SmartSecure: Machine Learning Based Intrusion Prevention System

Submitted in partial fulfilment of the requirements of the degree of

BACHELOR OF COMPUTER ENGINEERING

by

Harmit Saini, 20102125

Siddharth Singh, 20102176

Vandan Savla, 20102137

Sarvesh Shirwalkar, 20102109

Guide

Dr. Sachin H. Malave

Department of Computer Engineering

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE
University of Mumbai

2023-2024

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

CERTIFICATE

This is to certify that the project entitled "**SmartSecure: Machine Learning Based Intrusion Prevention System**" is a bonafide work of **Harmit Saini (20102125), Siddharth Singh (20102176), Vandan Savla (20102137), Sarvesh Shirwalkar (20102109)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Engineering.**

Guide	Project Coordinator
Dr. Sachin H. Malave	Prof. Rushikesh R. Nikam
Head of Department	Principal
Dr. Sachin H. Malave	Dr. Uttam D. Kolekar

Date:

A. P. SHAH INSTITUTE OF TECHNOLOGY, THANE

Project Report Approval for B.E.

This project report entitled **"SmartSecure: Machine Learning Based Intrusion Prevention System"** by **Harmit Saini, Siddharth Singh, Vandan Savla, Sarvesh Shirwalkar** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering**, **2023-2024**.

Examiner Name	Signature
1. _____	_____
2. _____	_____

Date:

Place:

Declaration

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

Harmit Saini - 20102125

Siddharth Singh - 20102176

Vandan Savla - 20102137

Sarvesh Shirwalkar - 20102109

Date:

Abstract

The project "SmartSecure " addresses the increasingly critical need for robust security measures on on-premise Servers. This project aims to develop a comprehensive security framework specifically tailored to on-premise server environments, focusing on proactive threat mitigation and adaptive defense mechanisms. The proposed framework combines cutting-edge cybersecurity technologies, best practices, and automation to create a robust defense strategy that streamlines server's security. This project introduces an innovative approach to security through the design and development of an automated server security tool that leverages Machine Learning (Machine Learning) to enhance security and detect potential cyber threats. This is an approach to design an intelligent IPS (Intrusion Prevention System) with very little performance overhead. The project is designed to detect various types of DOS/DDOS attacks with Machine Learning Model running in background. The architecture of the tool is designed in such a way to minimize the latency to end-users.

Keywords: Machine Learning, DOS/DDOS (Distributed Denial of Service), Intrusion Prevention System.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION

ML	Machine Learning
DDOS	Distributed Denial of Service
UML	Unified Modeling Language
DFD	Data Flow Diagram
IPS	Intrusion Prevention System

CHAPTER 1

Introduction

In the realm of information technology, servers stand as the cornerstone of digital infrastructure. They are specialized computer systems or software designed to respond to requests, process data, and distribute services to client devices. In the context of modern data processing and communication, servers are the essential workhorses that enable the retrieval of web pages, the delivery of emails, the storage of critical business data, and the execution of computational tasks.

Security is paramount for individuals and businesses hosting their digital assets on servers. Server hosting, with its isolated nature, offers inherent security advantages. However, it's crucial to take additional measures to safeguard the server. These include regularly updating the operating system and software to patch vulnerabilities, implementing robust firewall rules, employing intrusion detection systems, and using strong, unique passwords. Proper access control by limiting user privileges, monitoring server logs for unusual activities, and encrypting sensitive data is essential. Regular backups and disaster recovery plans are also vital for mitigating risks. By following these security best practices, server users can significantly enhance the protection of their hosted content and maintain a secure online presence.

Out of many attacks possible major threat that a server encounters is that of DOS/DDOS. There are some tools that are available that either check logs to detect the threat and then ban the IP address

for example fail2ban. Our approach is slightly different instead of an attack happening for the first time we try to prevent it from entering the system. This approach is divided into three steps , Network traffic analysis, threat detection using a machine learning algorithm, and finally configuring the iptables to secure the system.

1. **Network Analysis:** We start by capturing network traffic data using CICFlowMeter, a popular tool to generate Bi-directional flows. CICFlowMeter is a network traffic flow generator and analyser. It can be used to generate bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence more than 80 statistical network traffic features such as Duration, Number of packets, Number of bytes, Length of packets, etc. can be calculated separately in the forward and backward directions. Additional functionalities include, selecting features from the list of existing features, adding new features, and controlling the duration of flow timeout. The output of the application is the CSV format file that has six columns labeled for each flow (FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol) with more than 80 network traffic analysis features. The features of the dataset used to train our Machine Learning Model is generated by using the GUI variant of this tool that is developed in Java. But for our application to work as command-line application, we have used its command-line variant that is written in Python.

   Every packet holds essential information about each network connection, including where it's coming from, where it's going, its size, and more. Majorly two components are there in packets that are packet headers and payload. Packet headers contain all the critical information that helps network equipment decide what to do with each packet. Most important are the source and destination addresses and count of IP packets. This data becomes the foundation for our analysis and will be fed to our detection model.

2. **Threat Detection**: For threat detection we have used Random Forest Algorithm that is a powerful tool in the realm of cybersecurity for detecting and mitigating cyber threats. This machine learning technique helps security professionals make informed decisions by constructing a tree-like model of possible attack scenarios. Random Forest analyze incoming data and make decisions based on a series of predefined rules or conditions. In the

context of cyber threat detection, Random Forest can assess network traffic, user behavior, or system anomalies to classify and identify potential threats. By learning from historical data, the algorithm can recognize patterns associated with known threats and take appropriate actions in real-time, such as blocking suspicious activities, alerting security personnel, or initiating defensive measures. The Random Forest Algorithm offers the advantage of both automation and adaptability, making it an invaluable asset in the continuous battle against evolving cyber threats.

3. **System Configuration:** Using the results from the algorithm , we can decide what measures to be taken. In our approach instead of blacklisting all the IP addresses we whitelist the genuine connections. Whitelisting the IP adressess makes the system delay free by not checking those IP addresses everytime it makes connection, only new IP addresess are checked. Logs are also maintained alongside for future reference and maintenance.

CHAPTER 2

Literature Survey

In the contemporary landscape of cybersecurity, the proliferation of sophisticated threats poses significant challenges to the integrity and security of computer networks. Intrusion Prevention Systems (IPS) have emerged as crucial components in safeguarding networks by actively detecting and thwarting malicious activities. This literature survey aims to provide a comprehensive overview of the state-of-the-art Machine Learning-based Intrusion Prevention Systems. It delves into various Machine Learning algorithms, architectures, and methodologies employed in developing Intrusion Prevention Systems, along with their strengths, limitations, and potential challenges.

The article presents a response to cybersecurity challenges within cloud virtualization technology, which exposes virtual machines (VMs) to risks on remote servers. It introduces the MR-TPM (Multiple Risks analysis-based VM Threat Prediction Model) to predict threats and bolster data security proactively. The model efficiently analyzes cybersecurity risks by learning from historical and live data samples, significantly reducing threats by up to 88.9% when deployed alongside existing VM allocation policies. [1]

The study addresses the necessity for tools like network mapping and vulnerability scanning due to diverse attacks targeting all accessible network ports. Machine Learning (ML) has

emerged as a valuable tool for enhancing Intrusion Detection Systems (IDS) to identify malicious network traffic. The study introduces a detection framework that utilizes an Machine Learning model to augment IDS in spotting anomalies in network traffic. [3]

This paper highlights the critical importance of the Domain Name System (DNS) in the Internet's functionality and security. It discusses how Distributed Denial of Service (DDoS) attacks have posed a significant threat to the stability of DNS, especially when dealing with a large number of registered domain names. The paper presents a model based on Random Forest for traffic classification with a high accuracy of 99.2% on the Spark platform. The classification model, built on Spark, achieves a high level of accuracy with 0.0% False Positive Rate (FPR) and 4.36% False Negative Rate (FNR), meeting practical demands. [6]

The paper presents an innovative automation system designed to validate configuration and security in managed cloud services. It stands as a pioneering solution in this domain, aiming to facilitate proactive risk management and issue resolution prior to service deployment. The system is developed using J2EE and utilizes a local database, offering RESTful APIs for integration. It employs standardized scripts for evidence collection and introduces a policy mechanism to address exceptions, thereby averting configuration and security conflicts during server provisioning. [7]

The paper outlines three key principles essential for robust peer-to-peer (P2P) system design to mitigate Distributed Denial of Service (DDoS) attacks. It introduces an active probing method for membership validation, which operates effectively without dependence on centralized authorities. This approach is noted for its minimal impact on system performance, offering promising prospects for bolstering the resilience of P2P systems against DDoS attacks. Future research directions involve exploring mechanisms to limit amplification during DDoS attacks on participating nodes and investigating self-tuning mechanisms for system parameters. [10]

	Paper Name	Strengths	Drawback
[1]	1. D. Saxena, I. Gupta, R. Gupta, A. K. Singh and X. Wen, "An AI-Driven VM Threat Prediction Model for Multi-Risks Analysis-Based Cloud Cyber-security," in IEEE Transactions on Systems, Man, and Cybernetics: Systems	The article addresses cybersecurity concerns in cloud virtualization technology, which exposes users' virtual machines (VMs) to potential threats due to inherent vulnerabilities on remote servers. The model introduces a novel VM threat prediction system called MR-TPM (Multiple Risks analysis-based VM Threat Prediction Model) that anticipates threats and enhances data security in a proactive manner.	Proposed model efficiently computes the cybersecurity risks and learns the VM threat patterns from historical and live data samples. The deployment of this model with existing VM allocation policies reduces cybersecurity threats up to 88.9%.
[2]	Wang, K., Stolfo, S.J. (2004). Anomalous Payload-Based Network Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds) Recent Advances in Intrusion Detection. RAID 2004. Lecture Notes in Computer Science, vol 3224.	The presented study introduces a novel intrusion detection method named PAYL, focused on payload-based anomaly detection. PAYL operates in an entirely automated, unsupervised, and highly efficient manner. During the training phase, the technique creates a profile by	This approach can be applied to any network system, service or port for that site to compute its own “site-specific” payload anomaly detector, rather than being dependent upon others deploying a specific signature for a newly detected worm or exploit that

	Springer	calculating the byte frequency distribution and standard deviation of the normal application payload that flows to a specific host and port.	has already damaged other sites.
[3]	Alshammari, A., Aldribi, A. Apply machine learning techniques to detect malicious network traffic in cloud computing. J Big Data 8, 90 (2021)	Diverse attacks target all accessible network ports, necessitating tools like network mapping and vulnerability scanning. Machine learning (ML) has gained prominence for enhancing Intrusion Detection Systems (IDS) to identify malicious network traffic. This study introduces a detection framework that employs an ML model to augment IDS in identifying anomalies in network traffic.	A reliable model running in Real-time to detect malicious data flow traffic depending on the Machine Learning supervised techniques based on the ISOT-CID dataset that contains network traffic data features. IDS security systems for computer networks must be very fast where it is deployed in real-time to extract the communication traffic characteristics and give its response in real-time.
[4]	Souri, A., Hosseini, R. A state-of-the-art survey of malware detection approaches using data mining techniques. Hum. Cent. Comput. Inf. Sci. 8, 3	The ongoing battle between security analysts and malware researchers continues to intensify due to the ever-evolving landscape of technology. Existing	The malware detection approaches were compared and analyzed according to various essential factors such as classification approaches, data analysis methods, the

	(2018)	methods are struggling to keep up with the rapid changes in the complex and evolutionary nature of malware, making detection more challenging. The paper categorises these methods into two main groups: signature-based approaches and behaviour-based detection.	number of the used dataset, accuracy factor and case study analysis. The SVM method has most percentage for malware detection approach with 29%, j48 has 17%, Decision tree has 14%, NB has 10%, BF has 5% and the other methods have less than 3% usage in data mining results..
[5]	SYN flooding attack detection by TCP handshake anomalies Martine Bellaïche, Jean-Charles Grégoire 16 August 2011	The article introduces a novel method for detecting SYN flooding attacks from the victim's perspective, focusing on classifying various forms of TCP handshakes that occur during connection establishment between a client and a server (e.g., for Web traffic). The approach involves a real-time data structure to monitor TCP handshake states, with insights into its performance, initialization, and flow management.	The observation of the hand-shake sequence with the CUSUM algorithm is an appropri- ate choice for the detection of SYN flooding attacks. The detection system can be evaluated according to detection time, detection rate and false positive rate.

[6]	Detection of DNS DDoS Attacks with Random sForest Algorithm on Spark Liguo Chena, Yuedong Zhang, Qi Zhao, Guanggang Geng , ZhiWei Yan	This paper highlights the critical importance of the Domain Name System (DNS) in the Internet's functionality and security. It discusses how Distributed Denial of Service (DDoS) attacks have posed a significant threat to the stability of DNS, especially when dealing with a large number of registered domain names. The paper presents a model based on Random Forest for traffic classification with a high accuracy of 99.2% on the Spark platform. This model is shown to effectively handle large-scale DNS query flows, making it practical for real-world use.	This paper presents a novel method to reduce DDoS traffic on Top-Level Domain (TLD) servers by applying machine learning-based traffic filters to major recursive DNS servers on the Internet. The classification model, built on Spark, achieves a high level of accuracy with 0.0% False Positive Rate (FPR) and 4.36% False Negative Rate (FNR), meeting practical demands. Future work will focus on feature extraction and real-time streaming application for firewall rule generation. The paper also hints at the study of a real-time detection and prevention system using this traffic filtering model in the near future.
[7]	Automation System for Validation of Configuration and Security Compliance in Managed	This paper emphasizes the importance of validating configuration and security compliance during the	The paper introduces an automation system for validating configuration and security in managed cloud

	Cloud Services Trieu C. Chieu Manas Singh Chunqiang Tang, Mahesh Viswanathan, Ashu Gupta	creation of new IT services. It's crucial for managing security risks, governance controls, and vulnerabilities throughout the service lifecycle. However, the traditional manual validation process is time-consuming and prone to errors, which becomes a challenge when delivering managed cloud services with strict SLA fulfillment times.	services. It's the first of its kind, ensuring proactive risk management and issue remediation before service delivery. The system is built on J2EE with a local database and offers restful APIs. It uses standardized scripts for evidence collection and introduces a policy mechanism to handle exceptions, preventing configuration and security conflicts during server provisioning.
[8]	Overview of use of decision tree algorithms in machine learning Arundhati Navada; Aamir Nizam Ansari; Siddharth Patil; Balwant A. Sonkamble	Decision trees are a type of tree structure where internal nodes represent tests on input data, and leaf nodes represent categories for those patterns. Decision tree algorithms have versatile applications, including data analysis, text extraction, missing data prediction, search engine enhancement, and medical fields. Various algorithms exist, each with different accuracy and cost-	ID3 has limitations with discrete data and noise. IDA improves but still has constraints. C4.5 excels, handling continuous data and more.

		effectiveness. The ID3 algorithm, while useful for simple decision trees, loses accuracy as complexity increases. Therefore, more advanced algorithms like IDA (Intelligent Decision Tree Algorithm) and C4.5 have been developed to handle more complex decision-making tasks.	
[9]	A Security Aspects in Cloud Computing Gurudatt Kulkarni & Jayant Gambhir, Tejswini Patil, Amruta Dongare	Cloud computing, while efficient, faces challenges in data privacy and security. This paper discusses these issues, highlights the reluctance to adopt cloud services due to security concerns, and proposes a secure cloud computing approach using separate encryption and decryption services to address these challenges.	This paper delves into security issues across various levels of cloud computing services: SaaS, PaaS, and IaaS. It emphasizes the importance of safeguarding customer information. Risk assessment and management are key challenges, requiring a balance between identified risks and privacy controls. The paper highlights security considerations and challenges in cloud computing, emphasizing its potential to offer secure and cost-effective IT solutions.

[10]	Preventing DDoS attacks on internet servers exploiting P2P systems Author links open overlay panel Xin Sun, Ruben Torres, Sanjay Rao	This paper addresses the exploitation of P2P systems for launching DDoS attacks on external servers. It categorizes these attacks based on the underlying amplification cause and highlights three key principles for attack prevention. The paper proposes an active probing approach to validate membership information, applicable to both structured and unstructured P2P systems. This approach shows promise in mitigating DDoS attacks without compromising application performance, as demonstrated in evaluations on file-sharing and video broadcasting systems.	This paper identifies three crucial principles for robust P2P design to prevent DDoS attacks and proposes an active probing approach for membership validation, which is effective without relying on centralized authorities. The approach minimally impacts performance, making it promising for enhancing P2P system resilience. Future work includes investigating mechanisms to bound amplification during DDoS attacks on participating nodes and exploring self-tuning mechanisms for parameter optimization.

Table 2.1: Literature Survey Table

CHAPTER 3

Limitation of Existing System

- **Risk of oversights in system configuration:** Misconfigured firewalls, improper access control, and neglecting timely security patches are common vulnerabilities that can compromise the security of systems and networks. These oversights can create openings for unauthorized access, data breaches, and other malicious activities. It is essential for organizations to prioritize regular security audits and updates to ensure that firewalls are properly configured, access controls are appropriately enforced, and security patches are promptly applied to mitigate potential risks and safeguard against potential cyber threats.

- **Shared server environments pose vulnerability risks if accessed by attackers:** Shared server environments present inherent vulnerability risks if accessed by attackers. In such environments, the compromise of one user's account or system can potentially lead to unauthorized access to sensitive data or resources belonging to other users.

- **Conventional security systems lack a machine learning-based approach:** While effective against known threats, they struggle to adapt to evolving cyber threats. Constantly shifting tactics of malicious actors highlight the need for dynamic defense mechanisms.

- **Machine learning offers transformative potential in cybersecurity:** Machine learning algorithms operate autonomously, leveraging data patterns and anomalies to continuously

learn and evolve their threat detection capabilities. Their ability to analyze vast datasets in real-time facilitates the identification of subtle and emerging attack vectors, ensuring proactive defense against evolving cyber threats. Furthermore, these models continuously refine their understanding of threats, enhancing their adaptability to new attack methods and bolstering overall cybersecurity resilience.

- **Resource Intensiveness:** Depending on the implementation, Intrusion Prevention System solutions can be resource-intensive, requiring significant processing power and memory. This can impact the performance of the network or system on which they are deployed.

- **Compatibility Issues:** Integrating Intrusion Prevention System solutions with existing network infrastructure and security tools can sometimes lead to compatibility issues, requiring additional effort to ensure seamless operation and interoperability.

CHAPTER 4
Problem Statement, Objectives, and Scope

4.1 Problem Statement

In today's digital era, the proliferation of cyber threats, notably Distributed Denial of Service (DDoS) attacks, poses a significant challenge to network security. These attacks disrupt online services by overwhelming servers with malicious traffic, leading to downtime and potential data breaches. Traditional Intrusion Prevention Systems (IPS) struggle to mitigate DDoS attacks effectively, necessitating a more advanced solution.

Our project aims to address this gap by developing a Machine Learning (ML) based IPS specifically tailored to detect and mitigate DDoS attacks. By analyzing network traffic patterns, the system can identify and block malicious connections associated with DDoS attacks while allowing legitimate traffic to access servers unimpeded. This innovative approach promises to enhance network resilience and ensure uninterrupted service availability in the face of evolving cyber threats.

4.2 Objectives

- Develop a robust Machine Learning (ML) algorithm capable of accurately identifying Distributed Denial of Service (DDoS) attacks within network traffic patterns.

- Design and implement a threat detection and prevention tool that integrates seamlessly into existing on-premise server infrastructures, enabling real-time monitoring and response to potential DDoS threats.

- To Implement multi-threading and multiprocessing to reduce the delay and increase efficiency of the machine learning model.

- To make use of the AI paradigm for implementing detection of possible attacks/threats via malicious users instead of signature based detection.

4.3 Scope

- **Detection Focus**: The project primarily focuses on developing an advanced Machine Learning (ML) based threat detection tool specifically tailored for detecting and mitigating Distributed Denial of Service (DDoS) attacks within network traffic. The scope encompasses the design, implementation, and evaluation of algorithms aimed at accurately identifying DDoS attack patterns while minimizing false positives.

- **Real-time Response**: The scope includes the implementation of a real-time monitoring and response mechanism within the system's framework to enable swift action upon detection of potential DDoS threats. Multithreading will also be employed to enhance processing speed and ensure uninterrupted monitoring of network traffic.

- **Integration Compatibility**: The scope extends to ensuring seamless integration of the Machine Learning-based solution into existing network infrastructures, thereby minimizing deployment complexities and operational overhead.

CHAPTER 5
Proposed System

5.1 Proposed System:

The proposed system of our project includes a series of modules in which the project work is breakdown and implementation of each module is done by using various software engineering skills required during the process.

5.1.1 Modules:

a. Network Analysis: This is the very first module of our project. It is exposed to internet for analysis of incoming connections. Feature extraction happens in this stage only.

b. Machine Learning Model: At the heart of the project lies this module, which processes features extracted from previous stages to pinpoint malicious connections**.** The features are fed to the trained Machine Learning model for threat detection. It detects which connection is malicious and further modules are activated. Random Forest algorithm is used for classification purposes

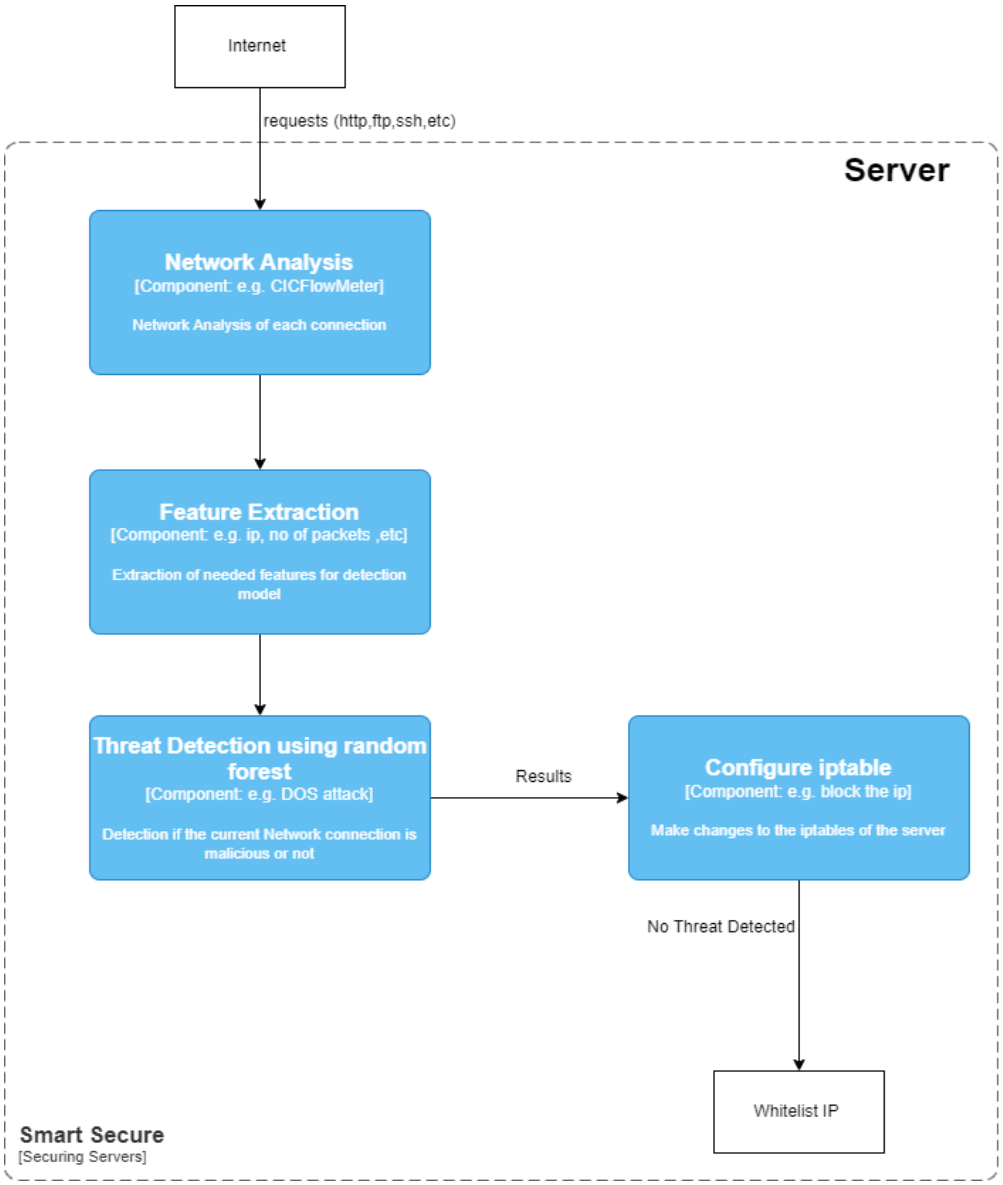
c. **System Configuration**: Following detection, the system modifies iptables as a preventive measure to limit malicious traffic. To minimize latency, rather than blacklisting each malicious IP address, genuine IP addresses are whitelisted.

Furthermore, the project operates on a multithreading system to optimize server uptime, ensuring continuous functionality.

5.2 Architecture Diagram:

The architecture diagram depicts the system's structure and components, illustrating how different parts interact. It provides a visual overview of the system's design, helping to understand its functionality and relationships between elements. There are major three components in our system the network analysis component, machine learning component and system configuration component. The traffic from the internet is analysed and saved in a file. The Machine Learning component then read the contents and gives prediction. Finally, system configuration components runs using the results from Machine Learning model.

Figure 5.2 – Architecture Diagram



5.3 UML Diagrams:

A UML diagram is a diagram based on the UML (Unified Modelling Language) to visually represent a system along with its main actors, roles, actions, artifacts, or classes, to better understand, alter, maintain, or document information about the system.

5.3.1 DFD Diagram:

Figure 5.3.1.1 – DFD Level 0 Diagram

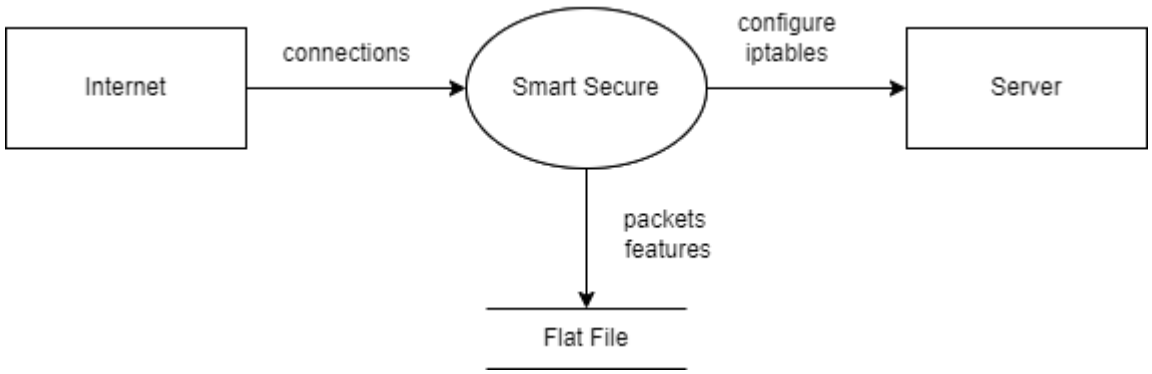
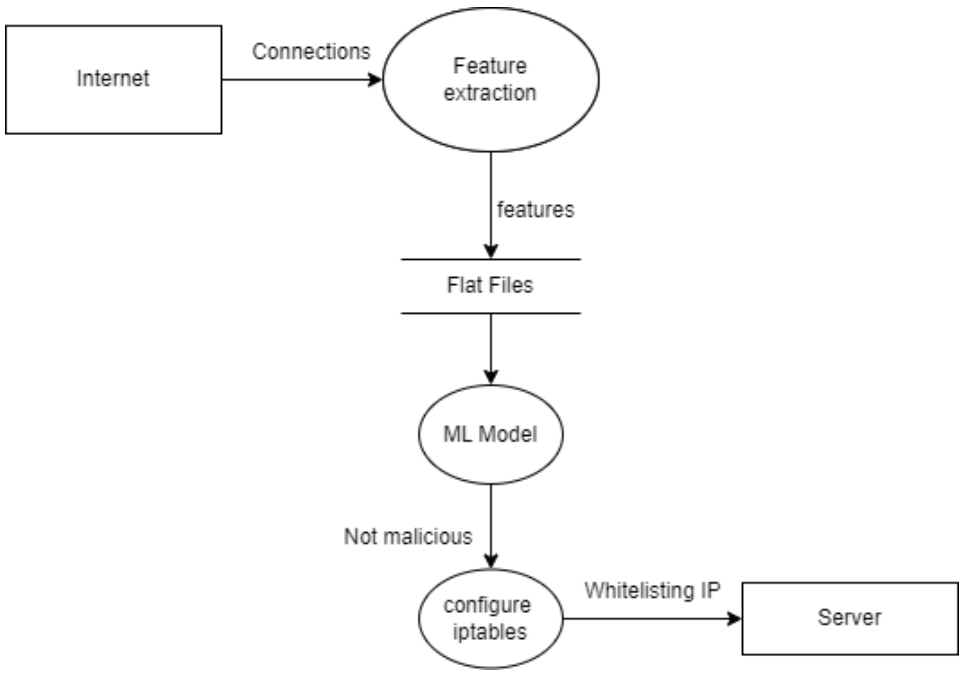


Figure 5.3.1.2 – DFD Level 1 Diagram



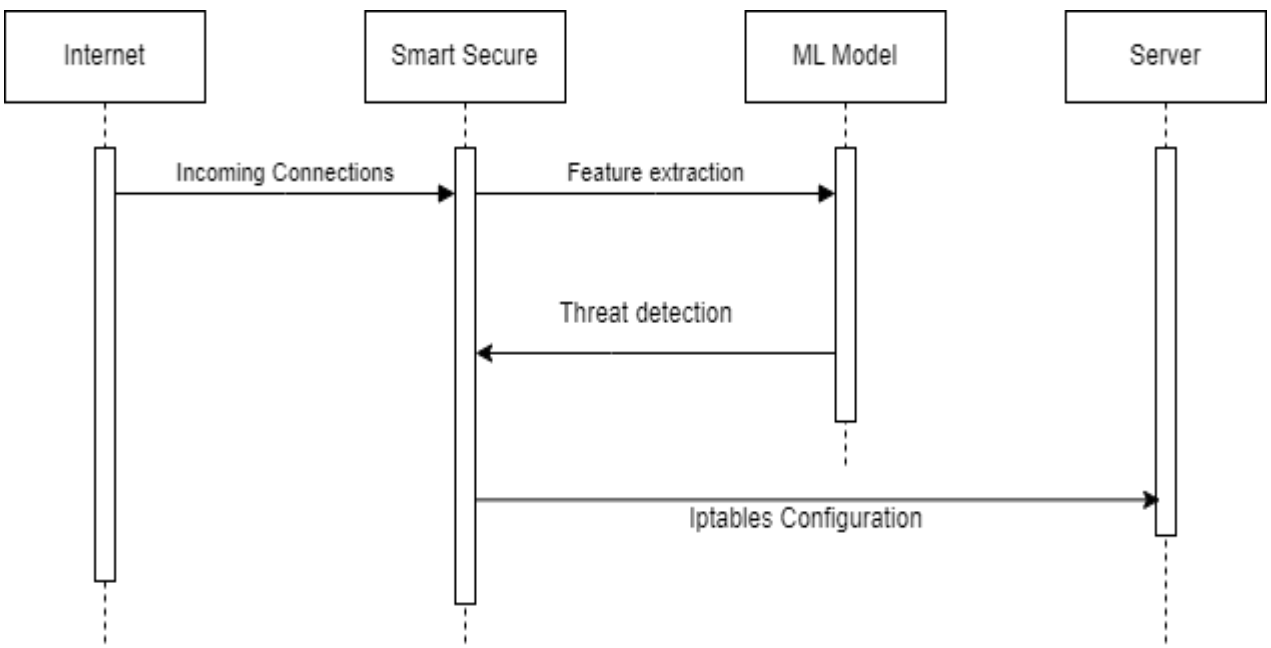
A DFD maps out the flow of information for any process or system. It uses defined symbols like

rectangles, circles, and arrows, plus short text labels, to show data inputs, outputs, storage points, and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews , to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. The data flow diagram gives clear a picture of the project and the movement of the data from one object to another object as detailed information about the process and functionalities available in the project.

5.4 Sequence Diagram

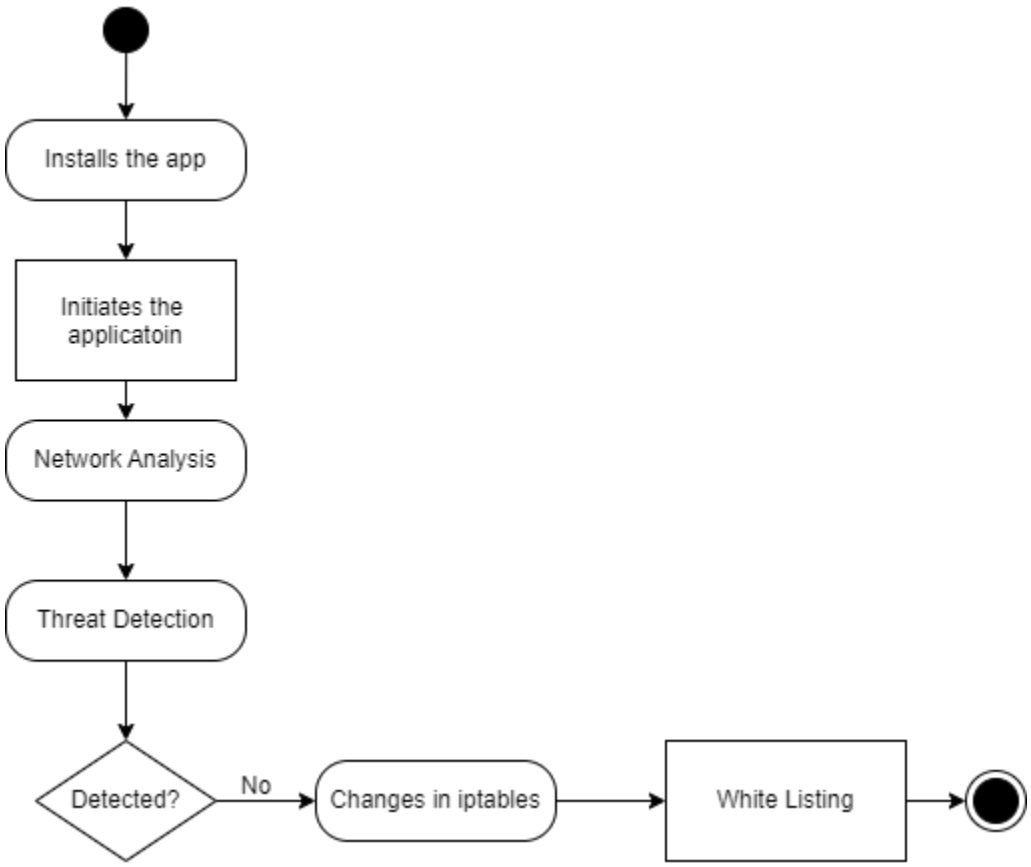
A sequence diagram simply depicts the interaction between objects in sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems. The above sequence diagram talks about the order in which happenings are taking place, and how the objects and their functions are dependent on each other. In the above diagram, objects are shown in the form of a rectangle shape whereas functionalities are represented in a flowing manner by using solid and dotted arrows.

Figure 5.4 – Sequence Diagram



5.5 Activity Diagram

Figure 5.5 – Activity Diagram



The activity diagram is another important diagram in UML to describe the dynamic aspects of the system. An activity diagram is a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc. The activity diagram represents the entire work of the application in the flow schema and illustrates each component of the project simply so that the user or the student will be able to understand the whole work of the project.

CHAPTER 6

Experimental Setup

6.1 Software and Hardware Setup:

Parameters	**Development Machine Configuration**	**Client Machine Configuration**
Operating System	Linux	Linux
Processor	Intel i7 CPU, Nvidia DGX Station A100	Multi-Core CPU
RAM	16 GB	Min. 8 GB
Software Dependencies	Python, pip, CICflowmeter, numpy, scapy, scipy, requests, IPtables	Python, pip, CICflowmeter, numpy, scapy, scipy, requests, IPtables

Table 6.1: Software and Hardware Setup Table

6.2 Project Setup:

- **CICFlowmeter Installation Script:**

```
#!/bin/bash
sudo apt-get install git -y
sudo apt-get install python3 -y
sudo apt install python3-pip
git clone https://gitlab.abo.fi/tahmad/cicflowmeter-py
cd cicflowmeter-py
pip install -r requirements.txt
sudo python3 setup.py install
```

- **Machine Learning:**

Machine learning (ML) is a field of study in artificial intelligence concerned with the development and study of statistical algorithms that can learn from data and generalize to unseen data, and thus perform tasks without explicit instructions. Recently, artificial neural networks have been able to surpass many previous approaches in performance.

- **Supervised Learning:**

Supervised learning (SL) is a paradigm in machine learning where input objects (for example, a vector of predictor variables) and a desired output value (also known as human-labeled supervisory signal) train a model. The training data is processed, building a function that maps new data on expected output values. An optimal scenario will allow for the algorithm to correctly determine output values for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way (see inductive bias). This statistical quality of an algorithm is measured through the so-called generalization error.

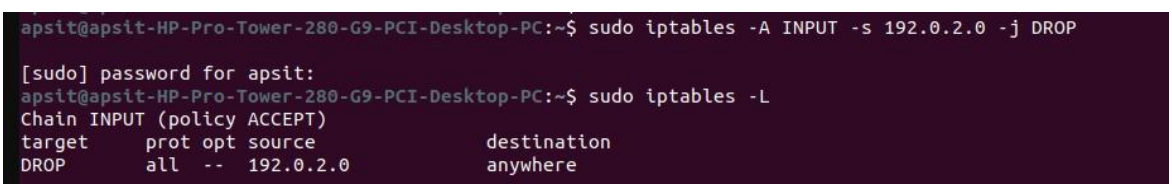
- **Iptables Setup commands:**

The iptables command in Linux is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel. It is used with the syntax, iptables [option] [paramter] [action]. It is a highly customizable command that can control a variety of parameters.

Example Commands:

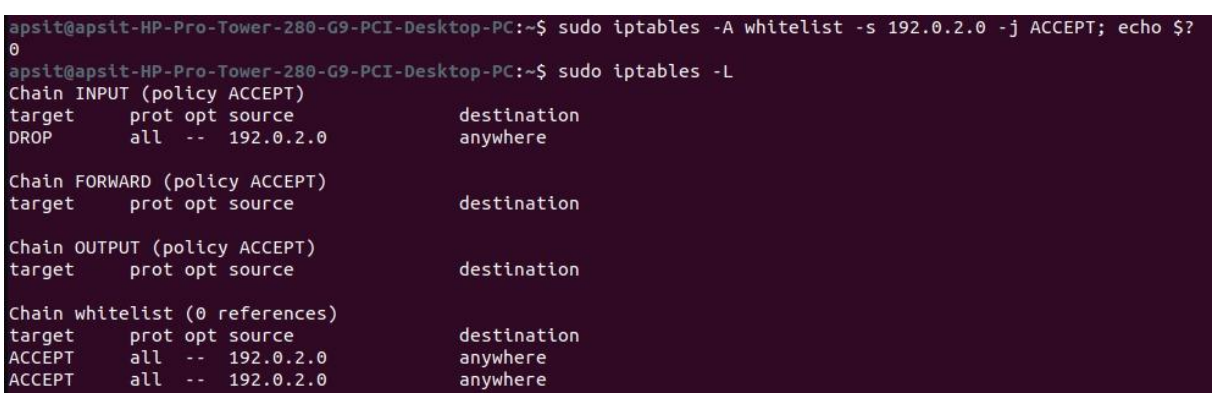
To add a rule to iptables, use the -A option followed by the chain name and the rule details. For example, to block all incoming traffic from a specific IP address (192.0.2.0 for instance), you can use:

Figure 6.1 – Iptables commands for blacklisting



For whitelisting a particular ip address in a network use the following command.

Figure 6.2 – Iptables commands for whitelisting



CHAPTER 7

Implementation and Result

7.1 Network Analysis using CICFlowmeter:

```
import subprocess
import os
import sys
import time

class CicflowmeterService:
    def __init__(self, interface):
        self.interface = interface
        self.process = None

    def start(self):
        if self.process is None or self.process.poll() is not None:
            self.process = self._run_cicflowmeter()
            print("cicflowmeter process started in the background.")
        else:
            print("cicflowmeter process is already running.")

    def stop(self):
        if self.process is not None and self.process.poll() is None:
            self.process.terminate()
            self.process.wait()
            print("cicflowmeter process stopped.")
        else:
            print("No cicflowmeter process is currently running.")

    def restart(self):
```

```
            self.stop()
            self.start()

        def _run_cicflowmeter(self):
            try:
                # Construct the command
                command = ["sudo", "cicflowmeter", "-i", self.interface, "-
c", "--dir", "output_flows/"]

                # Open the process in the background and detach it
                with open(os.devnull, 'w') as devnull:
                    return subprocess.Popen(command, stdout=devnull,
stderr=devnull, stdin=devnull, preexec_fn=os.setsid)
            except FileNotFoundError:
                print("Error: cicflowmeter command not found. Make sure it
is installed and in your system PATH.")
                return None

    def print_usage():
        print("Usage: python script_name.py <interface>
<start|stop|restart>")
        sys.exit(1)

    def main():
        if len(sys.argv) != 3:
            print_usage()

        interface = sys.argv[1]
        action = sys.argv[2].lower()

        if action not in ['start', 'stop', 'restart']:
            print("Error: Invalid action.")
            print_usage()

        service = CicflowmeterService(interface)

        if action == 'start':
            service.start()
        elif action == 'stop':
            service.stop()
        elif action == 'restart':
            service.restart()

    if __name__ == "__main__":
        main()
```

CICFlowmeter is a network flow analysis tool designed to dissect and interpret network traffic patterns. It allows users to capture and analyze network flows in real-time or from

pre-captured network traffic data. The tool provides insights into various aspects of network behavior, such as traffic volume, flow duration, and communication patterns between network endpoints. Additionally, CICFlowmeter offers flexibility in output formats and supports the generation of flow records in CSV format, among others.

This Python script serves as a user-friendly interface for utilizing CICFlowmeter, a tool designed for analyzing network flows. By prompting the user to input the network interface (e.g., eth0), it simplifies the execution of CICFlowmeter commands. Once provided, the script invokes CICFlowmeter with the specified interface and options, facilitating the generation of flow records in CSV format stored within the designated output directory.

7.2 Machine Learning Model Building:

7.2.1 Dataset for training and testing:

CICIDS2017 dataset contains benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). It also includes the results of the network traffic analysis using CICFlowMeter with labeled flows based on the time stamp, source, and destination IPs, source and destination ports, protocols and attack (CSV files). Also available is the extracted features definition.

Generating realistic background traffic was our top priority in building this dataset. We have used our proposed B-Profile system (Sharafaldin, et al. 2016) to profile the abstract behavior of human interactions and generates naturalistic benign background traffic. For this dataset, we built the abstract behaviour of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols.

The data capturing period started at 9 a.m., Monday, July 3, 2017 and ended at 5 p.m. on Friday July 7, 2017, for a total of 5 days. Monday is the normal day and only includes the benign traffic. The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. They have been executed both morning and afternoon on Tuesday, Wednesday, Thursday and Friday.

We have only considered DoS and BENIGN values from the dataset according to our use case and further attacks can be classified in future scope of our project.

7.2.2 Data normalization and encoding:

```
# Min-max normalization
numeric_features = df.dtypes[df.dtypes != 'object'].index
df[numeric_features] = df[numeric_features].apply(
lambda x: (x - x.min()) / (x.max()-x.min()))

# Fill empty values by 0
df = df.fillna(0)

labelencoder = LabelEncoder()
df.iloc[:, -1] = labelencoder.fit_transform(df.iloc[:, -1])
```

Initially, we applied min-max normalization to numeric features, ensuring that their values are scaled to a range between 0 and 1. This normalization method is achieved by subtracting each feature's minimum value and dividing by the range between the maximum and minimum values. Subsequently, any missing values within the DataFrame are filled with zeros. Lastly, categorical labels in the last column of the DataFrame are encoded into numerical representations using the `LabelEncoder()` function. This process assigns a unique integer to each distinct category, facilitating the model's ability to work with categorical data. Overall, these preprocessing steps aim to standardize and prepare the dataset for use in machine learning algorithms.

7.2.3 Feature Extraction:

```
rf_feature = rf.feature_importances_


Sum = 0
fs = []
ts= []
for i in range(0, len(f_list)):
    Sum = Sum + f_list[i][0]
    fs.append(f_list[i][1])
    if(f_list[i][1] in features):
        ts.append(f_list[i][1])
    if Sum>=0.9:
```

```
            break
print(len(fs))
```

This code snippet iterates through a list of tuples `f_list`, accumulating the values of the first element in each tuple until the sum exceeds or equals 0.9. During each iteration, it appends the second element of the tuple to a list `fs`. If the second element is also present in another list `features`, it appends it to a separate list `ts`. Finally, it prints the length of the `fs` list. This selects the important features from top-importance to bottom-importance until the accumulated importance reaches 0.9 (out of 1).

After feature extraction process random forest algorithm deduced these columns as most important ones -

- Fwd Packet Length Mean: Average size of forward packets sent.
- Avg Fwd Segment Size: Average size of segments in forward packets.
- act_data_pkt_fwd: Number of actual data packets sent forward.
- Fwd IAT Total: Total time between two consecutive forward packets.
- Fwd Packet Length Max: Maximum size of a forward packet.
- Init_Win_bytes_forward: Initial window size in bytes for the forward direction.
- Total Length of Fwd Packets: Total length of all forward packets.
- Fwd IAT Max: Maximum time between two consecutive forward packets.
- Fwd IAT Mean: Average time between two consecutive forward packets.
- Fwd Header Length.1: Length of header in forward direction.
- Total Fwd Packets: Total number of forward packets.
- Fwd Header Length: Length of header in forward direction.

- Subflow Fwd Bytes: Bytes of data in forward subflows.
- Subflow Fwd Packets: Number of packets in forward subflows.
- Bwd Packet Length Min: Minimum size of a backward packet.
- Fwd IAT Std: Standard deviation of time between two consecutive forward packets.
- Fwd Packet Length Std: Standard deviation of sizes of forward packets.
- Fwd Packet Length Min: Minimum size of a forward packet.
- URG Flag Count: Count of urgent packets.
- Init_Win_bytes_backward: Initial window size in bytes for the backward direction.
- Bwd IAT Total: Total time between two consecutive backward packets.
- min_seg_size_forward: Minimum segment size allowed in the forward direction.
- Bwd Header Length: Length of header in backward direction.

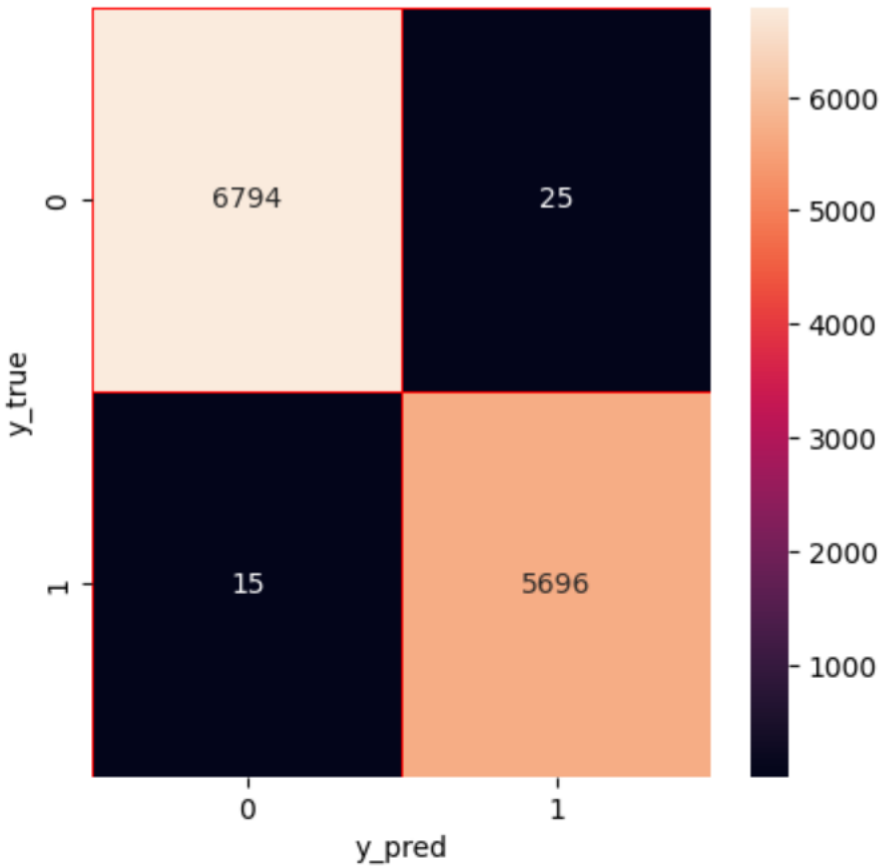
7.2.4 Training and Testing of Dataset using Random Forest Algorithm:

```
X = df.drop(['Label'],axis=1).values
y = df.iloc[:, -1].values.reshape(-1,1)
y=np.ravel(y)
X_train, X_test, y_train, y_test = train_test_split(X,y,
train_size = 0.7, test_size = 0.3, random_state = 42,stratify = y)
# Random Forest training and prediction
rf = RandomForestClassifier(random_state = 0)
rf.fit(X_train,y_train)
rf_score=rf.score(X_test,y_test)
y_predict=rf.predict(X_test)
y_true=y_test
print('Accuracy of RF: '+ str(rf_score))
precision,recall,fscore,none=
precision_recall_fscore_support(y_true, y_predict,
average='weighted')
print('Precision of RF: '+(str(precision)))
print('Recall of RF: '+(str(recall)))
```

```
print('F1-score of RF: '+(str(fscore)))
print(classification_report(y_true,y_predict))
cm=confusion_matrix(y_true,y_predict)
f,ax=plt.subplots(figsize=(5,5))
sns.heatmap(cm,annot=True,linewidth=0.5,linecolor="red",fmt=".0f",
ax=ax)
plt.xlabel("y_pred")
plt.ylabel("y_true")
plt.show()
```

Initially, the dataset is split into training and testing sets using a 70-30 ratio, ensuring randomness is maintained with a specified random seed. The classifier is instantiated, trained on the training data, and subsequently used to predict labels for the test data. Performance metrics including accuracy, precision, recall, and F1-score are computed and printed to assess the model's effectiveness. Additionally, a detailed classification report is generated, providing insights into the classifier's performance across different classes. Finally, a confusion matrix along with a heatmap visualization is created to visualize the distribution of predicted versus actual labels, aiding in the interpretation of the model's behavior.

Figure 7.1 – Confusion Matrix of Random Forest



Above is the diagram of confusion matrix after testing the random forest model on the test dataset split of 30%. We can infer the following:

The model provides us with an accuracy of 99.68% on the testing set and we can now use this model to predict the results from CICFlowmeter.

7.2.5 Pickle file for real time prediction:

```
import pickle
pickle.dump(rf, open('model.pkl','wb'))
```

The above code creates a pickle file to use in production for faster prediction in real time.

7.3 Configuring the Server:

7.3.1 Parsing the predicted results:

```
import csv
import time
def create_ip_txt(csv_file, output_file):
    while True:
        genuine_ips = []
        try:
            with open(csv_file, 'r') as file:
                reader = csv.DictReader(file)
                for row in reader:
                    ip_address = row['IP']
                    label = row['Label']
                    if label.lower() == 'good':
                        genuine_ips.append(ip_address)

            with open(output_file, 'w') as outfile:
                for ip in genuine_ips:
                    outfile.write(ip + '\n')

            print(f"TXT file '{output_file}' created with genuine
IPs.")
        except FileNotFoundError:
            print("File not found. Please provide a valid CSV
file.")

        time.sleep(300)  # Run every 5 minutes

def main():
    csv_file = "input.csv"
    output_file = "output.txt"
    create_ip_txt(csv_file, output_file)

if __name__ == "__main__":
    main()
```

This Python script continuously monitors a predefined CSV file containing IP addresses and their corresponding labels. It specifically looks for IP addresses labeled as "good", indicating that they are considered safe or genuine. Once identified, these genuine IP addresses are written to a TXT file. The script operates in an infinite loop, periodically checking the CSV file for any updates. If new genuine IP addresses are found, they are appended to the TXT file. This ensures that the TXT file remains up-to-date with the latest list of genuine IP addresses, allowing for easy access and reference.

7.3.2 Whitelisting good IP addresses:

```
import subprocess
import time

def whitelist_ips(txt_file):
    while True:
        try:
            with open(txt_file, 'r') as file:
                for line in file:
                    ip_address = line.strip()
                    # Add IP address to whitelist using iptables
                    subprocess.run(['iptables', '-A', 'INPUT', '-
s', ip_address, '-j', 'ACCEPT'])
                    print(f"Whitelisted IP address {ip_address}.")

            time.sleep(300)  # Run every 5 minutes
        except FileNotFoundError:
            print("File not found. Please provide a valid TXT
file.")

def main():
    txt_file = "path/to/input.txt"
    whitelist_ips(txt_file)

if __name__ == "__main__":
    main()
```

This Python script continuously monitors a predefined TXT file containing a list of genuine IP addresses. It reads each IP address from the TXT file and whitelists them using iptables, a firewall administration tool. Whitelisting an IP address means allowing it to bypass certain security measures, thereby ensuring that network traffic from these IP addresses is not blocked or restricted. The script runs in an infinite loop, periodically checking the TXT file for any updates. If new IP addresses are added to the TXT file, they are automatically whitelisted, ensuring that the network remains accessible to trusted sources.

7.4 Results and conclusion

Parameters	Machine learning Approach (Random Forest)	Traditional Log-Based Approach (OSSEC)
Accuracy and Detection Rate	Random Forest models can achieve high accuracy by combining multiple decision trees, which helps in effectively identifying patterns and anomalies associated with DoS attacks. The model can adapt to new attack patterns by retraining with new data, ensuring ongoing accuracy improvements.	OSSEC relies on predefined rules and signatures to detect anomalies, which might not capture new or unknown attack patterns. The accuracy heavily depends on the quality and comprehensiveness of predefined rules, potentially leading to false positives or negatives.
Scalability	Random Forest models can handle large datasets efficiently and scale with increasing data volume without significant performance degradation.	OSSEC might face scalability issues when dealing with large-scale environments, as the processing of logs based on predefined rules can become computationally intensive.
Adaptability to New Threats	ML models like Random Forest can adapt to new and evolving attack patterns by learning from new data, making them more resilient to zero-day attacks.	OSSEC requires manual updating of rules and signatures to detect new threats, which can lead to delays in responding to emerging attack vectors.

False Positive Rate	Random Forest models can minimize false positives by considering multiple decision trees and evaluating the consensus of predictions, leading to more reliable detection results.	OSSEC might produce false positives due to the static nature of predefined rules, especially when dealing with complex or legitimate traffic that resembles attack patterns
Complexity and Maintenance	While ML models require initial training and tuning, they can reduce the maintenance overhead in the long run by automating the detection process and adapting to changes in the environment.	OSSEC requires continuous manual rule management and updates to maintain effectiveness, which can be time-consuming and resource-intensive.

Table 7.1: Results and conclusion table

In conclusion, the ML approach using Random Forest for DoS detection offers several advantages over traditional log-based methods like OSSEC in terms of accuracy, scalability, adaptability to new threats, and reduced false positive rates. While the initial setup and training of ML models might require expertise, the automation and adaptability benefits make them a more effective and efficient choice for modern cybersecurity challenges.

By adopting an ML-based approach, organizations can enhance their DoS detection capabilities, improve response times, and better protect their infrastructure from evolving cyber threats.

CHAPTER 8

Project Plan

1 Gantt Chart

Figure 8.1 – Gantt Chart

CHAPTER 9

Conclusion

In conclusion, our project endeavors to address the pervasive threat of Distributed Denial of Service (DDoS) attacks by developing a sophisticated Machine Learning (ML) based Intrusion Prevention System (IPS). Through meticulous analysis of network traffic patterns and the implementation of advanced algorithms, our system can swiftly detect and mitigate DDoS attacks while minimizing disruption to legitimate network traffic. The integration of multithreading technology optimizes system performance, ensuring maximum uptime for servers. Additionally, our approach of whitelisting genuine IP addresses rather than blacklisting malicious ones reduces latency, enhancing the overall efficiency of our solution. By combining these elements, our project aims to fortify network security, safeguarding against evolving cyber threats and ensuring uninterrupted service availability for organizations and users.

CHAPTER 10

Future Scope

1. **Detecting More Types of Attacks**: We can improve our system to recognize and stop other kinds of cyber threats, not just DDoS attacks. This could include things like viruses, hijacking, or stealing data.

2. **Making Things Run Even Smoother**: We can improve how our system handles multiple tasks at once, making sure everything runs quickly and efficiently. This could mean making sure our servers stay online and responsive even when there's a lot of activity.

3. **Adapting to New Threats**: Our system could learn from new threats and automatically change how it works to keep up with them. This means it could get better at stopping attacks without us having to make constant updates.

4. **Working with Other Security Tools**: We could connect our system to other tools that track cyber threats. This way, it could use information from those tools to do an even better job of spotting and stopping attacks.

5. **Growing with the Network**: We want our system to be able to handle more and more traffic as networks get bigger. This might involve using cloud technology or spreading out the work across different computers to keep things running smoothly.

References

[1] Richardson, T., Stafford-Fraser, Q., Wood, K.R., Hopper, A.: Virtual network computing. In: Advances in Science and Engineering Technology International Conferences (ASET) 1998. https://ieeexplore.ieee.org/document/656066.

[2] Kibe, S., Koyama, T., Uehara, M.: The Evaluations of Desktop as a Service in an Educational Cloud. In: International Conference on Network-Based Information Systems 2012. https://ieeexplore.ieee.org/abstract/document/6354895.

[3] Kim, S., Choi, J., Kim, S., Kim, H.: Cloud-based virtual desktop service using lightweight network display protocol. In: International Conference on Information Networking (ICOIN) 2016. https://ieeexplore.ieee.org/document/7427070.

[4] "Docker Whitepaper" [Online]. https://www.docker.com/static/build-modern-secure-apps-at-scale-white-paper.pdf.

[5] Yadav, R.R., Sousa, E.T.G., Callou, G.R.A.: Performance Comparison Between Virtual Machines and Docker Containers. In: IEEE Latin America Transactions 2018. https://ieeexplore.ieee.org/document/8528247.

[6] Sotiriadis, S., Bessis, N., Xhafa, F., Antonopoulos, N.: Cloud Virtual Machine Scheduling: Modelling the Cloud Virtual Machine Instantiation. In: Sixth International Conference on Complex, Intelligent, and Software Intensive Systems 2012. https://ieeexplore.ieee.org/document/6245618.

[7] Benedicic, L., Cruz, F.A., Madonna, A., Mariotti, K.: Highly Scalable Docker Containers for HPC Systems 2019. https://link.springer.com/chapter/10.1007/978-3-030-34356-9_5.

[8] Chung, M.T., Quang-Hung, N., Nguyen, M.T., Thoai, N.: Using Docker in high-performance computing applications. In: IEEE Sixth International Conference on Communications and Electronics (ICCE) 2016. https://ieeexplore.ieee.org/document/7562612.

[9] Kangjin, W., Yong, Y., Ying, L., Hanmei, L., Lin, M.: FID: A Faster Image Distribution System for Docker Platform. In: IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W) 2017. https://ieeexplore.ieee.org/abstract/document/8064123.

[10] Fuzi, M.F.M., Hamid, R.S., Ahmad, M.A.: Virtual desktop environment on Cloud computing platform. In: IEEE 5th Control and System Graduate Research Colloquium 2014. https://ieeexplore.ieee.org/document/6908700.

[11] Dhall, M., Tan, Q.: A Profitable Hybrid Desktop as a Service Solution. In: IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA) 2019. https://ieeexplore.ieee.org/abstract/document/8725659.

[12] Huang, Z., Wu, S., Jiang, S., Jin, H.: FastBuild: Accelerating Docker Image Building for Efficient Development and Deployment of Container. In: 35th Symposium on Mass Storage Systems and Technologies (MSST) 2019. https://ieeexplore.ieee.org/abstract/document/8890109.

The paper entitled "Streamlining server security against cyber threats" is waiting for approval in CIS 2024 (https://scrs.in/conference/cis2024) - CHRIST University Bangalore | Proceedings in SCOPUS indexed Springer Series LNNS.

Streamlining Server Security against Cyber Threats

Abstract. The project "Streamlining Server Security against Cyber Threats" addresses the increasingly critical need for robust security measures in servers to protect against a wide range of cyber threats such as DOS and DDOS attacks. With the growing reliance on server solutions for hosting websites, applications and data, ensuring the security of these servers is of paramount importance.

This project aims to develop a comprehensive security framework specifically tailored to server environments, focusing on proactive threat mitigation and adaptive defense mechanisms. The proposed framework combines cutting-edge cybersecurity technologies, best practices, and automation to create a robust defense strategy that streamlines server security.

This project introduces an innovative approach to server security through the design and development of an automated server security tool that leverages Machine Learning (ML) to enhance security, detect potential threats, and provide an intuitive user experience for server administrators.

Keywords: DDOS, Machine Learning, Server Administrators.

1 Introduction

Within the field of information technology, servers are the fundamental building blocks of digital infrastructure. They are specialized software or computer systems made to handle data, react to requests, and provide services to client devices. Servers are the fundamental workhorses of modern data processing and communication; they make it possible to get online pages, send emails, store important business data, and carry out computational operations.

For people and companies that host their digital assets on servers, security is crucial. Because it is isolated, server hosting has built-in security benefits. To secure the server, though, more precautions must be taken. These include using secure, one-of-a-kind passwords, putting strong firewall rules into place, deploying intrusion detection systems, and updating the operating system and software regularly to fix vulnerabilities. Encrypting sensitive data, limiting user capabilities, and keeping an eye on server logs for odd activity are all crucial components of effective access control.

Among the various threats that a server may face, DOS/DDOS. There isn't a tool that stops it from happening yet; the available ones either use log analysis to identify threats before blocking IP addresses (like fail2ban) or use intrusion detection systems (IDS) to notify system administrators solely of system intrusions. We use a slightly different approach in that we want to stop it from entering the system rather than waiting for an initial attack to occur. The three parts of this strategy are packet capture, applying a machine learning algorithm to detect threats, and then configuring iptables to safeguard the system.

Our method begins with a thorough study of network traffic. Using the popular utility Scapy, we begin by gathering network traffic statistics. Scapy captures crucial details about every network packet, such as its size, origin, destination, and other details. Packets consist mostly of two parts: the payload and the packet headers. All the essential information needed by network equipment to determine how to handle each packet is contained in the packet headers. The IP packet count and the source and destination addresses are the most crucial factors. This information serves as the basis for our investigation and feeds into our detection model.

ML algorithms are used in the second step to accurately identify possible threats. We have employed the Random Forest Algorithm, a potent cybersecurity tool for identifying and reducing cyber threats. This machine learning method creates a tree-like model of potential attack scenarios to assist security professionals in making well-informed decisions. Incoming data is analyzed by Random Forest, which then applies a set of predefined rules or conditions to determine its decisions. Random Forest can evaluate user

behavior, network traffic, or system anomalies in the context of cyber threat detection to categorize and identify possible threats. The algorithm can identify patterns linked to known threats by analyzing past data, and it can then act appropriately in real-time by blocking suspicious activity and starting defensive measures. The Random Forest Algorithm is a vital tool in the ongoing fight against changing cyber threats because it provides the benefits of automation and adaptability.

Based on the algorithm's output, we can determine the appropriate course of action. By whitelisting all benign IP addresses and blacklisting malicious ones, the system operates without delay because only newly added IP addresses are examined each time. Additionally, logs are kept up to date for future reference and upkeep.

To address the latency issues, we are utilizing the power of parallel computing, which makes the system robust and delay-free. All of these tasks must be completed before the user is served with the website.

1.1 Existing System

Fail2ban is designed to keep an eye on common service logs to identify patterns in failed authentication attempts. When fail2ban is set up to watch a service's logs, it examines a filter that has been set up especially for that service. The filter uses sophisticated regular expressions to detect authentication failures for that particular service. A popular templating language for pattern matching is regular expressions. These regular expression patterns are defined within the failregex internal variable.

Fail2ban comes with filter files for popular services by default. A predetermined action is performed for each service, such as a web server, whose log matches the failregex in its filter. The administrator can customize the action variable to do various functions based on their preferences. Changing the local firewall rules to prohibit the offending host or IP address is the default course of action. This operation can be expanded to, for instance, email your system administrator. The default ban period is ten minutes, and action is initiated by default if three authentication failures are found in ten minutes.

You can adjust this. Upon starting the service, fail2ban generates a new chain of firewall rules when utilizing the built-in iptables firewall. All TCP traffic aimed at port 22 is routed to the new chain when a new rule is added to the INPUT chain. It adds a single rule that goes back to the INPUT chain to the new chain. If the Fail2ban service is terminated, the chain and related rules are eliminated.
Fail2ban uses log files to store the packets it has captured and processes on the information after a connection is over. This allows the attacker to attack atleast once on the target machine which is a big limitation. It also uses tradtional file systems to store the data, we can use a machine learning model instead. To overcome all these problems, parallel processing on the system can be a plus point and a modern approach in tackling the modern attackers.

2 Input Data

Live packet capturing, a dynamic process that comprises the real-time interception and analysis of network packets moving over a computer network, is one of the input data types. This method uses Python tools such as Scapy to record packets during transmission, giving an image of the current network conversation. The dataset includes elements like "Bwd Packet Length Std," "Average Packet Size," and "Fwd IAT Max," among other packet attributes. These characteristics provide insights into network behavior and make it possible to identify connections, protocol usage, and possible security abnormalities when they are seen during live capturing. This dynamic information is essential for network analysis, security applications, and monitoring since it makes it easier to continuously evaluate the performance and health of the network.

3 General Approach

The incoming packets from clients are captured using the Scapy library. The feature extraction module extracts these packets features like source ip, no. of packets, request duration, etc. The next module is the ML model which segregates the clients into Blacklist or Whitelist. The iptables are configured in the next module based on the Blacklist. This whole architecture is shown in **Fig.1**

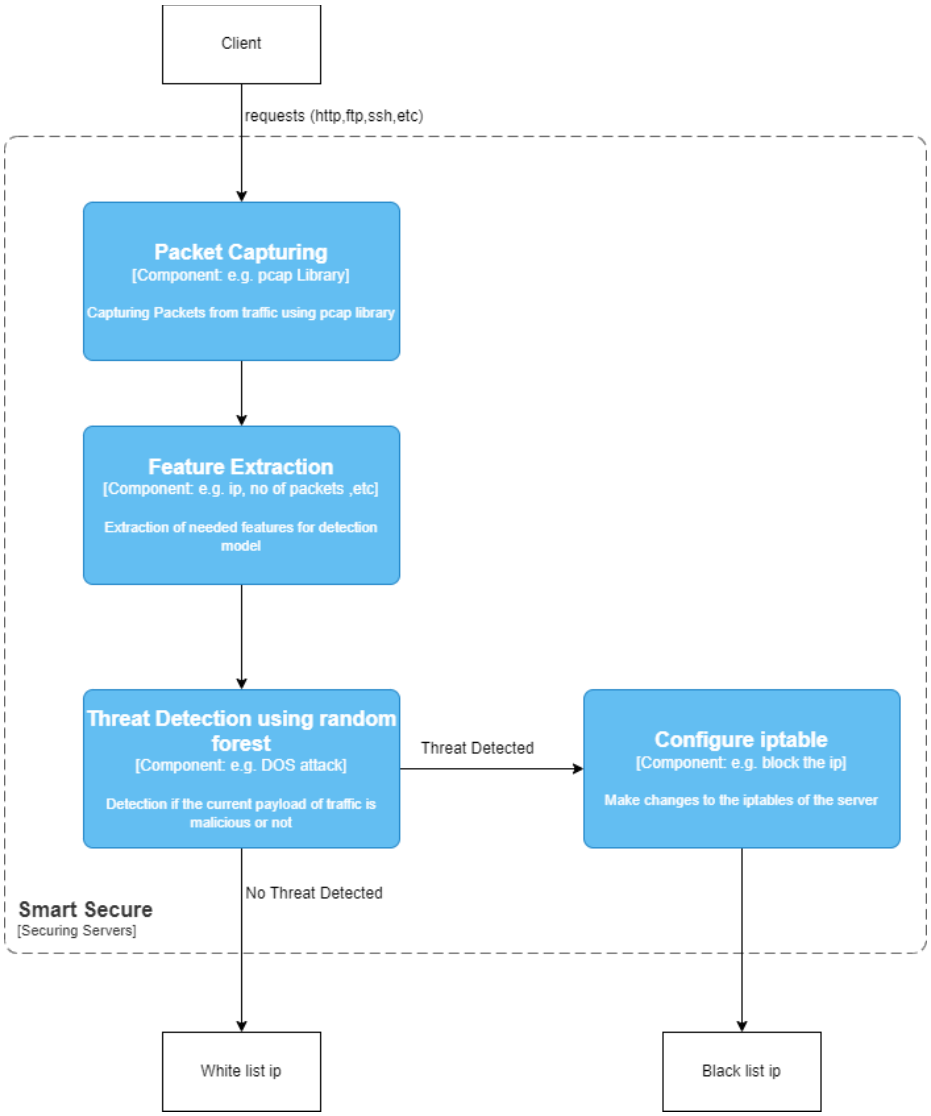


Fig. 1. Architecture Diagram

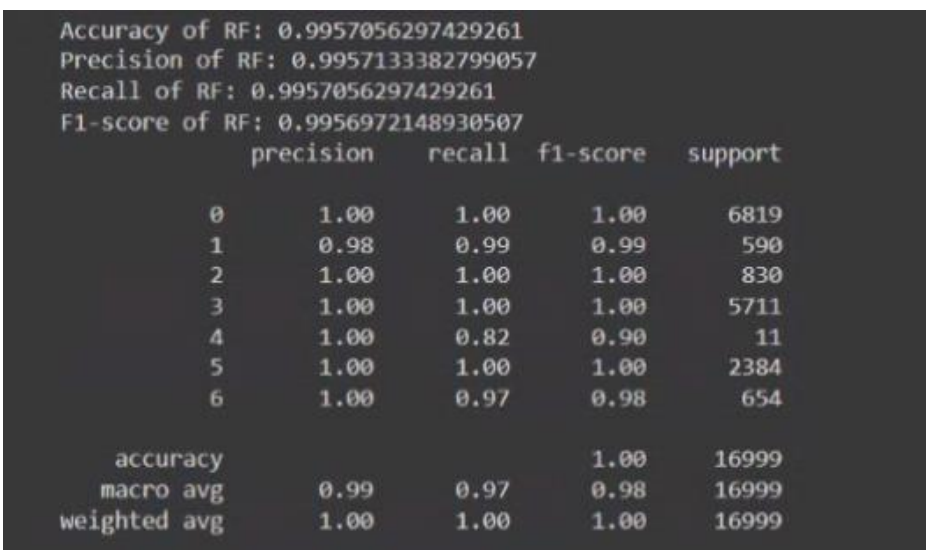
3.1 Packet Capturing

Clients send a variety of requests that initially enter the system before reaching the server. These incoming requests are captured using SCAPY (Packet Capture) and stored in buffers for subsequent processing. These captured packets are then efficiently stored in buffers, serving as a temporary repository for data awaiting further processing. SCAPY files, instrumental in this process, are meticulously generated through a dedicated program, creating structured data files that encapsulate the intricate details of the packets coursing through the network. These files are an indispensable resource for network analysts and administrators, as they provide a comprehensive dataset for evaluating the

network's behavior and characteristics. Subsequently, these SCAPY files become essential inputs for subsequent system modules, which are tasked with extracting and isolating the specific features and information required for in-depth analysis, security monitoring, or other critical functions within the network environment. This comprehensive process plays a vital role in ensuring the efficient and secure operation of networks, allowing for the meticulous analysis and management of data flows in modern digital landscapes.

3.2 Feature Extraction

In the process of network traffic analysis and security, a fundamental step involves capturing the requests initiated by clients through the use of packet capture (Scapy) techniques. These captured packets serve as a valuable source of data for subsequent analysis. Each type of request, be it HTTP, FTP, SSH, or any other protocol, is characterized by a specific header format that encapsulates essential information pertaining to the communication session. This information encompasses various attributes and characteristics of the data packets, such as source and destination addresses, packet size, payload content, and more. To enable the application of machine learning models in the context of client classification into Blacklist and Whitelist categories, the next crucial step is feature extraction. During this phase, the relevant features required by the machine learning model are meticulously isolated from the packet headers. For e.g. source ip, destination ip, no. of packets, duration of request, etc. These extracted features serve as the foundation for the model's decision-making process, facilitating the classification of clients based on their network behavior, ultimately enhancing the security of the system by identifying potential threats and distinguishing them from legitimate users. This information we have fed to the different algorithms such as XGboost, Random forest, Decision Tree, and Extra Tree. Among all the algorithms best result are shown in **Fig2.**



Accuracy of RF: 0.9957056297429261
Precision of RF: 0.9957133382799057
Recall of RF: 0.9957056297429261
F1-score of RF: 0.9956972148930507

	precision	recall	f1-score	support
0	1.00	1.00	1.00	6819
1	0.98	0.99	0.99	590
2	1.00	1.00	1.00	830
3	1.00	1.00	1.00	5711
4	1.00	0.82	0.90	11
5	1.00	1.00	1.00	2384
6	1.00	0.97	0.98	654
accuracy			1.00	16999
macro avg	0.99	0.97	0.98	16999
weighted avg	1.00	1.00	1.00	16999

Fig. 2. Random Forest

The robustness of a Random Forest (RF) classification model's capacity to reliably identify instances in a dataset is demonstrated by its performance measures, which display remarkable accuracy and precision. With an accuracy of 99.57%, the model's overall efficacy is confirmed by the percentage of correctly classified cases among all forecasts. The precision of the model, which is also 99.57%, indicates how effectively it can distinguish positive occurrences from all the instances it anticipated to be positive. Furthermore, the recall, which is likewise 99.57%, highlights how well the model captures the majority of true positive cases while reducing false negatives. With a harmonic mean of precision and recall of 99.57%, the F1-score provides a fair evaluation of the model's overall performance.Together, these high-performance measures indicate that the Random Forest model performs exceptionally well in terms of accuracy and precision, proving its dependability in producing correct predictions with a low percentage of false positives and false negatives.

3.3 Multi-Threading

When clients initiate requests, a series of critical operations, such as packet capturing and feature extraction, are executed on the incoming packets. Moreover, the machine learning (ML) model, responsible for deciding whether a packet should be classified as Blacklisted or Whitelisted, requires a certain amount of processing time for each packet analysis. Unfortunately, these collective processes can introduce a noticeable delay in handling the clients' requests, potentially impacting their overall experience. To mitigate this delay and ensure a smoother client interaction, the implementation of multi-threading proves indispensable. By utilizing multi-threading, the delay can be significantly reduced, if not virtually eliminated. This approach involves the creation of multiple independent threads or processes, each equipped with its own input buffer. These buffers serve as temporary memory storage for incoming requests, allowing the ML model to operate in parallel. Consequently, the model doesn't have to wait for the arrival of each individual request, enabling rapid and concurrent processing of multiple requests, thus effectively minimizing the delay and enhancing the overall efficiency and responsiveness of the system, ultimately resulting in a more seamless and user-friendly client experience.

4 Expected Outcome

The project aimed at streamlining Server security against cyber threats through the development of an automated security tool, leveraging AI-driven threat detection and a user-friendly interface. After a rigorous evaluation of various machine learning algorithms for threat detection, random forest emerged as the optimal choice. We aim to develop a comprehensive command-line system capable of configuring server security in alignment with established guidelines while simultaneously executing threat detection on the server to offer solutions for mitigating vulnerabilities. The system will be designed with the capability to effectively categorize clients into either Blacklist or Whitelist groups, serving as a robust security measure. For Blacklist clients, the system will ensure that their requests are promptly blocked through the use of iptables, thereby preventing any potentially harmful or unauthorized access to the server. Conversely, for clients in the Whitelist category, their requests will be allowed access to the server while also undergoing continuous surveillance. This vigilant monitoring enables the system to promptly detect and respond to any anomalous or malicious activities, providing an added layer of protection and ensuring the integrity and security of the server's operations.

5 References

1. D. Saxena, I. Gupta, R. Gupta, A. K. Singh and X. Wen, "An AI-Driven VM Threat Prediction Model for Multi-Risks Analysis-Based Cloud Cyber- security," in IEEE Transactions on Systems, Man, and Cybernetics: Systems
2. Wang, K., Stolfo, S.J. (2004). Anomalous Payload-Based Network Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds) Recent Advances in Intrusion Detection. RAID 2004. Lecture Notes in Computer Science, vol 3224. Springer
3. Alshammari, A., Aldribi, A. Apply machine learning techniques to detect malicious network traffic in cloud computing. J Big Data 8, 90 (2021)
4. Souri, A., Hosseini, R. A state-of-the-art survey of malware detection approaches using data mining techniques. Hum. Cent. Comput. Inf. Sci. 8, 3 (2018)
5. SYN flooding attack detection by TCP handshake anomalies Martine Bellaïche, Jean-Charles Grégoire 16 August 2011
6. Detection of DNS DDoS Attacks with Random Forest Algorithm on Spark Liguo Chena, Yuedong Zhang, Qi Zhao, Guanggang Geng , ZhiWei Yan
7. Automation System for Validation of Configuration and Security Compliance in Managed Cloud Services Trieu C. Chieu Manas Singh Chunqiang Tang, Mahesh Viswanathan, Ashu Gupta
8. Overview of use of Decision tree algorithms in machine learning Arundhati Navada; Aamir Nizam Ansari; Siddharth Patil; Balwant A. Sonkamble
9. A Security Aspects in Cloud Computing Gurudatt Kulkarni & Jayant Gambhir, Tejswini Patil, Amruta Dongare
10. Preventing DDoS attacks on internet servers exploiting P2P systems Author links open overlay panelXin Sun, Ruben Torres, Sanjay Rao