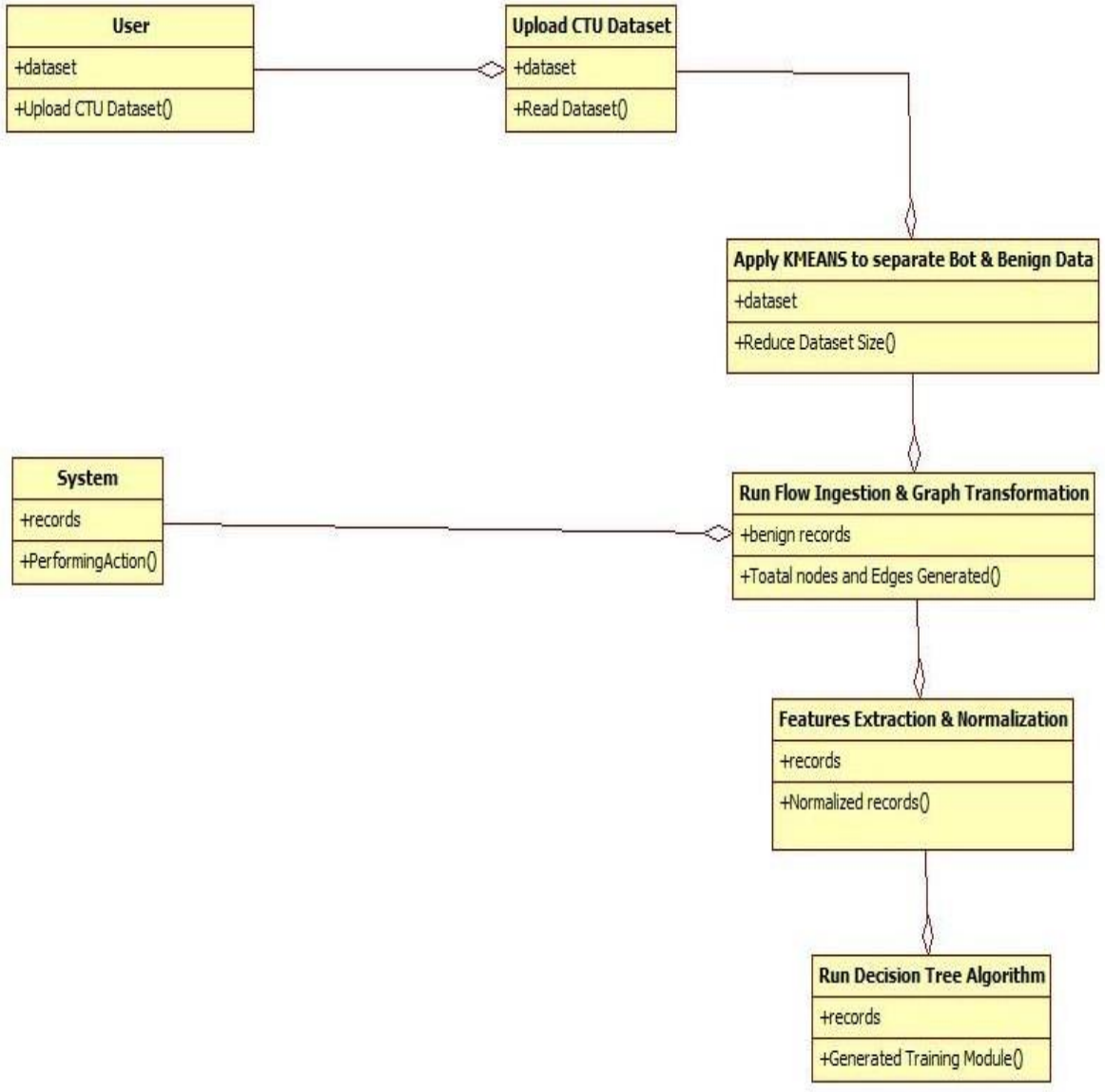


FIG.4 CLASS DIAGRAM

3.9.3 Object diagram:

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.

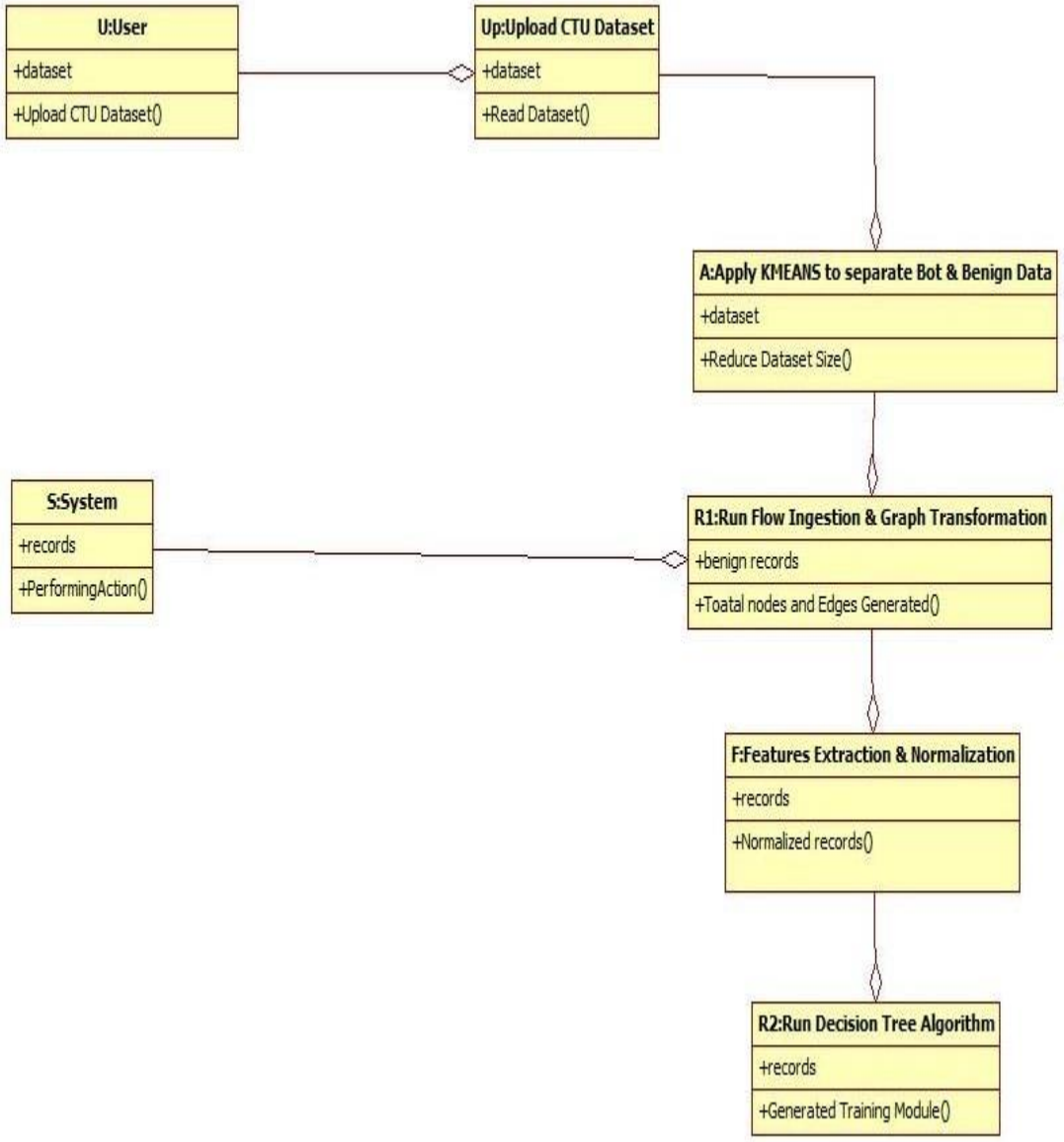


FIG 5:OBJECT DIAGRAM

3.9.4 State diagram:

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

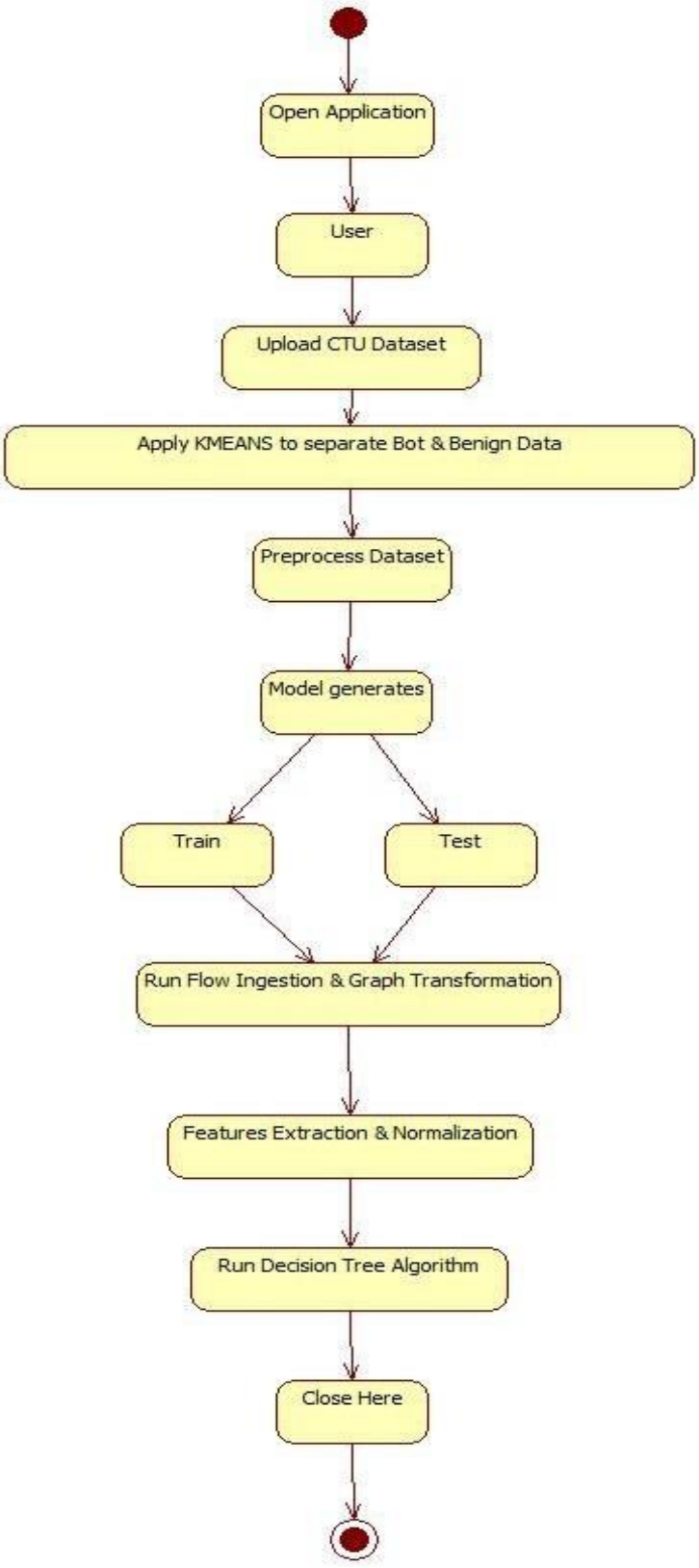


FIG 6:STATE DIAGRAM

3.9.5 Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

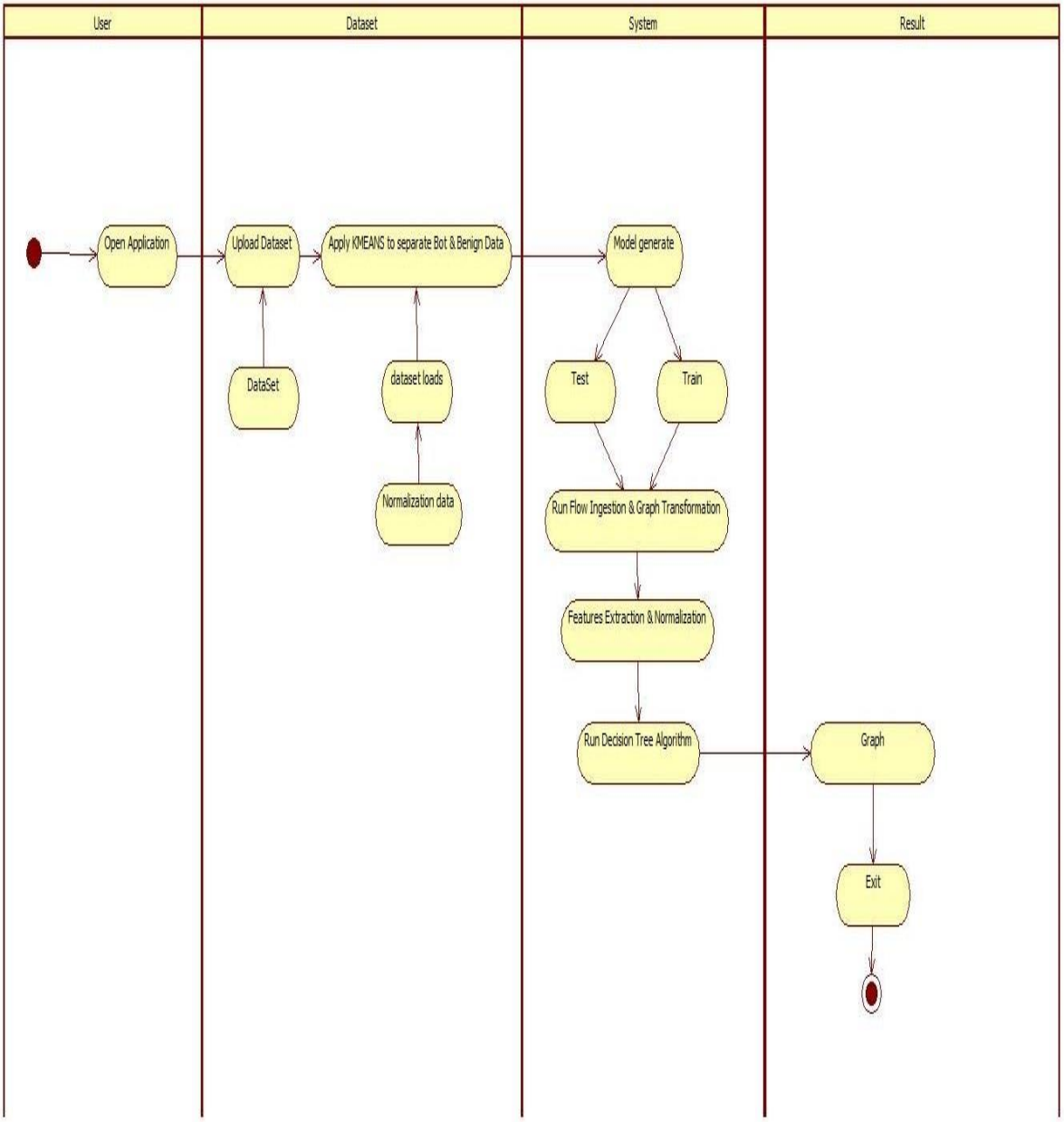


FIG 7:ACTIVITY DIAGRAM

3.9.6 Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

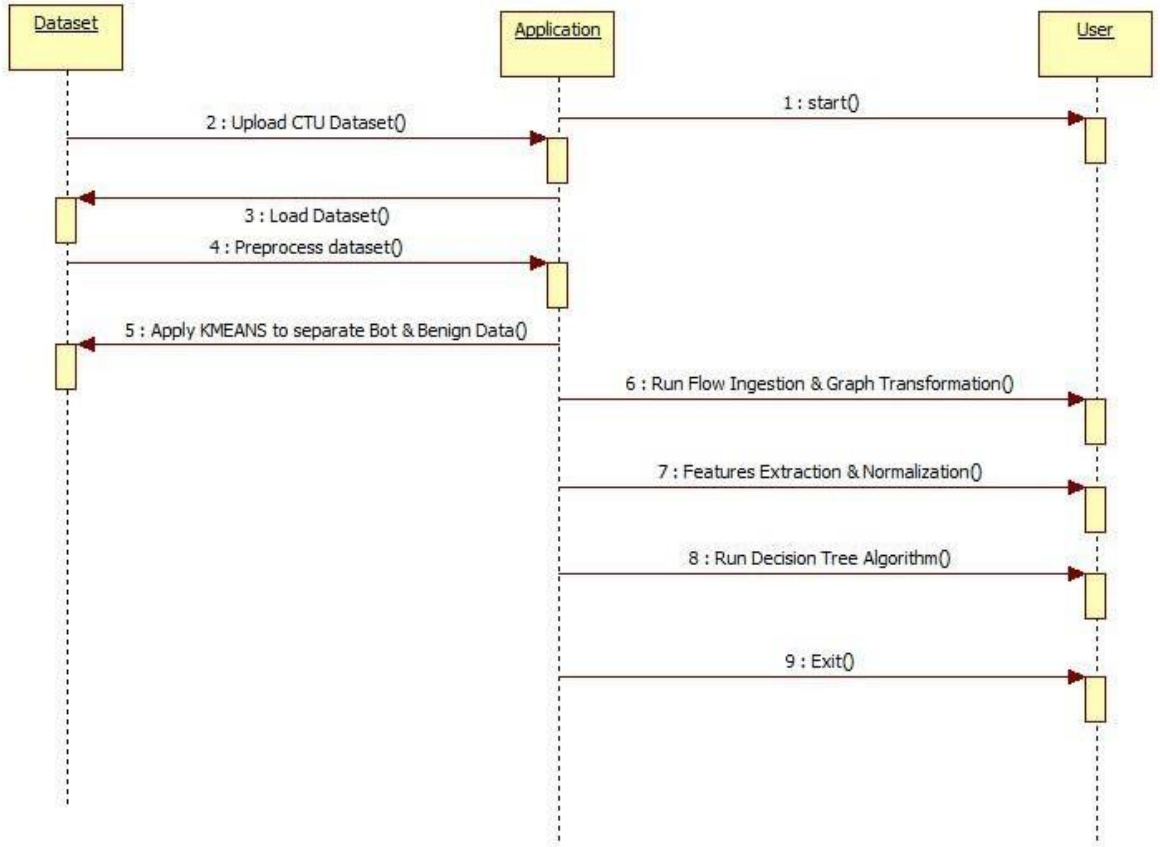


FIG 8:SEQUENCE DIAGRAM

3.9.7 Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

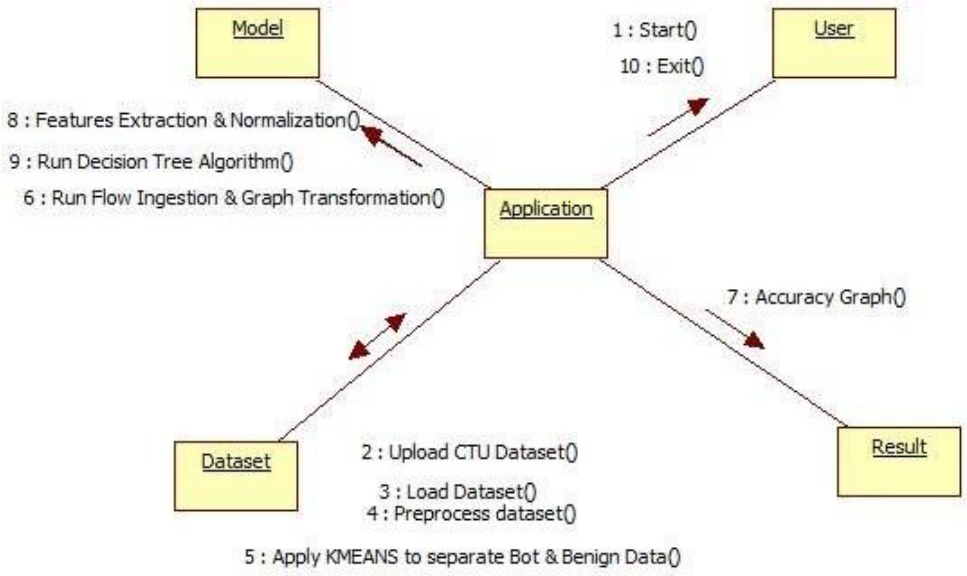


FIG 9:COLLABORATION DIAGRAM

3.9.8 Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.

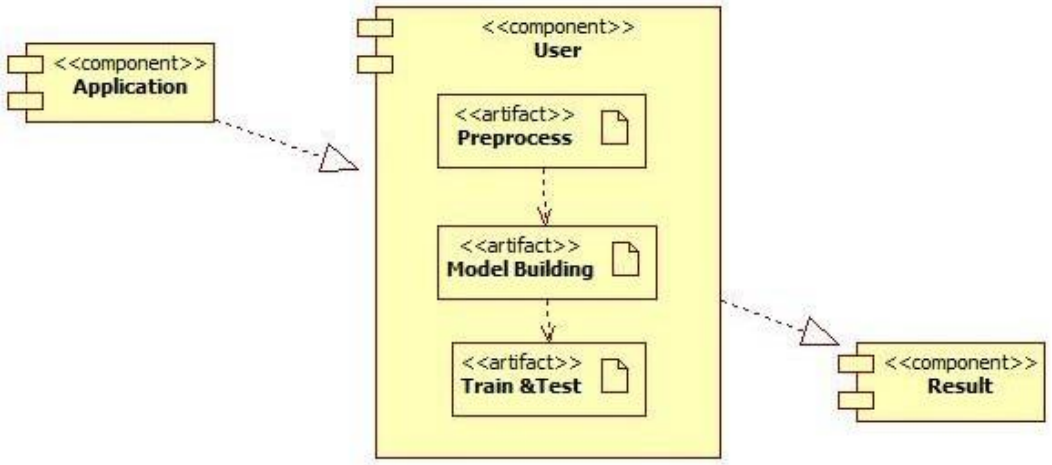


FIG 10: COMPONENT DIAGRAM

3.9.9 Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.

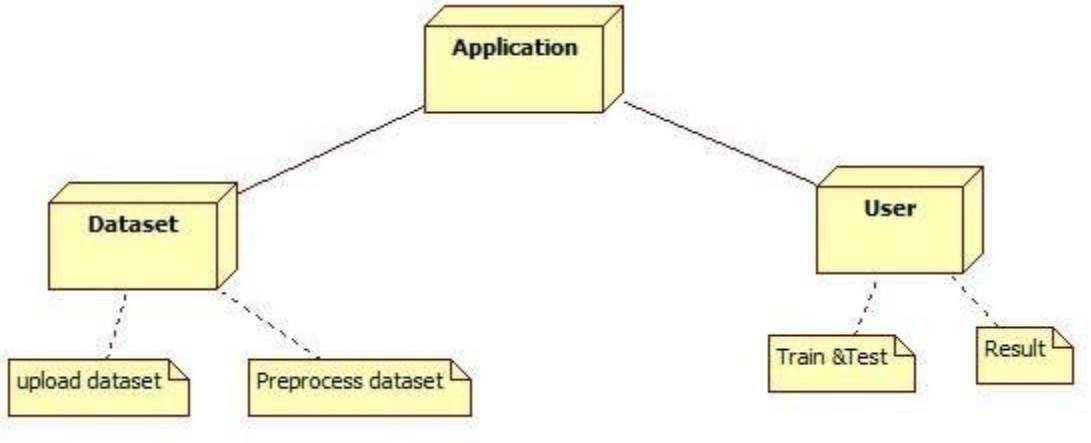


FIG 11: DEPLOYMENT DIAGRAM

3.10 Modules:

- Upload CTU Dataset
- Apply KMEANS to separate Bot & Benign Data
- Run Flow Ingestion & Graph Transformation
- Features Extraction & Normalization
- Run Decision Tree Algorithm
- Exit

A .Uploading ctu-13 datset

Upload ctu-13 dataset button ,it open the files there we select the dataset click on open, after uploading the dataset on screen it shows the path from where we are taking dataset , dataset size, by mentioning total rows and total columns, and showing the StartTime, Duration, Protoc ,Srcorce-Addrress ,Sport,Dire,DstAddress,Dport,State,sTos,dTos,TotalPackets,TotalBytes,SrurceBytes, Label and also the rows and columns in side square braces.

B. Apply k-means to separate bot and benign data:

Apply k-means to separate bot and benign data from the data set , it gives us the dataset size before removing benign records total rows and columns, and also it gives the dataset size after removing the benign records total rows and columns by using k-means we separate there data.

C. RunFlow Integration and graph transformation

After clicking on run flow integration it shoes two screens extract which we need to close ,when we have a look at the CMD there it show as generated bot graph points , on ui it shoes the number nodes , number of edges, number of graph created , between-Ness centrality for all IP address or node , here ip address nothing but nodes, Execution time, clustering time calculation, alpha centrality time calculation Alpha Centrality time.

D.Features Extraction and normalization:

After clicking on it, Normalizing features process completed & below are some sample records out out- degree-weight in-degree-wt outdegree ,indegree bot bc lcc ac, all the values of it which are normalized, Normalized & transformed data saved inside normalize_data.csv file, as well as we can have a look at the CMD there it show as features normalization module 100 percent done and shoes the record in it.

E. Run Decision Tree Algorithm

It shows Normalized data loading to decision tree classifier, Total dataset size to build model, Model training records size, Model testing records size, Decision Tree Accuracy , Decision Tree Precision ,Decision Tree Recall ,True -Positive ,False-Positive ,True-Negative , False-Negative . we have test 20 % of data, and training 80% of data .The Accuracy of this model is 99%.

F. View Graph

In the final module there will be input textbox where we can enter some number into it , so that it generate the graph after clicking on the view graph.it pop up another screen shoes all the ip address and its connections. After completing the whole project clicking on exit we exit from the GUI interface.

G.EXIT

Clicking on exit button we will exit from GUI interface.come out of project.

3.10.1 Algorithm:

k-Means, Density-Based Spatial Clustering (DBScan) and SOM, Decision Tree, Feed-forward Neural Network (FNN), Logistic Regression (LR) and Support Vector Machine (SVM).

• **k-Means**—The k-Means clustering algorithm attempts to find an optimal assignment of nodes to k pre-determined clusters, such that the sum of the pairwise distance from the cluster mean is minimized. k-Means is static, it results in the same cluster composition for a given dataset across different runs of the algorithm, with the same number of clusters and iterations. Assume k is set to the cardinality of the label set. Idealistically, there should be a clean assignment of hosts to corresponding clusters.

However, in reality, some benign hosts exhibit an outlier behavior. For example, network nodes that host webservers and public APIs will depict a huge amount of data and connections, thus impacting ID, IDW, OD and ODW. Therefore, depending on the dataset, altering k may adversely affect clustering performance.

• Density-Based Spatial Clustering (DBScan)—Unlike kMeans, DBScan does not require the parameter k, the predetermined number of clusters. In contrast, it computes the clusters and assignment of nodes according to a rigid set of density-based rules. DBScan requires a pair of parameters: (i) p, the minimum number of points required to be assigned as core points, and (ii) e, the minimum distance required to detect points as neighbors. DBScan classifies points as core, edge or noise, where core points must have p points in their neighborhood with a distance less than e. Otherwise, if the point is reachable via e distance from at least one of the core points, it is considered an edge. The remaining points are considered noise and are not clustered. That is, points are not forcefully assigned to clusters as some points may just be noise. Therefore, DBScan is capable of detecting non-linearly separable clusters.

• Self-Organizing Map (SOM)—A SOM is a special purpose artificial neural network that applies competitive learning instead of error-correction. It is frequently used for dimensionality reduction and clusters similar data. However, the notion of similarity in SOM is looser than that of k-Means and DBScan. In SOM, neurons are pushed towards the data points for a certain number of iterations. It uses the best matching unit to determine the winner neuron and updates its weights accordingly. Furthermore, SOMs also apply a learning radius that affects all the other neurons, when a close-by neuron is updated. The number of neurons also play an important role in clustering. Higher number of neurons result in dispersion of nodes away from a single cluster. Importantly, the same logic applies to k-Means, hence the classifier with the best assignment must be selected, according to the objectives outlined in this phase. 2) Phase 2: Phase 1 separates the dataset between nodes that are inside and outside the benign cluster. All the nodes, ideally small, that reside outside the benign cluster are input to Phase 2 for further classification. Optimally, all the bots

should be outside the benign cluster, regardless of whether or not they are co-located in the same cluster. Depending on the amount of hosts outside the benign cluster, the supervised learning (SL) classifiers used in this phase will exhibit different results. The primary objective in this phase is to maximize recall. Recall is a measure of how many bots are recalled correctly i.e., do not go unnoticed. It is proportional to the number of true positives (TPs) and inversely proportional to false negatives (FNs). Various SL classifiers can be deployed in this phase to achieve this objective, such as logistic regression (LR), support vector machine (SVM), feed-forward neural network (FNN) and decision tree (DT).

• Logistic Regression (LR) and Support Vector Machine (SVM)—LR focuses on binary classification of its input, based on a sigmoid function. Input features are coupled with corresponding weights and fed into the function. Once a threshold p is defined, usually 0.5 for the logistic function, it establishes the differentiator between positive and negative points. Unlike LR, SVM is a non-probabilistic model for classification. It is not restricted to linearly separable datasets. There are various methods of computing SVM, including the renowned gradient-descent algorithm.

• Feed-forward Neural Network (FNN)—FNNs are artificial neural networks that do not contain any cyclic dependencies. For a given feed-forward network with multiple layers, a feature vector is dispersed into the input layer, fed to the hidden layer of the network, and then to its output layer. While the input layer is constrained by the number of features exposed, the hidden and output layers are not. Every neuron may rely on a separate activation function that shapes the output. Popular activation functions for FNNs include identity, sigmoid, ReLU and binary step, among others. FNNs and the previously mentioned SL techniques are online classifiers. An online classifier is capable of incremental learning, as the weights associated with the deployed perceptrons are not static. This makes FNNs an attractive candidate for production-grade deployment.

• Decision Tree (DT)—DTs rely heavily on information entropy (IE) and gain to conjure its conditional routing procedure. Generally, IE states how many bits are needed to represent certain stochastic information in the dataset. By using DT, information gain is maximized from the observed data and the taken path. After training a DT, newly observed data points can be predicted. However, unlike all the other classifiers, DTs are not online. That is, optimally retraining a DT must be done from scratch. Recall the objective from Phase 1 i.e., minimize hosts outside the benign cluster (HOB), while maximizing bots outside the benign cluster (BOB). This results in a minimal training dataset for Phase 2. Also, it is expected that the resultant training dataset from Phase 1 would be unbalanced, with a bias towards benign hosts. This may prove problematic for LR, SVM and FNN in achieving high recall rates.

CHAPTER 4
RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

The aim of this paper is to develop a user interface which can detect the Botnet records based on graph. This application will detect Botnet records in the internet connected system by using Machine learning algorithms and also detect the newly attacks based on the graph which is plotted using the k means algorithm. Where as k means is an unsupervised learning algorithm it will detect the newly created attacks by the distance formula

The internet connected device owner can provide security to their systems by our User Interface.

CHAPTER 5
SUMMARY AND CONCLUSIONS

In this paper, we propose Botnet detection, a system that is capable of efficiently transforming network flows into an aggregated graph model. It leverages two ML phases to differentiate bots from benign hosts. Botnet allows you to combination community flows into graphical version based on network flow facts. In the primary phase, SOM is used to make sure an Maximizing the benign clusters but maintaining an acceptable compromise while alienating the malicious bots. Additionally, the

consequences show high TPs ,coffee FPs for DT. Without the F Norm, the effects of the SOM have been made worse, i.E., fewer bots within the normal (bengin) cluster, and the size of the benign cluster reduced. In addition to detecting bots that use one of a kind protocols, BotChase is also capable to educate and infer ML fashions for pass-network ML education is attacked by go-community. Graph-based totally capabilities outperform go with the flow-based features in BotChase. Further, BotChase outperforms an quit-to-cease device that is predicated on float-based capabilities and compares favorably with the graph-based Bot detection. BotChase, in web-primarily based surroundings, applies incremental learning using HAT. FNorm requires longer to converge, however the model performs extremely nicely in its very last country. Future research consciousness on tuning the classifiers, investigating superior ensemble gaining knowledge of and feature engineering strategies, and increasing FNorm to better degrees.

REFERENCES

Textbooks:

1.Programming Python, Mark Lutz
2.Head First Python, Paul Barry
3.Core Python Programming, R. Nageswara Rao
4. Learning with Python, Allen B. Downey

Journals:

[1]. Jay N. Paranjape ., Misha Mehra ., Jay N. Paranjape ., Vinay Joseph Ribeiro., " Improving ML Detection of IoT Botnets using Comprehensive Data and Feature Sets "., 2021.

[2]. Abdallah Moubayed ., MohammadNoor Injadat ., Abdallah Shami ., " Optimized Random Forest Model for Botnet Detection Based on DNS Queries " ., 2021.

[3]. Mrutyunjaya Panda ., Abd Allah A. Mousa, Aboul Ella Hassanien ., " Developing an Efficient Feature Engineering and Machine Learning Model for Detecting IoT-Botnet Cyber Attacks " ., 2021.

[4]. Kostas E. Psannis ., Vasileios A. Memos ., " AI- Powered Honeypots for Enhanced – IoT Botnet Detection " .,2020.

5] Sina Hojjatinia ., Hadis Mohseni ., Sajad Hamzenejadi ., " Android Botnet Detection using Convolutional Neural Networks ", 2020.

[6]. Paul D. Yoo, Sami Muhaidat, Omar Y. Al-Jarrah ., Omar Alhussein ., Kwangjo Kim., , Kamal Taha ., " Data Randomization and Cluster- Based Partitioning for Botnet Intrusion Detection " ., 2015.

[7]. Duc C. Le ., Nur Zincir-Heywood ., " Learning From Evolving Network Data for Dependable Botnet Detection "., 2020.

[8]. Khalid Alsubhi., Afnan Alharbi ., Khalid Alsubhi., " Botnet Detection Approach Using Graph-Based Machine Learning "., 2021.

[9] S. Sriram ., Mamoun Alazab ., R. Vinayakumar, .,Soman KP " Network Flow based IoT Botnet Attack Detection using Deep Learning " ., 2020.

[10] Abdallah Moubayed ., MohammadNoor Injadat ., Abdallah Shami ., " Detecting Botnet Attacks in IoT Environments: An Optimized Machine Learning Approach " ., 2021.

[11] . Sean Miller ., Curtis Busby-Earle ., " The role of machine learning in botnet detection " ., 2017.

[12] Rafael L. Gomes ., Antonia Raiane S. Araujo Cruz ., Marcial P. Fernandez ., " An Intelligent Mechanism to Detect Cyberattack of Mirai Botnet in IoT Networks " ., 2021.

[13] Stefano Secci ., Mathieu Bouet ., Agathe Blaise ., Vania Conan ., Stefano Secci ., " Botnet Fingerprinting: A Frequency Distributions Scheme for Lightweight Bot Detection " ., 2020.

[14] Raouf Boutaba ., Mohammad A . Salahuddin ., Abbas Abou Daya ., Noura Limam ., " A Graph-Based Machine Learning Approach for Bot- Detection " ., 2019.

[15] Madhuri Gurunathrao Desai ., Kun Suo ., Yong Shi ., " IoT Bonet and Network Intrusion Detection using Dimensionality Reduction and Supervised Machine Learning " ., 2020.

SCREENSHOTS:

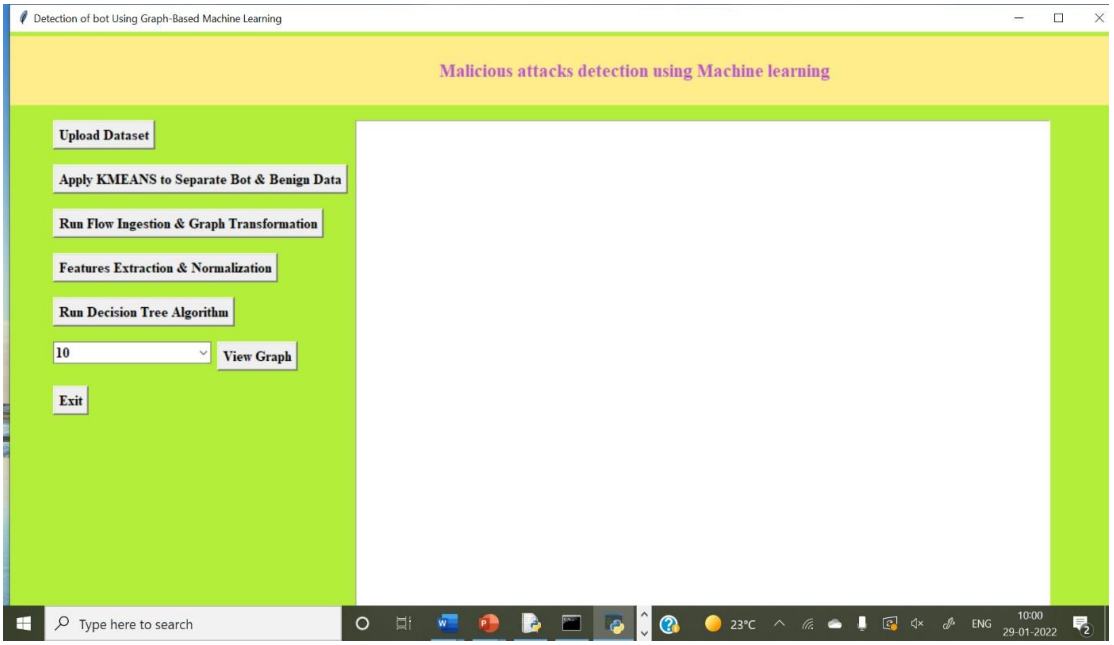


Fig 2: GUI screen

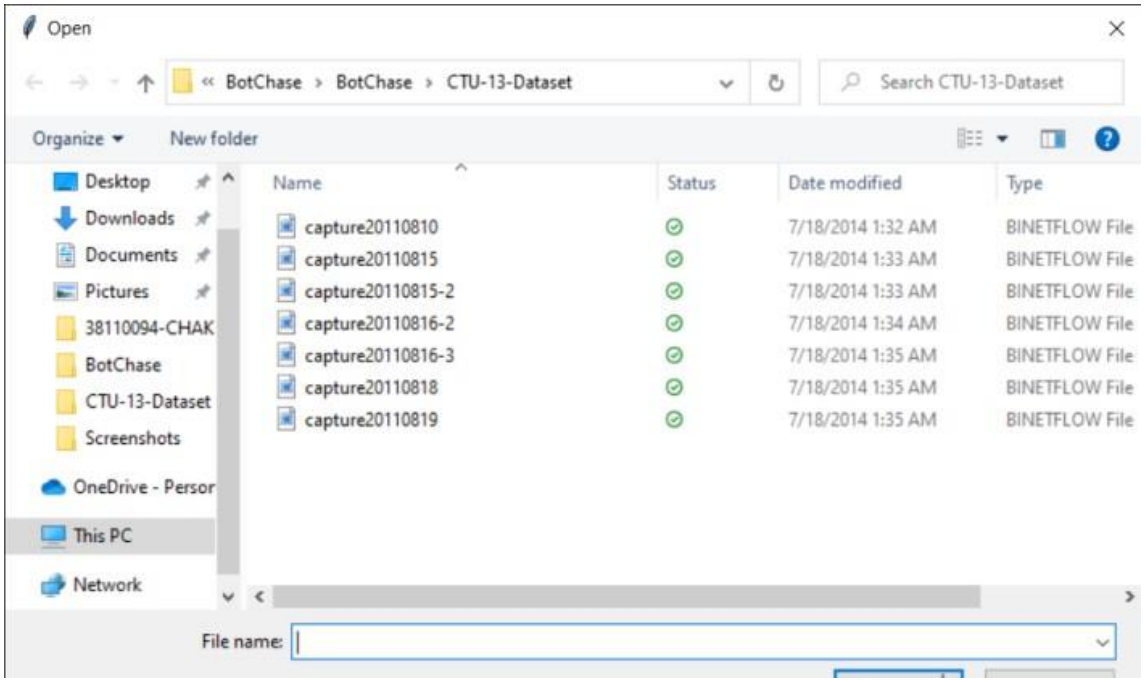


Fig 3:CTU-13 dataset

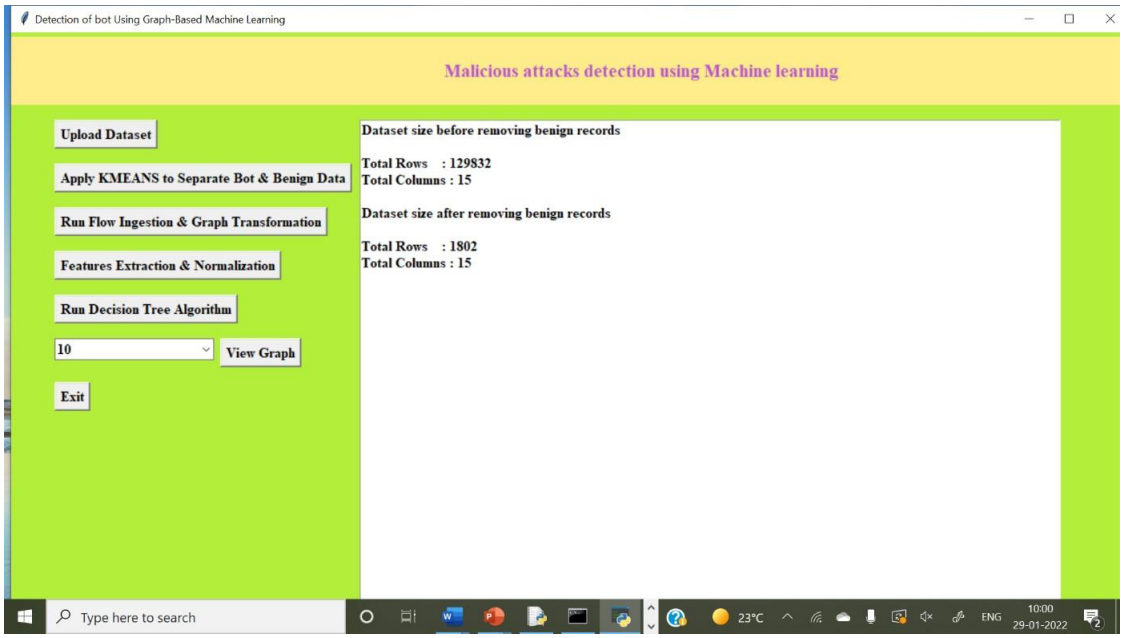


Fig 4: K-means

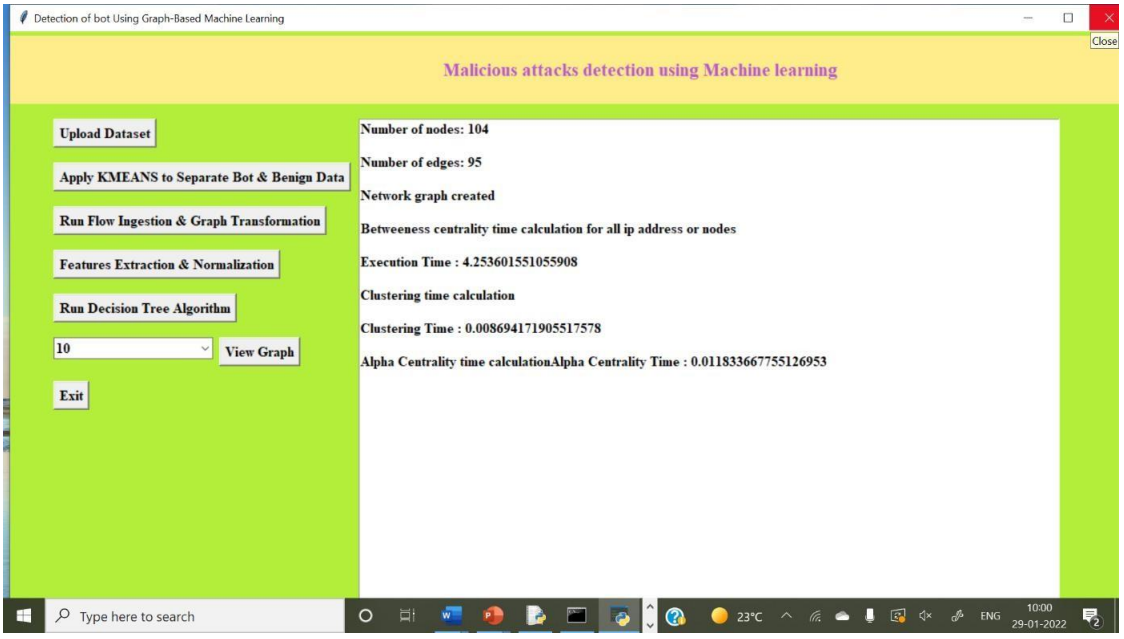


Fig 5: RunFlow Integration and graph transformation

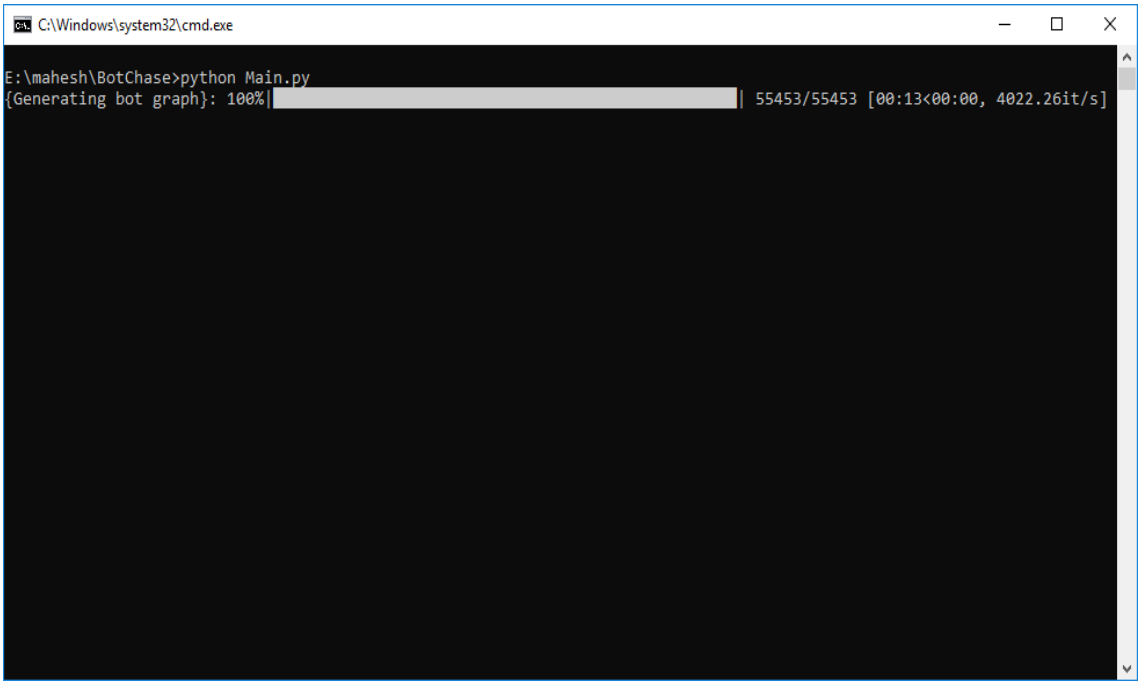


Fig 6: CMD graph build

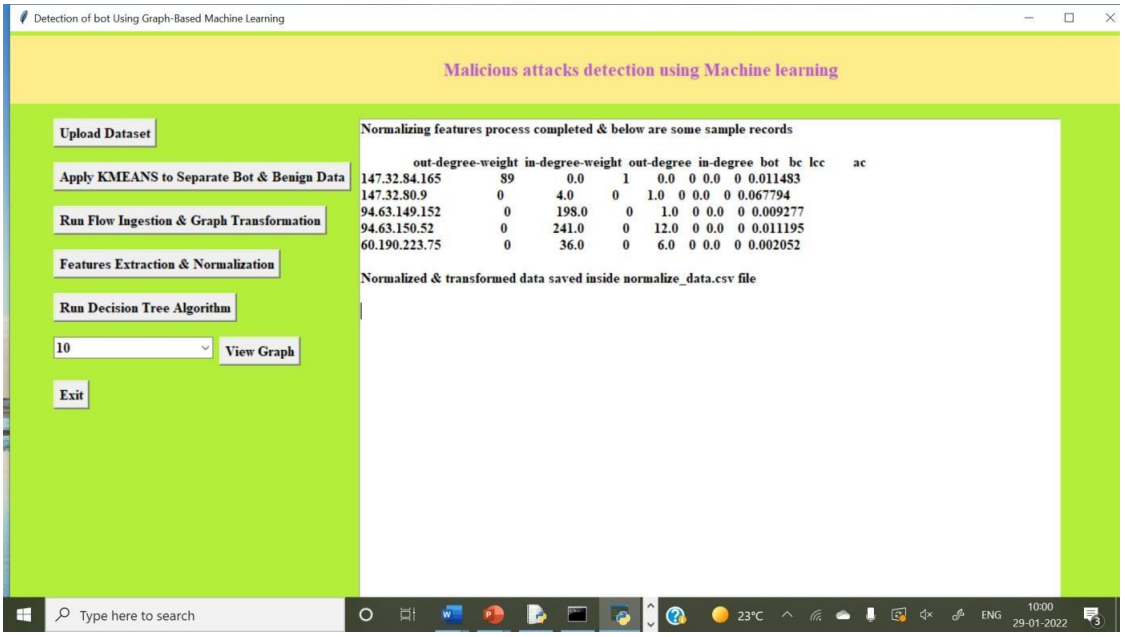


Fig 7: Features Extraction and normalization

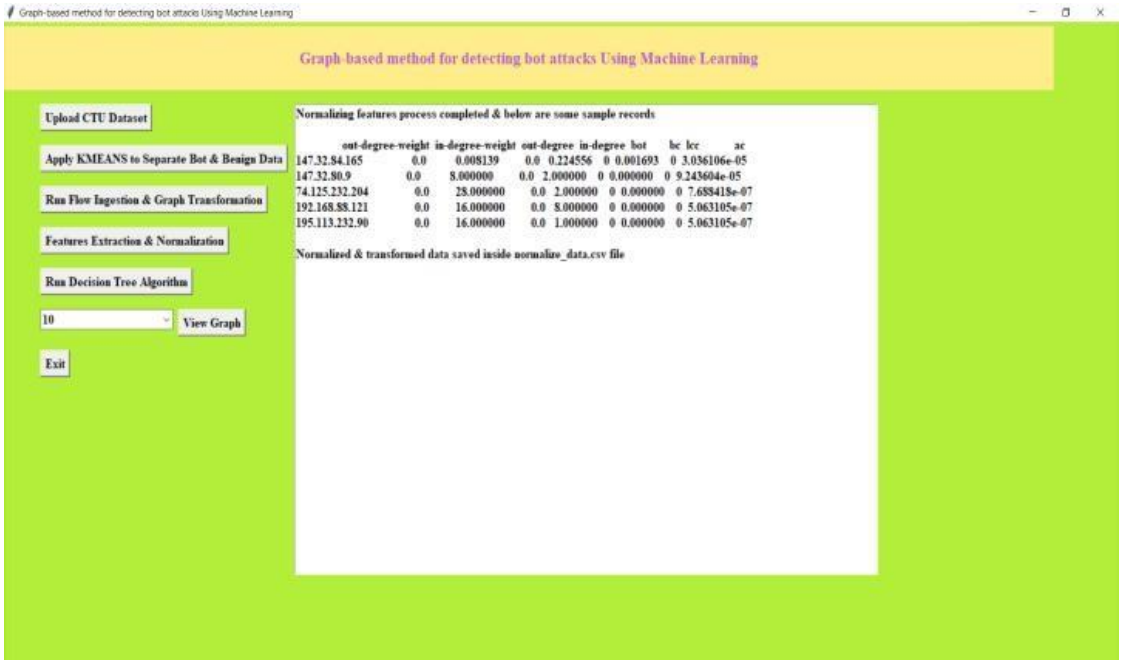


Fig 8: Run Decision Tree

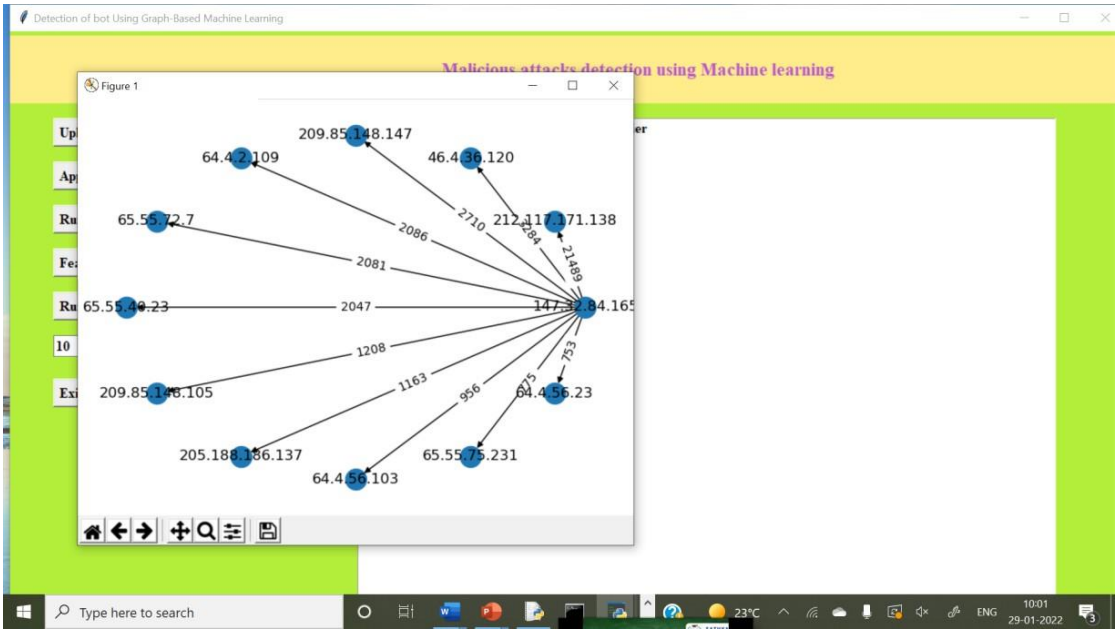


Fig 9: Graph

Source code:

```
font = ('times', 16, 'bold')
title = Label(main, text='Detection of bot Using Graph-Based Machine
Learning')
title.config(bg='LightGoldenrod1', fg='medium orchid')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=100)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=400,y=100)
text.config(font=font1)
font1 = ('times', 12, 'bold')
uploadButton = Button(main, text="Upload Dataset", command=upload)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

kmeansButton = Button(main, text="Apply KMEANS to Separate Bot & Benign
Data", command=kmeans)
kmeansButton.place(x=50,y=150)
kmeansButton.config(font=font1)

transformButton = Button(main, text="Run Flow Ingestion & Graph
Transformation", command=graphTransform)
transformButton.place(x=50,y=200)
transformButton.config(font=font1)

normalizationButton = Button(main, text="Features Extraction &
Normalization", command=featuresNormalization)
normalizationButton.place(x=50,y=250)
```

```
normalizationButton.config(font=font1)
dtButton = Button(main, text="Run Decision Tree Algorithm",
command=decisionTree)
dtButton.place(x=50,y=300)
dtButton.config(font=font1)

graphselection_list = []
graphselection_list.append(10)
graphselection_list.append(20)
graphselection_list.append(30)
graphselection_list.append(40)
graphselection_list.append(50)
graphselection_list.append(60)
graphselection_list.append(70)
graphselection_list.append(80)
graphselection_list.append(90)
graphselection_list.append(100)
graphlist =
ttk.Combobox(main,values=graphselection_list,postcommand=lambda:
graphlist.configure(values=graphselection_list))
graphlist.place(x=50,y=350)
graphlist.current(0)
graphlist.config(font=font1)
graphButton = Button(main, text="View Graph", command=viewGraph)
graphButton.place(x=240,y=350)
graphButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)
exitButton.place(x=50,y=400)
exitButton.config(font=font1)


main.config(bg='OliveDrab2')
main.mainloop()
```

C. PLAGARISM REPORT

Malicious Attacks Detection Using Machine Learning

Chakravarma Sai Tejaswi[1], Yemireddy Chaitanya[2], Jesudoss A.(*)[3], Prayla Shyry[4]
U.G Students[1,2], Associate Professor[3], Professor[4]
Department of CSE, Sathyabama Institute of Science and Technology,
Chennai, India
saitejaswi.chakravaram05@gmail.com, yemireddychaitanya569@gmail.com, jesudossas@gmail.com,
praylashyry.cse@sathyabama.ac.in

Abstract: **— Botnet, which are used for cybercrime, have recently become a powerful threat on the Internet. Using machine learning techniques, we have examined ways to detect botnet. There are various types of botnet attack, such as DDOS, spamming, fraud, etc., that can be used by malicious users to attack systems. In order to detect such attacks, packet analysis signatures are used and marked as normal or as human attacks. In order to identify if a new request packet is an attack or not, signatures will be applied, and this method requires manual effort and is updated every time a new attack occurs. The author will utilise machine learning algorithms in order to overcome the above problem. Machine learning algorithms will be used to train and create a model, which will then be applied to new request data to detect normal and abnormal actions. Using the KMEANS algorithm, we will separate the dataset into BOT and BENIGN records. In this approach, we will use graph-based features to extract features from the dataset. Data will be sent to a graph, where each address will be represented as a vertex, and edge connections will be made between the source and destination. Edge weights will be calculated based on incoming and outgoing link connections. To determine edge weights, we will calculate between_ness centrality, incoming edge weight, outgoing edge weight, and alpha_centrality weight. We combine the results from all these calculations into in-deg, out-deg, in-deg_wt, out_deg_wt, clustering, and alpha_centrality as features. If there are a high number of connections, then the label will read "1" (BOT); if not, then "0"(normal).**

Keywords :
Machine Learning (ML) ,GUI - application ., decision tree ., K- mean clustering ., database ., Graph ., botnet ., detection ., ctu-13 .

I. INTRODUCTION

Now a days everyone is storing their information in their systems. Here comes a problem in providing security to their systems. On other hand cyber-attacks are also increasing randomly which can hack your personal data like photos, social media and chats. Bot attacks increased worldwide. There are also some servers getting hacked which contains data of some lakhs people, where hacking a server is equal to hacking some lakhs people data. Botnet is also a type of cyber-attack which is a collection of internet-connected devices, where these devices are called as bot. By using this bots the attacker can also hack a big servers. These bots all together called as bot army. Botnet can make time-consuming tasks easier because of its army. Botnet also perform helpful tasks like keeping track of points during online game or managing chatrooms. But most people are using it for malicious works. It is also a source of many malicious activities. The different models of botnet are Client/Server model, Star Network Topology, Multi-server Network Topology, Hierarchical Network Topology and Peer-tp-Peer. There are many types in botnet like centralized, client-server, decentralized and peer-to-peer models and attacks like distributed denial-of service (DDoS), phishing, cryptojacking, snooping, bricking, Brute force and spambots. Common Botnet actions are Email spam, Financial breach, Targeted intrusions. A bot herder can do a collective of hijacked devices by using remote commands. Once your machine is infected, it becomes a bot, you may not even know. Botnet leads to Financial theft, Informational theft,

Sabotage of services, Selling access to other criminals. The 3 main components of botnet are the bots, the command and control servers(C&C) and the botnet operator. Botnet attacks has been increased in the recent years at the same time different types of Botnet detection frameworks are also increased.

The hacker can access the device only when his application was in the device. Once his application started running in the device then he can steal, change or destroy information. The hacker can also steal money, username and passwords. The hacker can also change your confidential data. Also install and run any application in your system he want. All the devices which are connected to the internet can be hacked by the hacker. The more targeted devices like desktop and laptops which runs on Windows OS or macOS. Mobiles are next target devices as more people are using by connecting them to the internet. Recent years connecting devices to the internet has increased rapidly botnets also create from connected devices has become more noted.

First the hacker will start by injecting the malware infection to your device. Then use techniques like web downloads, popup ads, email attachments, exploit kits a. Send some download links to the target device to hack the device. For example Trojan Horse (Happy New Year! Click here to see magic). If the owner of the device does not know about whether the download link is an attacker link and if he click on the link then the hacker application will get download in the device and sit around wait for command from the main system (hacker system). Now the hacker can access everything from his device. In order not to get attacked by hackers he should know all the malware links, so he can save his device from hacker. To stay away from malware links his device should able to find the malware links or prevent the initial infection or identify an existing infection. Botnet attacks are hard to detect. Preventing botnet attacks is more difficult. Yet we can still take certain measures to prevent botnet attacks.

The aim of our research is to explore new graph-based features. A unsupervised algorithm k means and a supervised algorithm Decision Tree were used as classifiers.

II. RELATED WORk

Misha Mehra et al [1] has proposed a system to detect Botnet in Linux systems because embedded Linux is popular in recent IoT devices and also systems, they extracted 3 features for model In this we understand how to extract features.

Abdallah Moubayed et al [2] has proposed a forest model to detect Botnet, they designed a framework based on ML using DNS, So we learnt how to design framework from this project.

Mrutyunjaya Panda et al [3] has proposed efficient feature extraction and ml model to detect Botnet, here we learnt about the efficient feature extraction for our project and also to select good ml model.

Afnan Alharbi et al [8] has proposed graph-based Botnet detection, we learnt about graph from here and also they used five filters for extracting features we also learnt about that filter process here.

MohammadNoor Injadat et al[10] proposed an optimized approach to detect Botnet, we learnt about optimization and also optimized decision tree model here.

Antonia Raiane et al [12] intelligent mechanism to detect Botnet, attacks related to scam, UDP learnt about network attacks from here.

Vasileios et al [4] has proposed efficient detection of Botnet and ML model training, we learnt to structure data in a good manner for good model training .

Sina Hojjatinia et al [5] has proposed to detect enhanced Botnet using ML model hybrid Artificial Intelligence we learnt ML models here.

Duc C. Le et al [7] proposed a dependable botnet detection from evolving network using genetic programming, learnt about network in this project.

Sriram et al [9] proposed to detect botnet using deep learning and network based flows and also released benchmark dataset also used machine learning models, designed a framework.

Agathe Blaise et al [13] proposed a technique to detect Botnet the name is BotFP signature plays a major role in this project and used supervised algorithm.

Madhuri Gurunathra et al [15] proposed a Botnet detection using supervised machine learning algorithms mainly focused on intrusions and other type of networks using dimensionally reduction technique.

Abbas Abou Daya et al [14] has proposed a graph based Botnet detection also used both supervised and unsupervised algorithms from ML, we learnt unsupervised algorithm from this project.

Sean Miller et al [11] defines the brief overview of all ML algorithms which plays crucial role in detecting Botnet and clearly understand about all the ML algorithms in Botnet detection.

Omar Y. Al-Jarrah et al [6] proposed a Botnet intrusion detection using RDPLM, learnt feature

selection technique from this project.

III. MOTIVATION

Cyberattacks are on the rise these days. Many systems are getting infected by attacks to overcome these attacks, In the past, we used signature-based research. However, as technology developed, attacks became more sophisticated and we used k-means and decision trees to see how many bots were targeted and how many were not. If there is an attack, we will find how many bots were attacked or detected and we will give the number.

IV. PROPOSED SYSTEM

The proposed work is to detect bot based on graph. Previously we have bot detection method but based on signature, which can not detect the new attacks. But our project can detect the newly generated attacks as we are using k means unsupervised algorithm.

Here we used two machine learning algorithms supervised and unsupervised. K means from unsupervised and decision tree from supervised. Where k means can separate bot and benign. Decision tree can give the accurate decision. We also found the accuracy of the Decision tree algorithm at last.

Supervised: A method of teaching machine learning labeled data by hand is called supervised learning. its already know output of the algorithm before it start working on it, example classifying a dataset in CTU-13, here it matches the input to output, here we will train the data and and tested the data , once algorithm is well trained, it is tested using the new data when it comes to unsupervised learning the training phase is big because the machine is only given the input,it has to figure out the output on its own, so there is no supervisor here or there's is no mentor over here.

Cyberattacks are on the rise these days. Many systems are getting infected by attacks to overcome these attacks, In the past, we used signature-based research. However, as technology developed, attacks became more sophisticated and we used k-means and decision trees to see how many bots were targeted and how many were not.

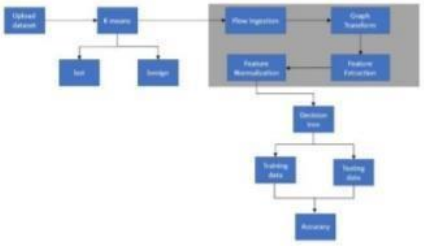
If there is an attack, we will find how many bots were attacked or detected and we will give the number.In fig[1] it consist of different Components like data bootstrap, model training.

Data-Bootstrap:

It consist of 4 internal components flow integration, graph transform, feature extraction, Feature Normalization .

Model Training:

Here we under go two phases , phase 1 is unsupervised , using k-means algorithm we separate bots and benign, in phase 2 it under goes supervised , using Decision tree algorithm to get the accurate values. using this we train, 80% , trained 20% data is tested.



SYSTEM ARCHITECTURE

Supervised learning:

All machine learning systems aim to predict an outcome. However, the whole process of supervised learning is designed so that it can directly predict the outcome, because it has well defined training phase, on other hand. Since it has already been trained, there is a direct feedback mechanism in supervised learning.

Types of problems: regression , classification

Unsupervised

Unsupervised learning involves the machine learning without any guidance in the form of unlabeled data.Here it forms as groups for example in this project like attack and non-attack, the only difference is it Cant add the labels, it understands how the cluster groups separate .Types of problems: Association ,clustering: separating on based Anomaly The detection of unusual activities can be used for detecting suspicious activity and the reinforcement of these activities is what we call reinforcement learning now.In unsupervised learning we must find patterns in data and keep exploring the data until it reaches the output. Observing patterns and extracting insights in unsupervised approaches is all about figuring out how to get the output, since the algorithm is only given input, it must find ways to gain insights from data by finding trends and associations, mapping the known input to known outputs.

K-means

It's a technique most of us do in our daily life, for example like group of people sharing tableClustering is the process of dispersing datasets into groups consisting of similar data points. For

example: k-means clustering. Exclusive clustering is hard clustering, where points/items belong only to one cluster.

Descion tree: (supervised)
Descion tree it can be used as both supervised and unsupervised, but in this project we are using decision tree as supervised algorithm. It has a root that grows as a number of different options is increased, similar to decision trees. Based on many different variables, a decision tree displays all possible options. and the condition now, here we will split the dataset into different subsets will become the input to child, the goal is produce the purest possible distribution of the labels at each nodes.

In this project we are using k-means and desicion tree algorithms for building this projecte.
To execute the project we have to click on run , then the CMD opens which shows the path of project where it located, after that the user interface opens, splits of 2 screens one screen contains buttons Other side it shows the executed functions output.

CHAPTER

A. Upload CTU Dataset
B. Apply KMEANS to separate Bot & Benign Data
C. Run Flow Ingestion & Graph Transformation
D. Features Extraction & Normalization
E. Run Decision Tree Algorithm
F. View Graph
G. Exit

A .Uploading ctu dataset

Click on "Upload ctu dataset" button ,it displays a set of datasets.Then choose a dataset from the given sets and click on open button.After uploading the dataset on the screen, it display the path from where we are taking dataset and dataset size.It displays total rows, total columns, Start Time, Duration, Protoc, Srcorce-Address, Sport, Dire, Dst Address, Dport, State, sTos, dTos, Total Packets, Total Bytes, SrurceBytes, Label and also the rows and columns in side square braces.

B. Apply k-means to separate bot and benign data:

Apply k-means to separate bot and benign data from the data set , it gives us the dataset size before removing benign records total rows and columns, and also it gives the dataset size after removing the benign records total rows and columns by using k-means we separate there data.

C. RunFlow Integration and graph transformation

After clicking on run flow integration it shoes two screens extract which we need to close ,when we have a look at the CMD there it show as generated bot graph points , on ui it shoes the number nodes , number of edges, number of graph created , between-Ness centrality for all IP address or node , here ip address nothing but nodes, Execution time, clustering time calculation, alpha centrality time calculation Alpha Centrality time.

D. Features Extraction and normalization:

After clicking on it, Normalizing features process completed & below are some sample records out out- degree-weight in-degree-wt outdegree ,indegree bot bc lcc ac, all the values of it which are normalized, Normalized & transformed data saved inside normalize_data.csv file, as well as we can have a look at the CMD there it show as features normalization module 100 percent done and shoes the record in it.

E. Run Decision Tree Algorithm

It shows Normalized data loading to decision tree classifier, Total dataset size to build model, Model training records size, Model testing records size, Decision Tree Accuracy , Decision Tree Precision , Decision Tree Recall ,True -Post , False- Post ,True-Nega, False-Nega . we have test 20 % of data, and training 80% of data .The Accuracy of this model is 99%.

F. View Graph

In the final module there will be input textbox where we can enter some number into it , so that it generate the graph after clicking on the view graph.it pop up another screen shoes all the ip address and its connections. After completing the whole project clicking on exit we exit from the GUI interface.

G.EXIT

Clicking on exit button we will exit from GUI interface. come out of project.

V. RESULTS AND DISCUSSION

In Fig[2] click on 'Upload CTU Dataset' button and upload dataset, In Fig[3] uploading first capture file and now click on 'Open' button to upload .In Fig[4] dataset contains total 2824636 records and each record contains 15 columns and below it I am displaying some dataset records. Now click on 'Apply KMEANS to separate Bot & Benign Data' button to remove benign records
In Fig[5] we can see dataset size before removing benign records and after removing benign records. By removing some benign records we can reduce dataset size. Now click on 'Run Flow Ingestion &

Graph Transformation' button to generate grap .In Fig[6] we can see progress bar which indicates graph based features extraction and while applying this technique it will open 2 empty windows and you just close those 2 empty windows to get window. In Fig[7] we can see total nodes and edges generated and time taken to calculate between_ness, alpha centrality and clustering. Now we generate graphs and now click on 'Features Extraction & Normalization' button to extract features and to perform normalization on extracted features.In Fig[8] after normalization I am displaying few records with out_degree, in degree and weight details. In Fig[9]'bc' refers to between_ness and 'lcc' refers to clustering and 'ac' refers to alpha centrality. All normalized records are saved inside 'normalize_data.csv' file and you can open and see that file from code folder. Now click on 'Run Decision Tree Algorithm' button to generate training model with decision tree classifier and to calculate metrics such as accuracy, precision etc. In Fig[10] after normalization we got total records as 3174 with 7 columns (in_degree, out_degree, weight etc.) and application split total records into train size as 2539 and test size as 635. After building train model we apply test records and got accuracy as 99%. Below screen showing normalization progressing step.

Fig 4: After uploading CTU dataset

Fig 5: K-means

Fig 2: GUI screen

Fig 6: RunFlow Integration and graph transformation

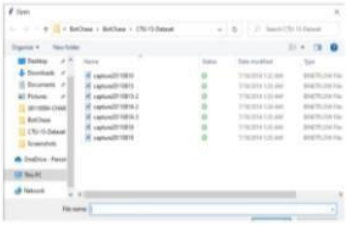


Fig 3: CTU dataset

Fig 7: Features Extraction and normalization

Fig 8: Run Decision Tree

Fig 9: Graph

Fig 10: CMD

VI CONCLUSION

Botnet allows you to combination community flows into graphical version based on network flow facts. In the primary phase, SOM is used to make sure an Maximizing the benign clusters but maintaining an acceptable compromise while alienating the malicious bots. Additionally, the consequences show high TPs ,coffee FPs for DT. Without the F Norm, the effects of the SOM have been made worse, i.E., fewer bots within the normal (bengin) cluster, and the size of the benign cluster reduced. In addition to detecting bots that use one of a kind protocols, BotChase is also capable to educate and infer ML fashions for pass-network ML education is attacked by go-community. Graph-based totally capabilities outperform go with the flow-based features in BotChase. Further, BotChase outperforms an quit-to-cease device that is predicated on float-based capabilities and compares favorably with the graph-based Bot detection. BotChase, in web-primarily based surroundings, applies incremental learning using HAT. FNorm requires longer to converge, however the model performs extremely nicely in its very last country. Future research consciousness on tuning the classifiers, investigating superior ensemble gaining knowledge of and feature engineering strategies, and increasing FNorm to better degrees.

VII. References

[1] Misha Mehra, Jay N. Paranjape and Vinay Joseph Ribeiro, "Improving ML Detection of IoT Botnets using Comprehensive Data and Feature Sets", 2021.

[2] Abdallah Moubayed, MohammadNoor Injadat and Abdallah Shami, "Optimized Random Forest Model for Botnet Detection Based on DNS Queries", 2021.

[3] Mrutyunjaya Panda, Abd Allah A. Mousa and Aboul Ella Hassanien, "Developing an Efficient Feature Engineering and Machine Learning Model for Detecting IoT-Botnet Cyber Attacks", 2021.

[4] Vasileios A. Memos and Kostas E. Psannis, "AI-Powered Honeypots for Enhanced IoT Botnet Detection",2020.

[5] Sina Hojjatinia, Sajad Hamzenejadi and Hadis Mohseni , "Android Botnet Detection using Convolutional Neural Networks" 2020.

[6] Omar Y. Al-Jarrah, Omar Alhussein, Paul D. Yoo, Sami Muhaidat, Kamal Taha and Kwangjo Kim, "Data Randomization and Cluster-Based Partitioning for Botnet Intrusion Detection", 2015.

[7] Duc C. Le and Nur Zincir-Heywood, "Learning From Evolving Network Data for Dependable Botnet Detection", 2020.

[8] Afnan Alharbi and Khalid Alsubhi, "Botnet Detection Approach Using Graph-Based Machine Learning", 2021.

[9] S. Sriram, R. Vinayakumar, Mamoun Alazab and Soman KP "Network Flow based IoT Botnet Attack Detection using Deep Learning", 2020.

[10] MohammadNoor Injadat, Abdallah Moubayed and Abdallah Shami, "Detecting Botnet Attacks in IoT Environments: An Optimized Machine

Learning Approach", 2021.
[11] Sean Miller and Curtis Busby-Earle, "The role of machine learning in botnet detection", 2017.
[12] Antonia Raiane S. Araujo Cruz, Rafael L. Gomes and Marcial P. Fernandez, "An Intelligent Mechanism to Detect Cyberattacks of Mirai Botnet in IoT Networks", 2021.
[13] Agathe Blaise, Mathieu Bouet, Vania Conan and Stefano Secci, "Botnet Fingerprinting: A Frequency Distributions Scheme for Lightweight Bot Detection", 2020.
[14] Abbas Abou Daya, Mohammad A. Salahuddin, Noura Limam and Raouf Boutaba, "A Graph-Based Machine Learning Approach for Bot Detection", 2019.
[15] Madhuri Gurunathrao Desai and Yong Shi, Kun Suo, "IoT Botnet and Network Intrusion Detection using Dimensionality Reduction and Supervised Machine Learning", 2020.

Malicious Attacks Detection Using Machine Learning

ORIGINALITY REPORT

3%	2%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.crowdstrike.com Internet Source	1%

Malicious Attacks Detection Using Machine Learning

ORIGINALITY REPORT

3%	2%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.crowdstrike.com Internet Source	1%
2	Afnan Alharbi, Khalid Alsubhi. "Botnet Detection Approach Using Graph-Based Machine Learning", IEEE Access, 2021 Publication	1%
3	Submitted to New Jersey Institute of Technology Student Paper	<1%
4	Jun Zhao, Xudong Liu, Qiben Yan, Bo Li, Minglai Shao, Hao Peng. "Multi-attributed heterogeneous graph convolutional network for bot detection", Information Sciences, 2020 Publication	<1%
5	ijsrcseit.com Internet Source	<1%
6	www.coursehero.com Internet Source	<1%
7	Li Yang, Dimitrios Michael Manias, Abdallah Shami. "PWPAE: An Ensemble Framework for Concept Drift Adaptation in IoT Data Streams", 2021 IEEE Global Communications Conference (GLOBECOM), 2021 Publication	<1%
8	mdpi-res.com Internet Source	<1%