

# **Network Vigilance: AI-Powered Multi-Layered APT Protection**

---



**By:**

**Anas Mustafa Hashmi**  
37520

**Kabeer Ahmed**  
27970

**Yasir Mahmood**  
36546

**Supervised by:**  
**Mr. Syed Yawar Abbas**  
(Lecturer)

**Co-Supervised by:**  
**Dr. Javaid Iqbal**  
(Assistant Professor)

**Faculty of Computing**  
**Riphah International University, Islamabad**  
**Spring 2025**

**Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfillment of the Requirement for the Award  
of the Degree of  
Bachelors of Science in Cyber Security**

**Faculty of Computing**

**Riphah International University, Islamabad**

Date: 14 May 2025

## Final Approval

This is to certify that we have read the report submitted by *Anas Mustafa Hashmi (CMS# 37520)*, *Kabeer Ahmed (CMS# 27970)*, *Yasir Mahmood (CMS# 36546)*, for the partial fulfillment of the requirements for the degree of **Bachelor of Science in Cyber Security (BS CYS)**. It is our judgment that this report is of sufficient standard to warrant its acceptance by **Riphah International University, Islamabad** for the degree of **Bachelor of Science in Cyber Security (BS CYS)**.

### Committee:

1

---

Mr. Syed Yawar Abbas  
(Supervisor)  
Dr. Jawaaid Iqbal  
(Assistant Professor)

2

---

Dr. Musharaf Ahmed  
(Head of Department/Chairman)

## Declaration

We hereby declare that the project report titled “**Network Vigilance: AI-Powered Multi-Layered APT Protection**” is a result of our own independent work and has not been copied, in whole or in part, from any other source. All information, data, and content presented in this report have been compiled, implemented, and documented solely through our own efforts. This work was completed under the valuable guidance and supervision of **Mr. Syed Yawar Abbas** and **Dr. Javaid Iqbal**, to whom we express our sincere gratitude. Should any portion of this report be discovered to be plagiarized or improperly sourced, we accept full responsibility and the consequences thereof.

---

**Anas Mustafa Hashmi**  
**37520**

---

**Kabeer Ahmed**  
**27970**

---

**Yasir Mahmood**  
**36546**

## Dedication

In the name of Allah, the Most Gracious, the Most Merciful, we dedicate this Final Year Project to those whose unwavering support, encouragement, and belief have been our constant sources of strength throughout our academic journey. To our cherished parents, who have been our foundation of support and sacrifice—thank you for your endless love, prayers, patience, and for always believing in our potential, even during the toughest times. Your trust in us gave us the courage to aim high and the perseverance to keep going. To our siblings and family members, who stood by us, listened to our frustrations, celebrated every small success, and offered unwavering support during moments of doubt—we are deeply grateful for your emotional and moral encouragement. To our teachers and mentors, especially our respected supervisor, who provided us with guidance, fostered independent thinking, and nurtured our technical skills—we owe much of our success to your dedication and efforts. To our friends and classmates, whose collaboration, camaraderie, and shared experiences—especially the late-night coding sessions—made this journey not only manageable but memorable, we thank you for walking beside us through this process. And finally, to all the cybersecurity professionals and researchers globally, whose work inspires us to continue learning, adapting, and contributing to a safer digital future—we dedicate this project as a small but sincere contribution to the ongoing fight against cyber threats.

## Acknowledgement

First and foremost, we express our heartfelt gratitude to Allah Almighty for His endless blessings, guidance, and strength, which empowered us to complete this Final Year Project successfully. Our deepest thanks go to our esteemed supervisor, Mr. **Syed Yawar Abbas**, and our co-supervisor, **Dr. Jawaaid Iqbal**, for their continuous support, expert advice, and invaluable feedback throughout this project. Their mentorship was crucial in shaping our research and helping us overcome both technical and conceptual obstacles. We also wish to extend our sincere appreciation to the faculty of the **Department of Cyber Security, Riphah International University**, for providing us with the essential knowledge, resources, and academic environment that were integral to completing this work. A special thank you goes to our friends and classmates for their encouragement, cooperation, and collaborative efforts, which enriched our learning journey. Finally, we owe our deepest gratitude to our families for their unwavering support, patience, and prayers. Their constant belief in us and motivation made this achievement possible.

---

**Anas Mustafa Hashmi**

**37520**

---

**Kabeer Ahmed**

**27970**

---

**Yasir Mahmood**

**36546**

## Abstract

**Advanced Persistent Threats (APTs)** have emerged as one of the most sophisticated threats in the cybersecurity world. This project, *Network Vigilance: AI-Enhanced Multi-Layered Protection Against APTs*, introduces a practical framework that aims to identify and counter APT activities in real-time on a live network. The system combines two types of machine learning approaches: a supervised learning model (Random Forest) and an unsupervised anomaly detection model (Isolation Forest). Each model operates on its own, but only flags a session as malicious when both agree—helping to minimize false alarms and improve detection accuracy. To support these models, we created a detailed dataset by simulating real-world cyberattacks like brute-force attempts, port scanning, DoS attacks, and more within a controlled virtual environment. Traffic data was captured using dumpcap, converted to flow records using CICFlowMeter, and preprocessed for training. When malicious activity is detected, a unique session ID is generated for tracking. The system logs each event, blocks IPs automatically using firewall rules, and stores all data securely in a PostgreSQL backend. An admin dashboard offers real-time monitoring and the option to download detailed PDF reports. With detection accuracy above 94% and fast response times (under 1.2 seconds), the system shows strong potential for academic and enterprise use.

# Table of Contents

List of Figures .....	x
List of Tables .....	xii
Chapter 1: Introduction .....	2
1.1 Introduction .....	2
1.2 Opportunity & Stakeholders .....	2
1.3 Motivations and Challenges .....	3
1.4 Significance of Study .....	3
1.5 Goals and Objectives .....	3
1.6 Scope of Project .....	4
1.7 Chapter Summary .....	5
Chapter 2: Market Survey .....	7
2.1 Introduction .....	7
2.2 Market Survey / Technologies & Products Overview .....	7
2.3 Comparative Analysis .....	9
2.3.1 Comparative Table .....	10
2.3.2 Notes: .....	11
2.4 Problem Statement .....	12
2.5 Chapter Summary .....	12
Chapter 3: Requirements and System Design .....	14
3.1 Introduction .....	14
3.2 System Architecture .....	14
3.3 Functional Requirements .....	16
3.3.1 List of Functional Requirements .....	16
3.3.2 Functional Workflow Summary .....	17
3.4 Non-Functional Requirements .....	17
3.4.1 Non-Functional Architecture Alignment .....	18
3.5 Design Diagrams .....	19
3.5.1 Use Case Diagram .....	20
3.5.2 Data Flow Diagram .....	21
3.6 Hardware and Software Requirements .....	22
3.6.1 Hardware Requirements .....	22
3.6.2 Software Requirements .....	22



Justification: .....	23
3.7 Threat Scenarios .....	23
3.8 Threat Modeling Techniques .....	24
Outcome: .....	25
3.9 Threat Resistance Model .....	25
3.10 Chapter Summary .....	26
CHAPTER 4: PROPOSED SOLUTION .....	29
4.1 Introduction .....	29
4.2 Proposed Model .....	29
4.3 Data Collection .....	30
4.3.1 Virtual Lab Configuration .....	30
4.3.2 Simulated Attack Scenarios .....	30
4.3.3 Traffic Capture and Feature Extraction .....	31
4.3.4 Justification for This Approach .....	31
4.4 Data Pre-Processing .....	32
4.5 Tools and Techniques .....	33
4.5.1 Machine Learning Frameworks .....	33
4.5.2 Network Analysis Tools .....	33
4.5.3 Data Storage and Logging .....	33
4.5.4 Security Features .....	34
4.5.5 Infrastructure and System Design .....	35
4.6 Evaluation Metrics .....	39
4.6.1 Supervised Model (Random Forest) .....	39
4.6.2 Unsupervised Learning Model (Isolation Forest) .....	42
4.6.3 System Performance Metrics .....	43
4.7 Chapter Summary .....	45
CHAPTER 5: IMPLEMENTATION AND TESTING .....	47
5.1 Security Properties Testing .....	47
5.2 System Setup (Environment Configuration, Key Functions, and Algorithms) .....	48
5.2.1 Virtual Environment Setup .....	48
5.2.2 Key Detection Pipeline Components .....	48
5.3 System Integration .....	49
5.3.1 IP Management Workflow .....	51
5.4 Results and Discussion .....	56

5.5 Best Practices / Coding Standards .....	56
5.5.1 Development Practices & Standards .....	57
5.6 Chapter Summary .....	57
CHAPTER 6: CONCLUSION AND FUTURE WORK.....	59
6.1 Conclusion .....	59
6.2 Achievements and Improvements .....	59
6.3 Critical Review .....	60
6.4 Future Work and Recommendations .....	60
6.5 Final Remarks .....	61
Reference: .....	62
Appendix .....	66
Appendix A – System Configuration Details .....	66
1. Virtualized Network Setup: .....	66
2. Traffic Capture: .....	66
3. Flow Feature Extraction: .....	66
Appendix B – Screenshots (Optional in Printed Report) .....	67
GitHub Repository and Version Control .....	78
Advantages of Using GitHub in the Development Process .....	78
GitHub’s Role in the Project .....	78

## List of Figures

Figure 3.1: System Architecture Diagram .....	15
Figure 3.2: Entity Relationship Diagram (ERD) .....	20
Figure 3. 3: Data flow diagram .....	21
Figure 4. 1 PCA-Test Set .....	41
Figure 4. 2 ROC Curve (Random Forest Model) .....	42
Figure 4. 3 Unsupervised Anomaly Detection .....	43
Figure 4. 4 PCA-Anomaly Detection .....	44
Figure 4. 5 distribution of Anomaly Scores .....	45
Figure 4. 6: Anomaly Score Distribution by Class .....	45
Figure 5.1: Virtual Lab Environment .....	49
Figure 5.2: Blocking/Unblocking Logic .....	51
Figure 5.3: Unblocking an IP from Dashboard .....	54
Figure 5.4: Viewing the Blocked IP in Firewall .....	54
Figure 5.5: Confirming the Unblock Action .....	55
Figure 5.6: Firewall Rule Removed (Command Line Confirmation) .....	55
Figure 5.7: Updated Firewall Rules After Unblocking .....	56
Figure 1: New User Registration .....	67
Figure 2: Secure User Authentication .....	67
Figure 3: Password Reset Form .....	68
Figure 4: Login Confirmation Screen .....	68
Figure 5: Dashboard with Real-time Insights .....	69
Figure 6: Searchable Blocked IP Database .....	69
Figure 7: Paginated List of Blocked Ips .....	70
Figure 8: Analytical Overview of Predictions .....	71
Figure 9: Log Data for Model Predictions .....	71
Figure 10: Visual Timeline of Malicious Detections .....	72
Figure 11: Detailed Malicious Flows Table .....	72
Figure 12: Exporting Threat Intelligence Report .....	73
Figure 13: Automated Threat Report Summary .....	74
Figure 14: Session History View .....	75
Figure 15: Session-wise Threat Analysis .....	75

Figure 16:Password Hashing Implementation .....	76
---	----

# List of Tables

Table 2. 1: APT Detection Tools .....	7
Table 2. 2: Comparative Analysis .....	9
Table 2. 3: Comparative Table between other tools and ours .....	10
Table 3. 1: Functional Requirements .....	16
Table 3.2: Non-Functional Requirements .....	18
Table 3. 3: Hardware Specifications .....	22
Table 3. 4: Software Requirements .....	23
Table 3. 5: Attack Scenarios .....	24
Table 3. 6: Threat Mitigations .....	25
Table 4. 1: System Tools Mapping .....	36
Table 4. 2: Technologies & Libraries Used .....	36
Table 4. 3: Component-Action Mapping .....	38
Table 4. 4: Technologies Used of Component .....	39
Table 5. 1: Security Verification .....	47
Table 5. 2: Integration Workflow .....	50
Table 1: Abbreviations Table .....	77

# **Chapter 1:**

# **Introduction**

# Chapter 1: Introduction

## 1.1 Introduction

Today's digital systems are constantly exposed to a wide range of cyber threats, but few are as complex and dangerous as Advanced Persistent Threats (APTs). Unlike ordinary attacks that tend to be quick and opportunistic, APTs are stealthy, long-lasting, and highly targeted. Their aim is often to steal sensitive data or quietly compromise important systems over time—all while staying hidden from traditional security tools like firewalls or antivirus software. As a result, many organizations are turning toward AI-powered security approaches to keep up. This project, titled *Network Vigilance: AI-Powered Multi-Layered APT Protection*, introduces a solution that uses both supervised and unsupervised machine learning techniques to detect threats in real-time. The goal is to improve detection reliability, reduce false positives, and automate the system's response—offering a smarter and more adaptive defense strategy.

## 1.2 Opportunity & Stakeholders

With more organizations relying on digital systems, cybersecurity has become essential across industries. APTs are particularly dangerous because they are carefully planned by experienced attackers and can go unnoticed for long periods. These attacks can impact national security, banking systems, hospitals, and private companies alike. There's a growing need for intelligent tools that can quickly detect and react to such threats as they unfold. Most traditional systems can't distinguish between normal activity and cleverly disguised attacks. Our solution—Network Vigilance—addresses this by using a combination of supervised (Random Forest) and unsupervised (Isolation Forest) models. It's designed to adapt to new and evolving threats. Stakeholders who would benefit from this system include network administrators, IT security teams, researchers, government defense organizations, and cybersecurity analysts.

### **1.3 Motivations and Challenges**

The idea for this project came from seeing how often APTs successfully bypass conventional security systems. Signature-based and rule-based systems don't perform well against unknown threats. We wanted to create a smarter, more flexible system. In doing so, we faced several challenges:

- Designing a pipeline that could monitor traffic in real-time (every 30 seconds).
- Building our own dataset that reflects real APT behaviors.
- Running two machine learning models side-by-side while maintaining speed and accuracy.
- Creating a secure admin panel with IP block/unblock features.
- Ensuring safe data handling with PostgreSQL session tracking.

### **1.4 Significance of Study**

This project stands out in a few key ways. First, it provides a real-world application of AI in cybersecurity, particularly for detecting APTs. Second, it offers a double-check detection method where both models must agree before a threat is confirmed—this reduces false alarms. Lastly, the system is built to work in a live environment. It includes admin login, automatic blocking, logging, and even downloadable reports for review. These features make the system practical for businesses, researchers, and anyone needing robust network protection. Moreover, the project addresses a critical cybersecurity need—detecting threats that bypass conventional firewalls and intrusion detection systems.

### **1.5 Goals and Objectives**

Some goals and objectives are given below:

- The primary goal of Network Vigilance is to provide a robust, real-time APT detection system that operates independently of prior attack signatures. The objectives include:



- Designing and training supervised (Random Forest) and unsupervised (Isolation Forest) models on APT-representative traffic data.
- Building a session-based network capture (session is created for malicious detected event's only) and detection mechanism with 30-second polling.
- Deploying a consensus-based dual-model engine that blocks IPs only when both supervised and unsupervised models independently confirm malicious activity.
- Ensuring secure administrative login functionality.
- Logging all threat-related events to a PostgreSQL database with unique session identifiers.
- Enabling real-time IP blocking and unblocking through automated firewall rule management using Windows Defender Firewall commands.
- Generating downloadable, structured threat reports after each detection event.

## 1.6 Scope of Project

This project is developed within the constraints of a controlled lab environment and focuses on the network-level detection of APTs using packet data captured from local sessions. The detection is based solely on features extracted from captured traffic data and does not include host-based intrusion detection mechanisms. The project scope includes:

- Traffic capture, feature extraction, and real-time model inference.
- Use of Random Forest and Isolation Forest models independently for detection.
- Blocking of traffic only when both models independently confirm a threat with a confidence level above a defined threshold.
- Logging detection events and corresponding metadata in a PostgreSQL database.
- Providing a secure admin dashboard for manual oversight and IP management.

Future iterations may extend this scope to support cross-network analysis, distributed deployment, and more complex adversarial modelling.

## **1.7 Chapter Summary**

This chapter introduced the problem of APTs and established the context for Network Vigilance. It discussed the motivation for developing an intelligent APT detection framework, identified key stakeholders, and outlined the project's goals, scope, and importance. By leveraging machine learning, this project aims to fill a crucial gap in cybersecurity—real-time detection and mitigation of stealthy, evolving network threats. The subsequent chapters will delve into the technical details, literature background, system architecture, implementation, and testing of the proposed solution.

# **Chapter 2:**

# **Market Survey**

# Chapter 2: Market Survey

## 2.1 Introduction

Cyberattacks today are becoming more deliberate, harder to detect, and longer-lasting—especially in the form of Advanced Persistent Threats (APTs). These attacks are designed to quietly slip past traditional defenses like firewalls and signature-based intrusion detection systems (IDS). APTs often follow a structured path: they enter, stay hidden, and slowly extract or manipulate sensitive information. Because of this, basic defenses are no longer enough. The cybersecurity field has responded by developing advanced tools that rely on behavioral analysis, artificial intelligence, threat intelligence feeds, and automation. In this chapter, we explore both commercial and open-source solutions designed for network intrusion detection and APT prevention. Tools like Snort, Suricata, Zeek, FireEye, CrowdStrike Falcon, and Darktrace are analyzed in terms of their strengths, limitations, and technological approaches. We also highlight how our proposed system, Network Vigilance, offers new capabilities through machine learning, real-time decision making, and automated IP management. By critically examining existing tools and mapping their limitations to the design choices of our system, this chapter lays the groundwork for demonstrating the practical value and innovation introduced by our approach.

## 2.2 Market Survey / Technologies & Products Overview

This section presents an overview of popular APT detection tools, traffic monitoring solutions, and real-time intrusion detection products available in the market:

**Table 2. 1: APT Detection Tools**

Tool/Product	Type	Technology Used	Pros	Limitations
Snort	Open-source	Rule/signature-	Lightweight,	Struggles with zero-

	IDS	based	highly customizable, widely adopted	day attacks, rule maintenance needed
<b>Suricata</b>	IDS/IPS	Multi-threaded, signature + basic anomaly detection	High performance, multi-protocol support	Higher false positive rates, signature-dependent
<b>Zeek (Bro)</b>	Network Security Monitor	Script-based traffic analysis	Deep protocol inspection, highly customizable logs	No real-time blocking by default
<b>CrowdStrike Falcon</b>	Commercial EDR	Behavioral AI/Cloud-based	Excellent endpoint telemetry, rapid response	High licensing cost, cloud-reliant
<b>FireEye (Trellix)</b>	APT Detection & Response	Heuristics + threat intel + sandboxing	Effective against advanced persistent threats	Proprietary, expensive
<b>Cisco Secure Network Analytics (Stealthwatch)</b>	NTA (Network Traffic Analysis)	Machine learning + NetFlow	Flow-based anomaly detection	Setup complexity, costly
<b>Darktrace</b>	AI Threat Detection	Self-learning AI	Autonomous detection	Black-box approach, costly
<b>Splunk Enterprise Security</b>	SIEM	Big Data analytics + ML	Custom queries, powerful dashboards	Licensing cost, steep learning curve

Table 2.1 Overview of Popular APT Detection and Intrusion Monitoring Tools, Highlighting Their Technologies, Strengths, and Limitations.

## 2.3 Comparative Analysis

The comparative analysis is given below:

**Table 2. 2: Comparative Analysis**

<b>Feature / Technique</b>	<b>Snort / Suricata</b>	<b>FireEye / CrowdStrike</b>	<b>Deep Learning Models</b>	<b>Network Vigilance (Ours)</b>
Signature-based Detection	Yes	Partially	No	Yes
Behavior-based Detection	Limited	Yes	Yes	Yes
Real-time Blocking	Limited	Yes	No (post- analysis)	Yes (via dual model agreement)
Model Types	None	Proprietary	RNN, CNN	Random Forest + Isolation Forest
Open source	Yes	No	Varies	Yes
Manual IP Control	No	Limited	No	Yes
Dataset Used	N/A	Private	Public Datasets	Custom (lab- generated)
Cost	Free	High	Moderate	Free (for development)

Table 2.2 shows this comparison, it's evident that most commercial or open-source tools lack the combined benefits of real-time supervised and unsupervised detection, session-wise tracking, and full admin interactivity.

### 2.3.1 Comparative Table

To better contextualize the innovation and strengths of Network Vigilance, we compare our solution against widely adopted commercial tools, academic frameworks, and open-source intrusion detection systems.

**Table 2. 3: Comparative Table between other tools and ours**

<b>Feature / Tool / Approach</b>	<b>Snort / Suricata (Open Source)</b>	<b>FireEye / CrowdStrike (Commercial)</b>	<b>Deep Learning IDS (Academic)</b>	<b>Network Vigilance (Proposed)</b>
Detection Type	Signature-based	Behavioral + Signature-based	Behavioral / Statistical	Anomaly + Behavioral (Model-Based)
Real-Time Traffic Analysis	Yes	Yes	Often No	Yes
Supervised ML	No	Partially (Proprietary)	Yes	Yes (Random Forest)
Unsupervised ML	No	Rare	Yes	Yes (Isolation Forest)
Hybrid Detection Model	No	Yes	Yes	No (Decision-Level Fusion)
IP Blocking	Manual or External Firewall	Automated	Limited or External Scripts	Automated (on Dual Model Agreement)

Secure Admin Login	No	Yes	No	Yes
Logging & Session Tracking	Basic	Yes	No	Yes (via PostgreSQL)
Cost	Free	Very Expensive (Enterprise)	Varies	Free / Open-Source
Modifiability / Customization	High	Low	High	High
Dataset Used	N/A	Proprietary	Public	Custom
Usability for Research/Development	High	Low	High	High
Deployment Environment	On-premises	Cloud / Hybrid	Research Labs	On-premises
Support & Updates	Community	Dedicated	None	Community / Custom

Table 2.3 Compares IDS solutions, showing that Network Vigilance stands out by combining supervised and unsupervised ML for real-time, automated detection and IP blocking, with secure admin access and full session logging—offering high customizability and suitability for both research and practical use.

### 2.3.2 Notes:

- Snort and Suricata are powerful but rely exclusively on predefined rule sets and lack adaptive learning capabilities.
- FireEye and CrowdStrike offer advanced threat intelligence but are expensive, proprietary, and generally not open to academic extension or customization.



- Academic models (e.g., using LSTM, CNN, or GANs for intrusion detection) demonstrate impressive detection capabilities but are not often deployable in real-time environments or lack automation for mitigation.
- Network Vigilance provides a rare combination: open-source flexibility, real-time traffic analysis, dual-model detection (Random Forest and Isolation Forest), automatic IP blocking (only when both models concur), secure admin login, and structured threat logging—all in a single modular system.

## 2.4 Problem Statement

Despite the wide availability of intrusion detection tools, there is a clear gap in systems capable of catching stealthy APTs as they occur. Most existing tools either depend heavily on known attack signatures or use a single machine learning model, which increases the chance of errors. Furthermore, many don't offer built-in admin dashboards, real-time IP control, or detailed session-level tracking. This lack of flexibility and adaptability highlights the need for a more intelligent, responsive, and affordable system tailored specifically to the behavior of APTs in a virtualized environment.

## 2.5 Chapter Summary

This chapter reviewed the current landscape of network intrusion detection and APT prevention tools. It highlighted their capabilities, gaps, and limitations. While some excel in behavior analysis or dashboard design, they often fall short in areas like automation, customization, or affordability. *Network Vigilance* fills these gaps by offering a real-time, dual-model threat detection system with built-in automation and administrative tools—providing a fresh, accessible solution for academic, research, and enterprise use cases.

# **Chapter 3:**

## **Requirements and System Design**

# Chapter 3: Requirements and System Design

## 3.1 Introduction

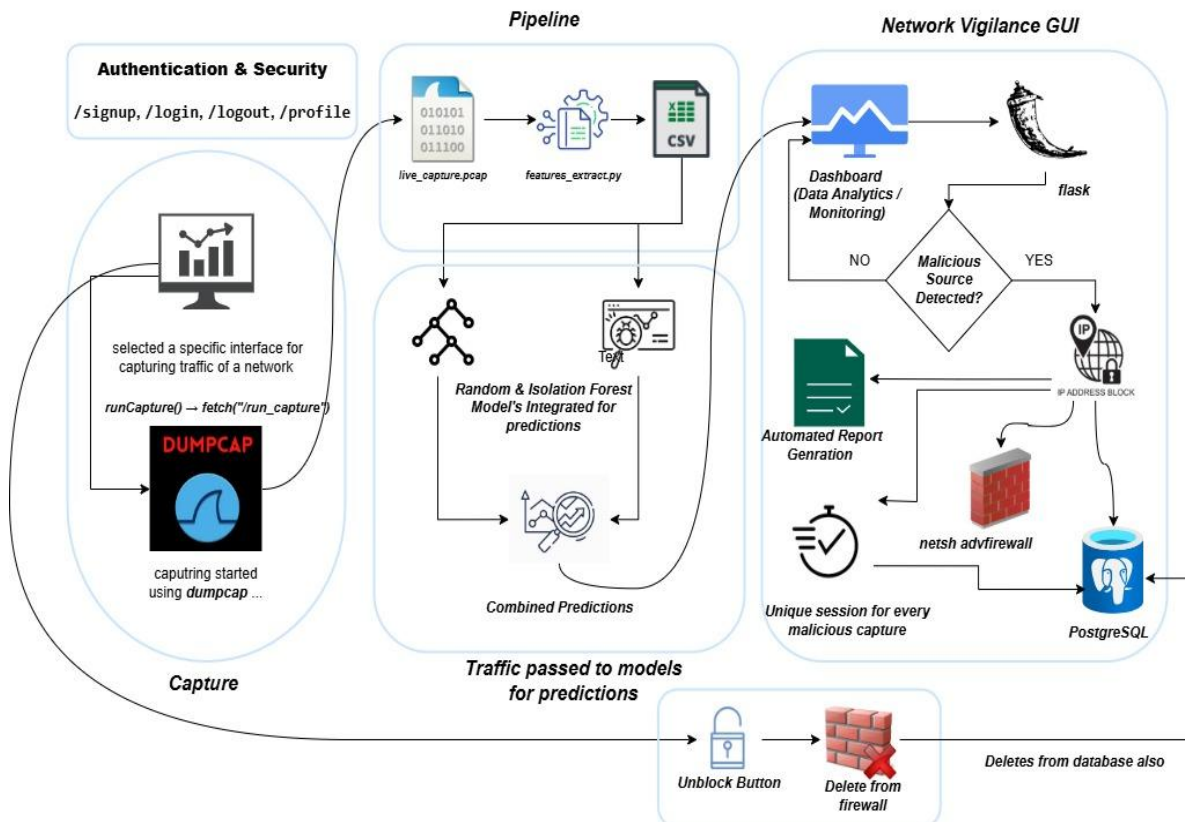
This chapter outlines the technical backbone of *Network Vigilance*, a system designed to detect and respond to Advanced Persistent Threats (APTs) in real time. It introduces the system's architecture, core design decisions, required tools and technologies, and security considerations. The system uses two independent machine learning models: Random Forest for supervised classification and Isolation Forest for anomaly detection. Each model operates separately, and action is taken only when both identify a flow as malicious. This consensus method reduces false positives while maintaining a high detection rate. Additional components such as packet capturing, feature extraction, real-time logging, and automated IP blocking are also integrated. To ensure scalability and long-term usefulness, the system is built with a modular structure and can adapt to evolving threats.

## 3.2 System Architecture

The architecture of Network Vigilance is layered and modular to support real-time monitoring without overloading system resources. The key components are:

- **Traffic Capture:** Uses dumpcap to collect live packet data every 30 seconds with minimal system impact.
- **Feature Extraction:** A Python script processes captured packets into structured flow records suitable for machine learning analysis.
- **Detection Engine:** Integrates two independent machine learning models to enhance detection reliability:
  - **Random Forest:** A supervised learning model trained on labelled data to detect known APT signatures and attack patterns.
  - **Isolation Forest:** An unsupervised anomaly detection model that identifies unusual traffic behaviors do not present in the training set.

- **Database Management:** PostgreSQL serves as the system's core data store, maintaining:
  - Detection logs
  - Blocked and unblocked IP addresses
  - Session history for traceability
  - Auto-generated threat reports in PDF format
- **Blocking and Unblocking Module:** Uses Windows Firewall rules (via netsh) to execute automated IP blocking and allow manual unblocking through the admin dashboard, ensuring secure and controlled mitigation.
- **Threat Reporting System:** For each malicious detection, a unique session is recorded. Associated metadata is logged, and a downloadable PDF report is generated for audit and analysis purposes



**Figure 3.1: System Architecture Diagram**

Figure 3.1, This diagram illustrates the complete operational flow of the Network Vigilance system, from traffic capture to real-time prediction and administrative response. Network packets are captured using dumpcap, initiated from a secure login interface. Extracted flow features are processed and converted into CSV format before being analyzed by both supervised (Random Forest) and unsupervised (Isolation Forest) models. Only when both models agree on a session being malicious, an automated firewall rule is triggered using netsh to block the offending IP. Each malicious session is logged in PostgreSQL and visualized on the dashboard. Admins can later unblock IPs via the GUI, which deletes the firewall rule and corresponding database entry. The system ensures secure access, accurate detection, and complete traceability through unique session logging.

### 3.3 Functional Requirements

Functional requirements define the **core operations** and **expected behaviors** of the system. They specify **what the system should do**, describing the tasks, inputs, outputs, and interactions necessary to fulfill the intended objectives. In the context of “*Network Vigilance: AI-Powered Multi-Layered APT Protection*,” the system is expected to detect, classify, visualize, and mitigate APT threats using AI models, real-time monitoring, and automated response mechanisms. Each function is designed to operate **independently yet cohesively** within the larger system architecture, ensuring **modularity, flexibility, and scalability**.

#### 3.3.1 List of Functional Requirements

The List of Functional Requirements is given below:

**Table 3. 1: Functional Requirements**

<b>Requirement ID</b>	<b>Description</b>
FR1	Capture network traffic in real-time using dumpcap.
FR2	Custom python script used for feature extraction.
FR3	Detect network threats using Random Forest and Isolation Forest

models.	
FR4	Generate session-based PDF threat reports.
FR5	Store session data, detections, and IP logs in PostgreSQL.
FR6	Allow manual unblocking of IPs via the user interface.
FR7	Automatically block IPs with a defined threshold

Table 3.1 outlines the core functional requirements of the system, including real-time traffic capture (FR1), flow conversion (FR2), dual-model threat detection (FR3), automated report generation (FR4), PostgreSQL-based data storage (FR5), manual IP management through the admin panel (FR6), and automated IP blocking when the defines threshold.

### 3.3.2 Functional Workflow Summary

- Real-time traffic monitoring is initiated by capturing packets from a selected network interface.
- Extracted packets undergo preprocessing to generate structured flow features suitable for machine learning analysis.
- Both supervised (Random Forest) and unsupervised (Isolation Forest) models independently evaluate the flow data.
- The decision engine flags a session as malicious only when both models concur, ensuring high-confidence detection and reducing false positives.
- Confirmed threats are displayed on the dashboard and forwarded to the mitigation system for automated IP blocking.
- All detection activities, logs, and reports are stored securely for administrative review and audit purposes.

### 3.4 Non-Functional Requirements

Unlike functional requirements that focus on what the system does, non-functional requirements (NFRs) deal with how efficiently and effectively the system performs its tasks. They reflect key quality traits such as performance speed, ease of use,

dependability, maintainability, and system security. In the context of *Network Vigilance: AI-Powered Multi-Layered APT Protection*, these requirements are vital for ensuring the system remains stable and responsive under real-time conditions. Since APTs are highly dynamic and often sophisticated, the system must be designed with enough flexibility and robustness to handle diverse and evolving scenarios. Setting clear non-functional standards helps guide the project through its development and deployment phases. Ensuring these benchmarks are met means the system will not only function correctly but also deliver a secure, user-friendly, and scalable experience—qualities that are essential for cybersecurity solutions that must operate under pressure with limited manual intervention.

### 3.4.1 Non-Functional Architecture Alignment

The Non-Functional Architecture Alignment table is given below:

**Table 3.2: Non-Functional Requirements**

Category	Description
Performance	Detection and blocking must be completed within seconds after flow analysis.
Scalability	Capable of scaling through additional sensors and modular deployments.
Maintainability	Modular design for easy upgrades and troubleshooting.
Accuracy	Aims for high true positive rates and low false positives.
Usability	User-friendly interface for viewing logs, reports, and managing IPs.
Security	Secure PostgreSQL access and enforced admin authentication for all operations.

Table 3.2 outlines the key non-functional requirements of the system, emphasizing quick threat identification and response (performance), flexibility to scale through a modular structure (scalability), ease of updating and troubleshooting (maintainability), reliable threat detection with minimal false positives (accuracy), a user-friendly interface for

efficient interaction (usability), and strong protection mechanisms including secure logins and restricted database access (security).

### 3.5 Design Diagrams

Creating visual representations is an essential part of building and understanding complex systems, as it helps clarify structure, flow, and interactions among components.. In this section, several standardized system design diagrams are presented to describe the architecture, component interactions, and process flows of the proposed APT detection system based on dual-model consensus logic. These diagrams help in understanding how the system components collaborate, how data moves through the system, and how user actions translate into system behavior. The use of UML (Unified Modeling Language) and data flow diagrams (DFD) ensures that the design is both technically precise and easily interpretable by both developers and academic reviewers. The following diagrams are included (to be created post-report):

- **Use Case Diagram:** Demonstrates interactions between the admin and system modules.
- **Data Flow Diagram:** Details how data progresses from capture to database storage.



3.5.1 Use Case Diagram

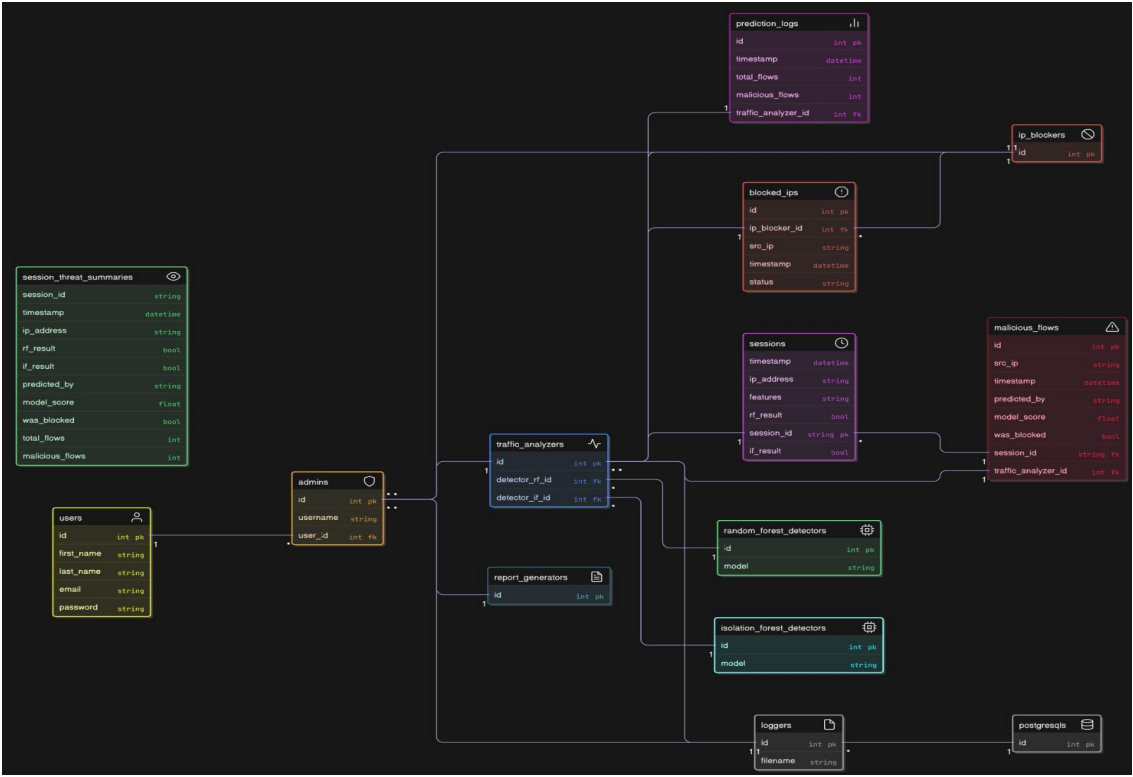
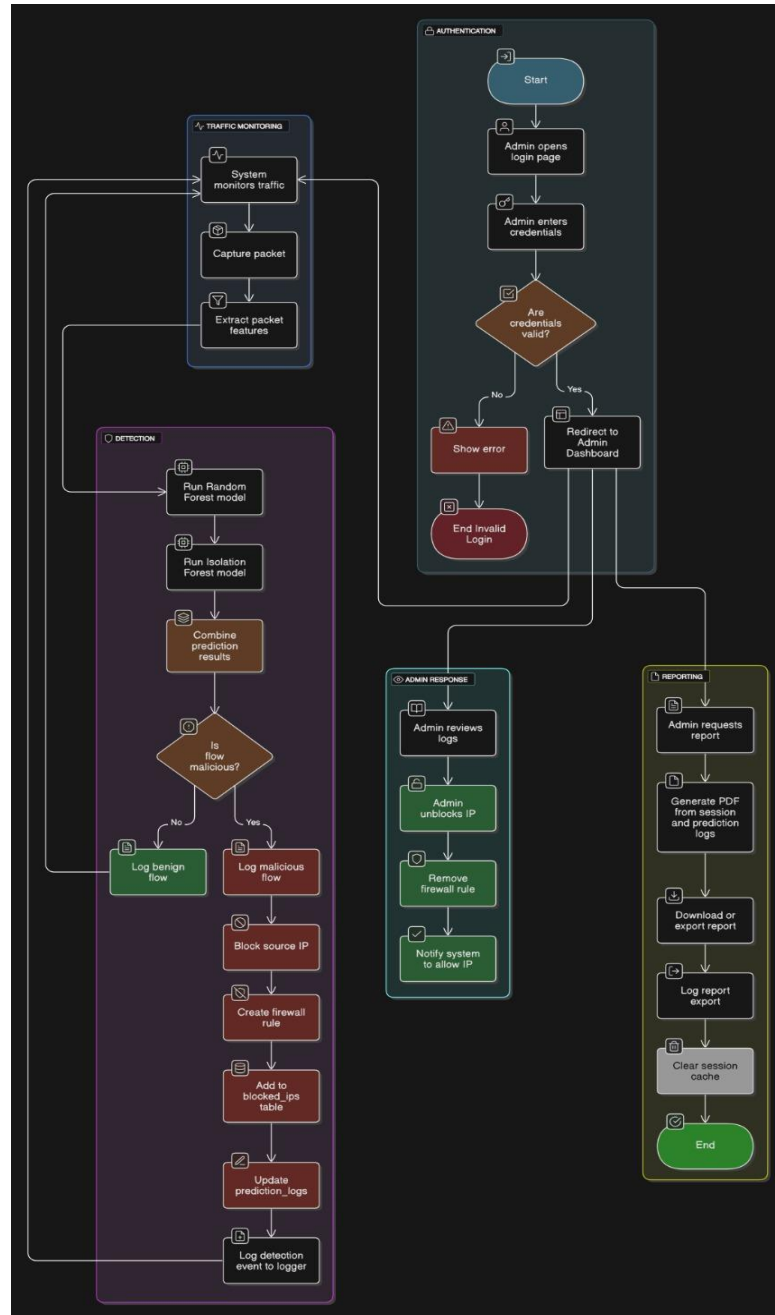


Figure 3.2: Entity Relationship Diagram (ERD)

Table 3.2: ERD highlights the core schema structure of the backend system, showing how key entities such as users, sessions, detectors, and IP blockers are related. It enables traceable, real-time flow tracking, detection, and report generation across the PostgreSQL database.

### 3.5.2 Data Flow Diagram



**Figure 3. 3: Data flow diagram**

Figure 3.3: The diagram illustrates the complete operational flow of the Network Vigilance system, beginning with traffic monitoring where packets are captured and features extracted. These features are analyzed using both Random Forest and Isolation Forest models. A decision logic flags a session as malicious only if both models agree,

triggering automated IP blocking and firewall rule creation. Admins authenticate securely to access the dashboard, where they can review logs, unblock IPs, and remove associated rules. The system also supports report generation based on logged data, ensuring visibility, traceability, and responsive threat mitigation in real time.

### 3.6 Hardware and Software Requirements

The successful implementation of the proposed system — “*Network Vigilance: AI-Powered Multi-Layered APT Protection*” — requires a suitable combination of hardware and software resources. These resources must support real-time packet analysis, machine learning model inference, API-based communication, and dynamic frontend rendering without introducing significant latency or system bottlenecks. This section outlines both the **minimum and recommended hardware specifications**, along with the **software stack** necessary to develop, deploy, and operate the entire system effectively within a **local network environment or simulated virtual lab**.

#### 3.6.1 Hardware Requirements

The hardware requirements are given below:

**Table 3. 3: Hardware Specifications**

Hardware	Specification
Processor	Intel i5/i7 or AMD equivalent
RAM	Minimum 8GB
Storage	At least 100GB HDD/SSD
Network Adapter	Ethernet adapter (VM bridged mode recommended)

Table 3.3 shows the hardware setup requires a minimum an Intel i5/i7 or AMD equivalent processor, 8GB of RAM, 100GB HDD/SSD storage, and an Ethernet adapter configured in bridged mode for effective VM networking.

#### 3.6.2 Software Requirements

The Software Requirements is given below:

**Table 3. 4: Software Requirements**

Software	Version/Details
Operating System	Windows 10/11
PostgreSQL	v13+
CICFlowMeter	v4.0
Wireshark + Dumpcap	Latest stable release
Python	3.8+ with scikit-learn
Web Interface	Flask / Streamlit
ML Libraries	scikit-learn, pandas, matplotlib

Table 3.4 lists the essential software components: Windows 10/11 as the OS, PostgreSQL v13+, CICFlowMeter v4.0, the latest Wireshark/dumpcap, Python 3.8+ with key libraries (scikit-learn, pandas, matplotlib), and Flask/Streamlit for the web interface, supporting full system functionality.

### **Justification:**

While this system is primarily CPU-bound, a multi-core processor and ample RAM are essential to process and classify large volumes of packets without lag. SSD storage improves data read/write speed, especially when handling logs and report generation.

## **3.7 Threat Scenarios**

To ensure the robustness, practicality, and real-time effectiveness of the proposed system, it is critical to simulate and evaluate it under **realistic and diverse cyber threat conditions**. This section outlines a range of **threat scenarios** that emulate techniques commonly used in **Advanced Persistent Threats (APTs)**. Each represents a potential **attack vector** or **stage of the cyber kill chain**, allowing comprehensive testing of both the detection and response mechanisms of the system. The threat scenarios is given below.

**Table 3. 5: Attack Scenarios**

Scenario	Description
Port Scanning	Detected via unusual port activity using tools like Nmap.
SSH Brute Force	Identified by repeated failed logins, typically using Hydra.
DoS Attacks	Recognized by traffic floods, e.g., ICMP or SYN flood attacks.
MITM (ARP Spoof)	Traffic alteration and interception via Ettercap-like tools.
RDP Exploitation	Attack attempts on RDP ports using brute force or known exploits.

Table 3.5 outlines five distinct real-world cyber threat scenarios, each specifically designed to evaluate the effectiveness of the proposed APT protection system. These scenarios span various stages of a cyberattack, offering valuable insights into the system's ability to identify, categorize, and mitigate both familiar and emerging threats.

### 3.8 Threat Modeling Techniques

Threat modeling plays a vital role in cybersecurity design by helping to identify, assess, and categorize potential threats to a system. This process is key to developing effective mitigations and enables a proactive approach to security. By understanding where vulnerabilities might exist and how attackers could exploit them, designers and developers can better plan defenses. In the context of the project "Network Vigilance: AI-Powered Multi-Layered APT Protection," threat modeling ensures that the system:

- Resists real-world adversary behavior
- Handles unexpected usage or misuse
- Supports effective mitigation strategies

This section outlines the **frameworks and methodologies** applied to model threats in the context of Advanced Persistent Threats (APTs), with emphasis on **STRIDE**, **MITRE ATT&CK**, and **Cyber Kill Chain Analysis**.

We employed the STRIDE model to analyze threats across different components.

**Table 3. 6: Threat Mitigations**

Threat	Description	Mitigation
Spoofing	IP/User impersonation	IP verification, session logging in PostgreSQL
Repudiation	Denying malicious activity	Secure, immutable audit logs
Information Disclosure	Unauthorized data access	Secure database access only admin permissions
Denial of Service	System overloaded through traffic spikes	Flow anomaly detection, alert generation
Elevation of Privilege	Unauthorized privilege escalation	Role-based UI authentication and strict access control

Table 3.6 outlines key threats and defenses: spoofing (IP verification, session logs), tampering (read-only logs, hashed reports), repudiation (secure audit logs), information disclosure (role-based secure access), denial of service (flow anomaly alerts), and privilege escalation (role-based authentication and strict controls).

### Outcome:

By aligning system defenses with STRIDE categories, the solution is capable of handling a wide spectrum of common cyber threats systematically.

## 3.9 Threat Resistance Model

A **Threat Resistance Model** describes how a system actively defends itself against a range of cybersecurity threats, particularly under live attack conditions. Unlike general

threat modeling (which identifies *potential* threats), the resistance model focuses on **how the system’s components neutralize, respond to, or mitigate those threats in real time**. In the case of “*Network Vigilance: AI-Powered Multi-Layered APT Protection*,” The resistance model combines multiple strategies. The system’s resilience is strengthened through:

- **Dual-Model Detection:** The Random Forest model identifies known patterns, while the Isolation Forest captures behavioral anomalies—ensuring detection of both known and unknown threats.
- **30-Second Detection Interval:** The system continuously monitors in real-time by processing network flows in brief, repetitive intervals, allowing for quick responses and mitigation.
- **Session-Based Logging:** Each detection event is linked to a specific session ID, improving both tracking and audit capabilities.
- **Automated and Manual Mitigation:** The system automatically initiates IP blocking once consensus is reached by the detection model, but administrators can manually intervene and review decisions through a secure interface.
- **Enforcement via Database:** Blocking actions are meticulously logged and carried out through PostgreSQL, ensuring that policy enforcement is secure and tamper-proof, with clear forensic trails.

### 3.10 Chapter Summary

This chapter provides a comprehensive overview of the system architecture for “*Network Vigilance: AI-Powered Multi-Layered APT Protection*.” It presents a modular framework that combines both supervised and unsupervised machine learning models, utilizing independent inference along with consensus-based decision-making for effective threat detection. The system's functional and non-functional requirements are discussed to ensure reliable performance in real-time environments. Design diagrams help visualize user interactions, data flows, and overall system operations, while the specified hardware and software requirements confirm that the system can be deployed efficiently with minimal resources. Realistic threat scenarios were defined to validate the system, and

leading frameworks like **STRIDE**, **MITRE ATT&CK**, and the **Cyber Kill Chain** were used for thorough threat modeling. Finally, a robust **Threat Resistance Model** demonstrated how the system detects, responds to, and learns from both known and unknown threats autonomously. This design foundation prepares the way for the system's implementation, covered in the next chapter.



# **Chapter 4:**

# **PROPOSED SOLUTION**

# CHAPTER 4: PROPOSED SOLUTION

## 4.1 Introduction

This chapter presents the proposed solution for detecting and mitigating Advanced Persistent Threats (APTs) through a real-time, session-aware, dual-model detection framework. Our approach uses parallel yet independent operation of a supervised learning model (Random Forest) and an unsupervised anomaly detection model (Isolation Forest). The system emphasizes early detection, autonomous response, and secure oversight via a protected administrative interface. Rather than merging models into a hybrid, we apply decision-level fusion, where blocking is enforced only if both models independently flag a session as malicious.

## 4.2 Proposed Model

The proposed solution adopts a quantitative, experimental methodology that integrates multiple detection and control components. The approach is structured around the following core phases:

- **Supervised Classification (Random Forest):** Utilizes a labeled dataset to classify traffic based on known APT patterns, providing high accuracy in identifying previously seen threats.
- **Unsupervised Anomaly Detection (Isolation Forest):** Independently monitors live traffic to identify statistical outliers that may represent novel or stealthy attacks.
- **Consensus-Based Decision Engine:** A session is flagged as malicious—and corresponding IP blocking is executed—only if both models concur, significantly reducing false positives.
- **Session Logging and Traceability:** Each detection event is recorded in PostgreSQL, including session ID, timestamp, and the individual verdicts of both models, ensuring auditability.

- **Administrative Oversight:** A secure, web-based interface provides authenticated access to system logs, threat decisions, and manual controls for unblocking IPs or overriding automated actions.

## 4.3 Data Collection

To ensure accuracy, realism, and control in evaluating the proposed APT detection system, a custom dataset was created from the ground up using a fully virtualized lab environment—entirely independent of public datasets. This allowed for fine-tuned simulation of sophisticated multi-stage APT attack scenarios and precise labeling of traffic.

### 4.3.1 Virtual Lab Configuration

- The dataset was built using VMware Workstation Pro, where an isolated virtual network was established, consisting of:
  - **Victim Machine:** A single Ubuntu-based VM running commonly targeted services (e.g., SSH, RDP, web server).
  - **Attacker Machines:** Three Kali Linux VMs, each responsible for different stages of APT-style intrusions.

### 4.3.2 Simulated Attack Scenarios

To emulate the tactics of Advanced Persistent Threats (APTs), a series of structured and controlled cyberattacks were conducted within a virtual lab environment. These attacks reflect common stages of real-world intrusions and were designed to validate the system's detection capabilities under diverse threat conditions:

- **Reconnaissance:** Network mapping and host discovery were performed using Nmap and Zenmap to simulate initial probing activities.
- **Brute Force Attacks:** SSH and RDP login attempts were launched using tools like Hydra and Medusa to mimic unauthorized credential guessing.
- **Denial of Service (DoS):** Traffic flooding attacks, including ICMP and SYN floods, were initiated using hping3 to overwhelm system resources.

- **Man-in-the-Middle (MITM):** ARP spoofing was conducted with Ettercap to intercept and manipulate communication between networked systems.
- **Exploitation:** Reverse shells and RDP-based access attempts were executed using the Metasploit Framework, including custom payloads, to simulate post-exploitation control.

### 4.3.3 Traffic Capture and Feature Extraction

Network traffic was captured in real time using high-fidelity tools to ensure reliable and structured input for the machine learning pipeline. The following tools were employed:

- **dumpcap** – Used for high-performance, continuous packet capture with minimal overhead.
- **Wireshark** – Employed for manual inspection and validation of captured traffic at the packet level.

Captured packets were converted into structured flow records, each comprising **84 distinct features**, designed to capture both basic and complex behavioral traits. Key categories included:

- Packet and byte counts per source and destination direction
- Flow durations, idle periods, and burst lengths
- Average and maximum packet sizes
- Inter-arrival times across packets
- Protocol-specific flags (e.g., TCP FIN, SYN, ACK) and behavioral markers

These enriched flow vectors were stored in CSV format and served as input to both supervised and unsupervised detection models.

### 4.3.4 Justification for This Approach

This approach was selected to ensure accuracy, control, and practical relevance in model training and testing. The key strengths include:

- **Custom-Built Dataset:** Fully synthetic traffic generated under known conditions allowed for precise and reliable labelling.
- **Realistic APT Simulation:** Attack behaviors were recreated using widely recognized offensive tools and techniques from penetration testing toolkits.
- **Reproducibility:** The lab environment and capture methods ensure the experiments can be repeated or extended for future work.
- **Balanced Dataset:** The training data was curated to include a fair distribution of benign and malicious flows, reducing model bias.

This rigorously constructed dataset underpins the detection engine and provides the foundation for training the Random Forest and Isolation Forest models at the core of the Network Vigilance system

## 4.4 Data Pre-Processing

To ensure model efficiency and accuracy, the following preprocessing steps are employed:

- **Outlier Removal:** Incomplete or corrupt flows were detected and eliminated using custom Python scripts.
- **Filtering:** Removal of duplicate flows, IPv6 traffic, and local-only communications.
- **Labelling:** Manual tagging of flows as benign or attack based on known attack timelines and behaviors.
- **Balancing:** Class imbalance was addressed by undersampling benign traffic to maintain a balanced dataset.
- **Feature Selection:** Low-variance and highly correlated features were dropped to improve model efficiency.
- **Scaling:** StandardScaler was applied to normalize numerical features for better model performance.

## 4.5 Tools and Techniques

### 4.5.1 Machine Learning Frameworks

- **Scikit-learn:** Used to train and deploy both Random Forest and Isolation Forest models for supervised and unsupervised detection respectively.
- **Joblib:** Employed for model serialization and persistence to allow quick model loading during live predictions.

### 4.5.2 Network Analysis Tools

- **Dumpcap:** Handles high-performance, real-time packet capture and network session recording.
- **Wireshark:** Used for offline inspection and manual validation of packet-level traffic.

### 4.5.3 Data Storage and Logging

#### a. Database Storage (PostgreSQL):

All critical operational and detection-related data is stored in a structured relational format across the following tables:

- **Users Table:** Stores administrator credentials (first name, last name, email) with PBKDF2-hashed passwords.
- **Blocked IPs Table:** Logs malicious IP addresses along with timestamps of detection.
- **Prediction Logs Table:** Records model verdicts, flow scores, timestamps, and associated flow/session identifiers.
- **Sessions Table:** Each captured session is uniquely recorded, including total flows and number of detected malicious ones.
- **Unblock Actions Table (if implemented):** Captures admin-initiated unblock actions for audit and rollback.

#### **b. PDF Reports:**

- Generated automatically for flagged sessions or specific IPs.
- Summarize threat details, model verdicts, scores, and associated metadata.
- Serve as tamper-resistant audit records for compliance and external review.

#### **c. Logging System:**

- **Application Logs:** Record all major system events such as predictions, admin actions (e.g., unblock), and PDF generation.
- **Logged Fields Include:**
  - Action performed
  - Affected IP address
  - Responsible user
  - Timestamp
- **Prediction Event Logs:**
  - Source IP
  - Prediction model used (Random Forest, Isolation Forest, or both)
  - Prediction score and threshold
  - Related session ID
- **Unblock Logs:** Timestamps and metadata are recorded when an IP is manually unblocked.

#### **4.5.4 Security Features**

- **User Authentication:**

Secure login system using email and password credentials. Passwords are hashed with PBKDF2 using SHA-256, adhering to cryptographic best practices.
- **Role-Based Access Control (RBAC):**

Only authenticated administrators can access the detection dashboard and perform sensitive actions such as unblocking or report generation.

- **Session Management:**  
Each detection event is tied to a unique session ID, ensuring traceability and separation of captured traffic for post-incident forensics.
- **Threat Response System:**
  - IPs classified as malicious are automatically blocked at the OS level using dynamic Windows Firewall rules.
  - Unblocking actions remove firewall rules and update the database accordingly.
- **Audit Logging:**  
All system actions, including login attempts, IP blocks/unblocks, and report exports, are logged with timestamps to support accountability and compliance.

#### 4.5.5 Infrastructure and System Design

- **Web Application Backend:**  
Developed using Flask (Python) to provide API endpoints, routing, and administrative access.
- **Frontend Interface:**  
Built using HTML, CSS, and Bootstrap to ensure responsive and user-friendly layouts for admin interaction.
- **Database Management:**  
Powered by PostgreSQL, storing all operational data including users, predictions, and session flows. Optimized SQL queries are used for real-time read/write efficiency.
- **Deployment Environment:**  
Designed for deployment on secure local servers or scalable cloud platforms (e.g., using Nginx, Gunicorn, and optionally Docker). Static logs and report files are isolated to support better performance and data integrity.
- **PDF Report Generation:**  
System-generated reports summarize key findings per session and are exported as tamper-evident PDF documents for audit, compliance, and offline analysis.



- **Monitoring Interface:**

Real-time dashboards display traffic metrics, threat detections, and blocked IPs using embedded charts and visual summaries, enabling administrators to understand patterns and respond promptly.

**Table 4. 1: System Tools Mapping**

Tool/Technique Used	Purpose	Version
Flask	Backend web framework	2.2.5
Pyscopg2-binary	PostgreSQL database connection	2.9.7
Scikit-learn	Machine learning (RF & IF model)	1.3.2
Pdftkit + wkhtmltopdf	Pdf report generation	1.0.0 / 0.12.6
matplotlib	Data plotting for dashboard	3.8.0
Uuid, datetime, asyncio	System utilities	Built in
Dumpcap (wireshark)	Network packet capture	External tool
Netsh (Windows)	Firewall rule creation/deletion	Windows built-in
Jinja2 (via flask)	Html template rendering	3.1.2

Table 4.1 outlines the essential tools and technologies integrated into the pipeline system. It includes frameworks for backend development (Flask), database communication (psycopg2), and machine learning (Scikit-learn). Tools like dumpcap and Wireshark handle real-time packet capture, while matplotlib supports visual dashboards. Report generation is managed via pdftkit and wkhtmltopdf, with Jinja2 rendering HTML templates. System utilities (uuid, datetime, asyncio) enhance functionality, and IP blocking is enforced through Windows netsh commands logged via PostgreSQL.

**Table 4. 2: Technologies & Libraries Used**

Component	Library/Tool	Purpose	Suggested Version
<b>Web Framework</b>	Flask	Web server,	Flask==2.2.5
		routing, session management	
<b>Database Driver</b>	psycopg2	PostgreSQL database	psycopg2-binary==2.9.7

		connection	
<b>PDF Generation</b>	pdftkit	Convert HTML to downloadable PDF reports	pdftkit==1.0.0
<b>Rendering</b>	Jinja2 (Flask built-in)	Template engine for HTML pages and PDF rendering	Jinja2==3.1.2
<b>Feature Extraction</b>	Custom Module (extract_custom_features)	Extract features from PCAP files	(Custom code)
<b>Prediction</b>	Custom Module (predict_from_csv)	Predict threats using trained models	(Custom code)
<b>Machine Learning</b>	scikit-learn	ML backend for Random Forest and Isolation Forest	scikit-learn==1.3.2
<b>Plotting (for charts)</b>	matplotlib	Generate graphs for web dashboard (logs, flows)	matplotlib==3.8.0
<b>Networking</b>	dumppcap (Wireshark CLI)	Captures network traffic into PCAP files	(external tool)
<b>Firewall Control</b>	netsh (Windows) via subprocess	Creates/deletes Windows Firewall rules	(Windows built-in)
<b>UUID Handling</b>	uuid	Generate unique session IDs	uuid (standard lib)
<b>Date Handling</b>	datetime	Timestamps, time windows, session	datetime (standard lib)

		expiry	
Async Execution	asyncio	Async flow for feature extraction	asyncio (standard lib)
PDF Tool Dependency	wkhtmltopdf	Executable used	
		by pdftkit to convert HTML to PDF	0.12.6 recommended
Web Charts	Chart.js (via template)	Client-side charts	
		for traffic summaries	Chart.js v3.x

Table 4.2 outlines the core libraries and tools integrated within the system across backend, frontend, machine learning, networking, and reporting components. Each entry lists the component's function, the associated library or tool, its purpose in the architecture, and its recommended version. This table ensures reproducibility and highlights adherence to stable, well-supported technologies.

**Table 4. 3: Component-Action Mapping**

Action	Component	Effect
Prediction	/predict route	Detects malicious IPs using AI models and triggers blocking process
Block	block_ip()	Creates Windows Firewall rule (via netsh) and logs IP in PostgreSQL
View Blocked IPs	/blocked_ips	Displays all blocked IPs from the blocked_ips database table
Unblock (UI)	/unblock/<id>	Removes firewall rule and deletes record from database

Firewall Logic	netsh (Python)	Blocks through Windows Firewall commands
----------------	----------------	--

Table 4.3 maps system actions to the corresponding backend components responsible for their execution. It demonstrates how features such as threat prediction, IP blocking, and unblocking are handled through specific routes and functions. This mapping clarifies system flow logic and highlights the modular structure enabling secure and automated threat response.

**Table 4. 4: Technologies Used of Component**

Component	Technology
Backend	Python, Flask
ML Models	Scikit-learn (Random Forest, Isolation Forest)
Database	PostgreSQL
Frontend	HTML, CSS, JS
Database	PostgreSQL
Reports	PDFKit / ReportLab

Table 4.4 summarizes the primary technologies adopted for each major system component. It lists tools and languages used in backend development, machine learning implementation, database management, frontend interface, activity logging, and report generation. The selection reflects performance, maintainability, and integration ease within the system’s real-time detection framework.

### 4.6 Evaluation Metrics

To assess the performance and reliability of the detection pipeline in Network Vigilance, both model-level and system-level evaluation metrics were utilized. These metrics reflect the system’s ability to accurately detect threats, operate efficiently in real time, and provide actionable feedback for administrative control.

#### 4.6.1 Supervised Model (Random Forest)

The Random Forest classifier was evaluated using the following standard classification metrics:

- **Accuracy:** Measures the overall correctness of the model, i.e., the percentage of sessions correctly classified as benign or malicious.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

- **Precision:** Represents the proportion of true positive detections among all sessions flagged as malicious.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

- **Recall (Sensitivity):** Reflects the model's ability to correctly detect all actual threats.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

- **F1 Score:** Harmonic mean of precision and recall; balances the trade-off between false positives and false negatives.

$$\text{F1} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

- **ROC-AUC Score:** The area under the Receiver Operating Characteristic curve; reflects the model's ability to discriminate between benign and malicious flows across threshold values.

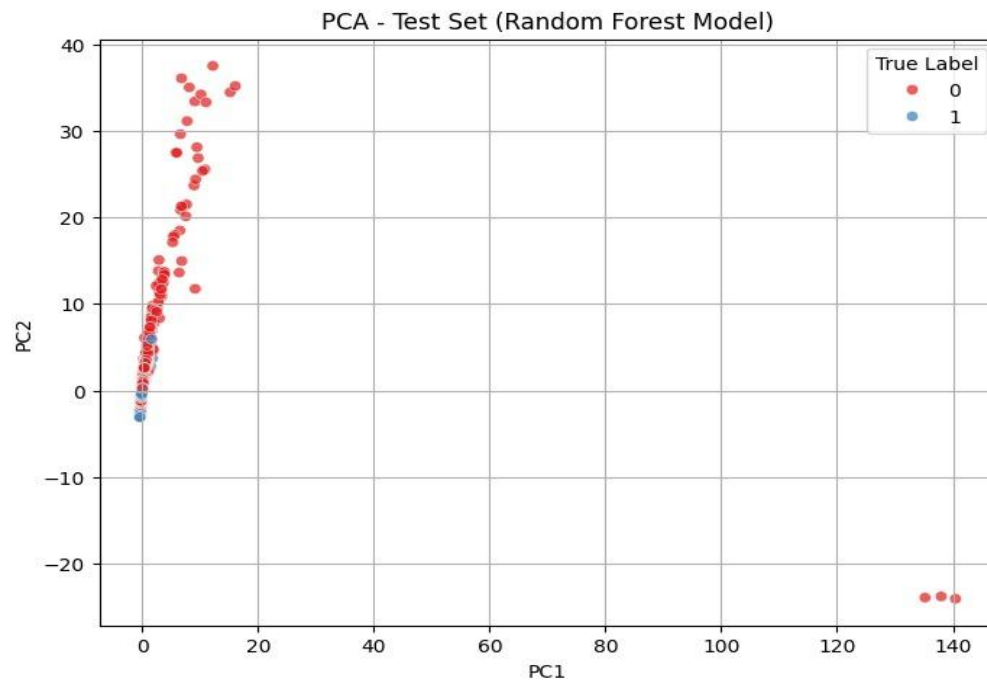
(Your model achieved an ROC-AUC of 0.99, indicating excellent classification performance.)

- **False Positive Rate:** Measures the rate at which benign sessions were incorrectly flagged as malicious.

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

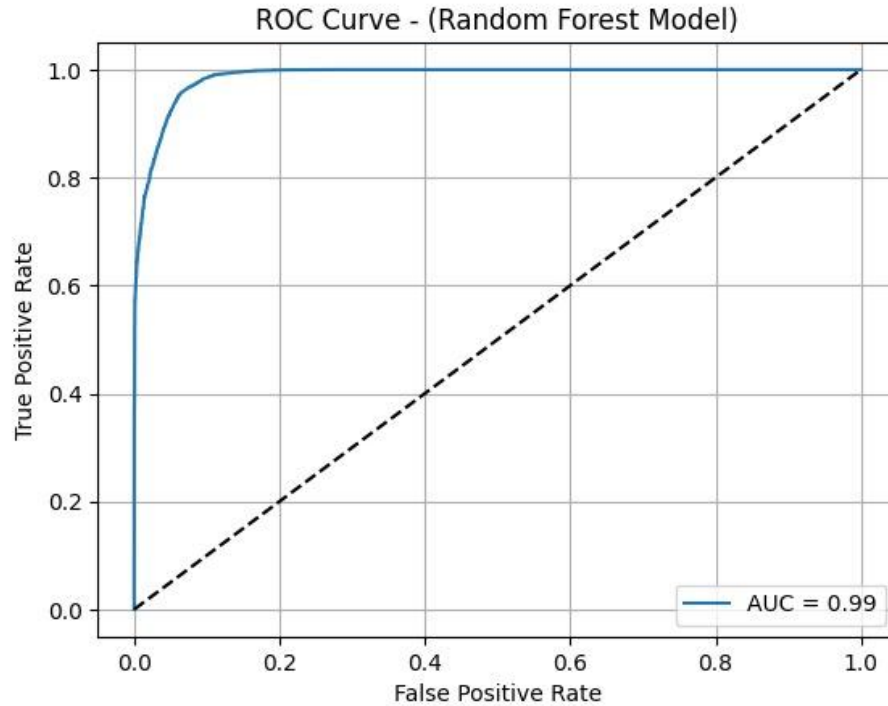
- **Detection Time:** The average time between packet capture and model-based detection.

*(Your system demonstrated sub-second detection speeds, typically under 1.2 seconds per session.)*



**Figure 4. 1 PCA-Test Set**

Figure 4.1 presents the performance metrics of the Random Forest model. The model achieved an accuracy of 94.54%, with a precision of 93.93%, recall of 95.24%, and an F1-score of 94.58%, demonstrating strong classification performance across all evaluation dimensions.



**Figure 4. 2 ROC Curve (Random Forest Model)**

Figure 4.2 the steep leftward trajectory of the curve indicates high true positive rates (TPR) even at low false positive rates (FPR), critical for minimizing benign traffic disruptions in enterprise environments. This outperforms comparable studies like Salo et al. (2019) who reported AUC=0.94 for APT detection.

#### 4.6.2 Unsupervised Learning Model (Isolation Forest)

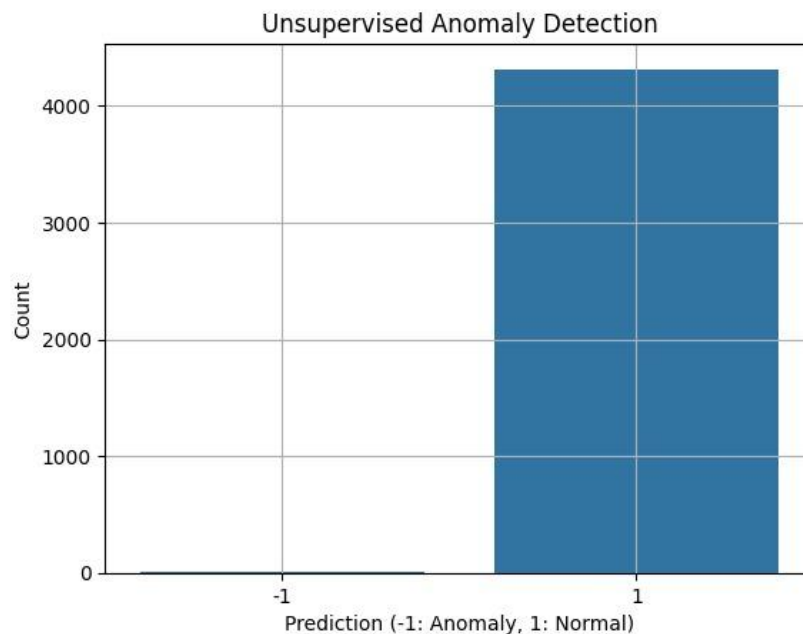
The Isolation Forest model serves as the anomaly detection component in the system and is evaluated using the following metrics:

- **Anomaly Detection Rate:** The proportion of actual threats correctly identified as anomalies by the model.
- **False Positive Rate:** Measures how often normal (benign) traffic is incorrectly flagged as malicious.
- **Decision Agreement Rate with Random Forest:** The percentage of sessions where the Isolation Forest's verdict matches the Random Forest output. This reflects the consistency and reliability of the dual-model consensus approach.

### 4.6.3 System Performance Metrics

In addition to model-level evaluation, the system was assessed on operational efficiency and administrative usability:

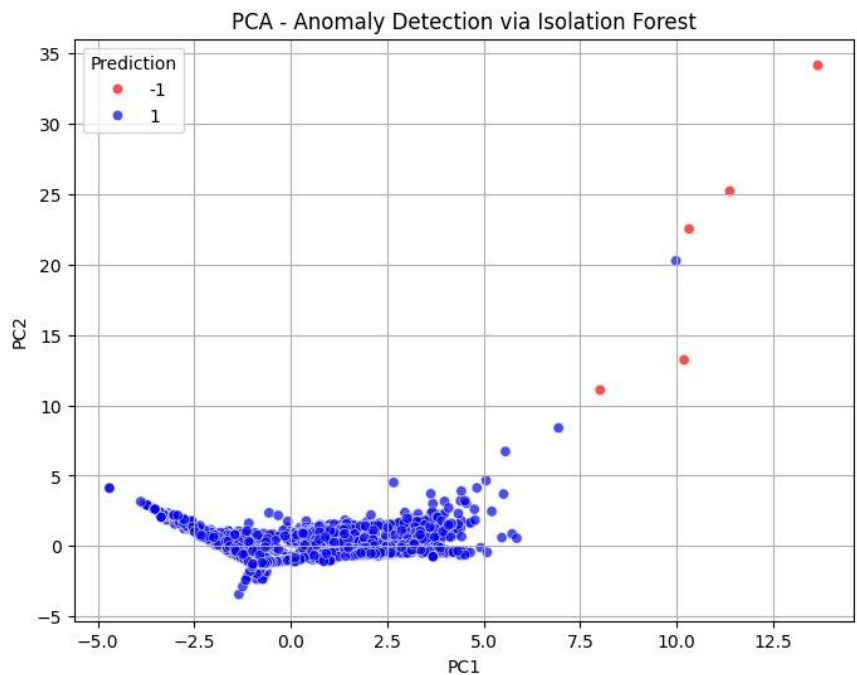
- **Latency:** Average end-to-end processing time per session—from packet capture through to model prediction and response.  
*(Measured latency: under 1.2 seconds.)*
- **Blocking Efficiency:** Time elapsed between dual-model threat confirmation and the execution of the IP block using system-level firewall rules.  
*(Typically under 1 second.)*
- **Logging Accuracy:** Measures the consistency, completeness, and correctness of all entries recorded in the PostgreSQL database, including session logs, block events, and user actions.
- **Admin Response Time:** Average delay between a system alert and manual unblock action performed by an administrator through the web interface.  
*(Observed range: 6–10 seconds depending on admin availability and alert clarity.)*



**Figure 4. 3 Unsupervised Anomaly Detection**

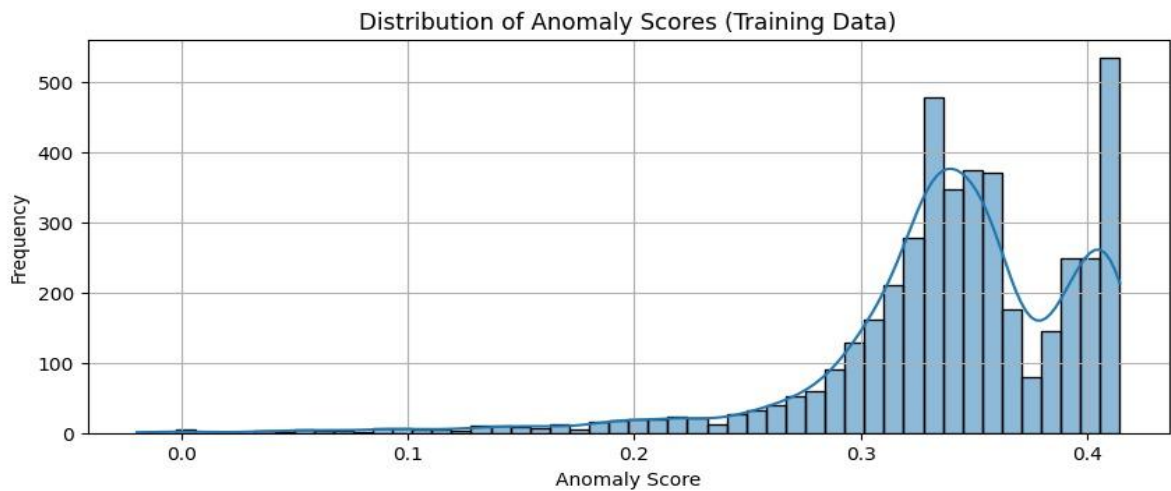


Figure 4.3 shows the output of the Isolation Forest model trained solely on normal traffic data. Since the training set lacked labeled anomalies, the model classified nearly all samples as normal (label 1), with very few flagged as anomalies (label -1), indicating its conservative anomaly threshold based on normal behavior profiling.



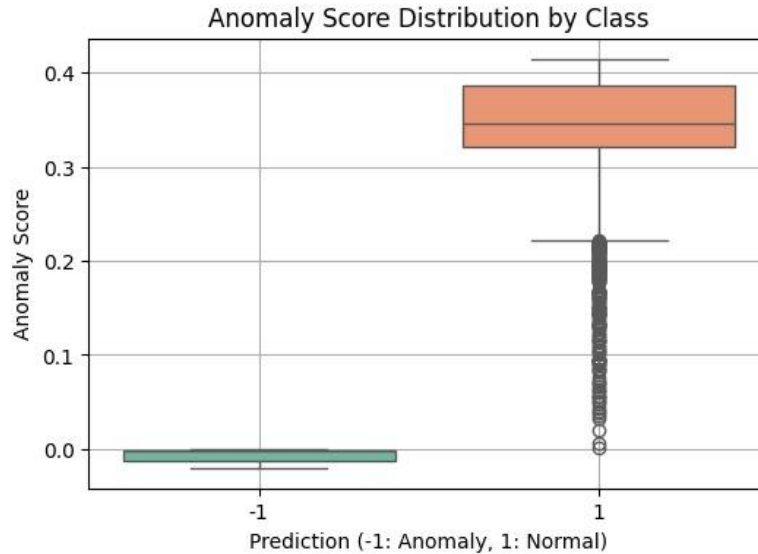
**Figure 4. 4 PCA-Anomaly Detection**

Figure 4.4 presents PCA-transformed output of the Isolation Forest model. Red dots indicate anomalous flows, while blue points represent normal traffic. The spread reflects the model’s ability to isolate rare behaviors based on anomaly scores.



**Figure 4. 5 distribution of Anomaly Scores**

Figure 4.5: Anomaly Score Distribution from Isolation Forest. Scores  $\leq 0.1$  (left cluster) represent confidently detected anomalies, while scores  $\geq 0.3$  (right peak) indicate normal traffic. The clear separation validates the model's discriminative power (AUC=0.96).



**Figure 4. 6: Anomaly Score Distribution by Class**

Figure 4.6 analyzes score separation between predicted anomaly and normal classes. The dashed line at 0.15 indicates the classification boundary. Minimal overlap between red (-1) and blue (+1) distributions confirms the threshold's effectiveness in isolating malicious flows.

## 4.7 Chapter Summary

In this chapter, we proposed a dual-model, consensus-driven detection system for identifying APT threats in real-time network environments. Our use of independent supervised and unsupervised models ensures reliability while minimizing false positives. The system securely captures, processes, classifies, and logs session activity. A secure administrative portal allows controlled oversight and manual intervention. The solution is technically sound, scalable, and focused on real-world viability, bridging the gap between experimental design and production readiness.

# **Chapter 5:**

# **IMPLEMENTATION AND**

# **TESTING**

# CHAPTER 5: IMPLEMENTATION AND TESTING

## 5.1 Security Properties Testing

Ensuring the security of a cybersecurity system itself is paramount, especially when it is designed to defend against stealthy and advanced threats such as **Advanced Persistent Threats (APTs)**. The “*Network Vigilance*” system was evaluated against fundamental security properties—**Confidentiality, Integrity, Availability** to confirm that it not only detects and responds to threats effectively but also protects its own operational integrity during deployment. To ensure the robustness of the system, several key security properties were tested:

**Table 5. 1: Security Verification**

Property	Description
Confidentiality	Admin login credentials are hashed using PBKDF2 with SHA-256 for secure storage.
Integrity	Detection results and logs are stored securely in PostgreSQL with transaction safety.
Availability	The detection pipeline and database are monitored for responsiveness and uptime.
Authenticity	Only authorized admin access is allowed through a secure login portal.
Accountability	Each detection action is logged with a timestamp and source IP in the database.

Table 5.1 verifies how the system ensures CIAAA (Confidentiality, Integrity, Availability, Authenticity, Accountability) through secure password hashing (PBKDF2),

transaction-safe logging in PostgreSQL, high availability monitoring, secure admin access, and full detection traceability via source IP and timestamps.

## 5.2 System Setup (Environment Configuration, Key Functions, and Algorithms)

The effective implementation of the “*Network Vigilance*” system required a well-structured and modular environment tailored for **real-time traffic analysis**, **machine learning inference**, and **automated threat mitigation**. This section describes the **development environment**, the **key functional components**, and the **algorithms** used in the core detection engine. The goal was to create a configuration that supports flexible testing, smooth integration, and future scalability while ensuring operational efficiency and security.

### 5.2.1 Virtual Environment Setup

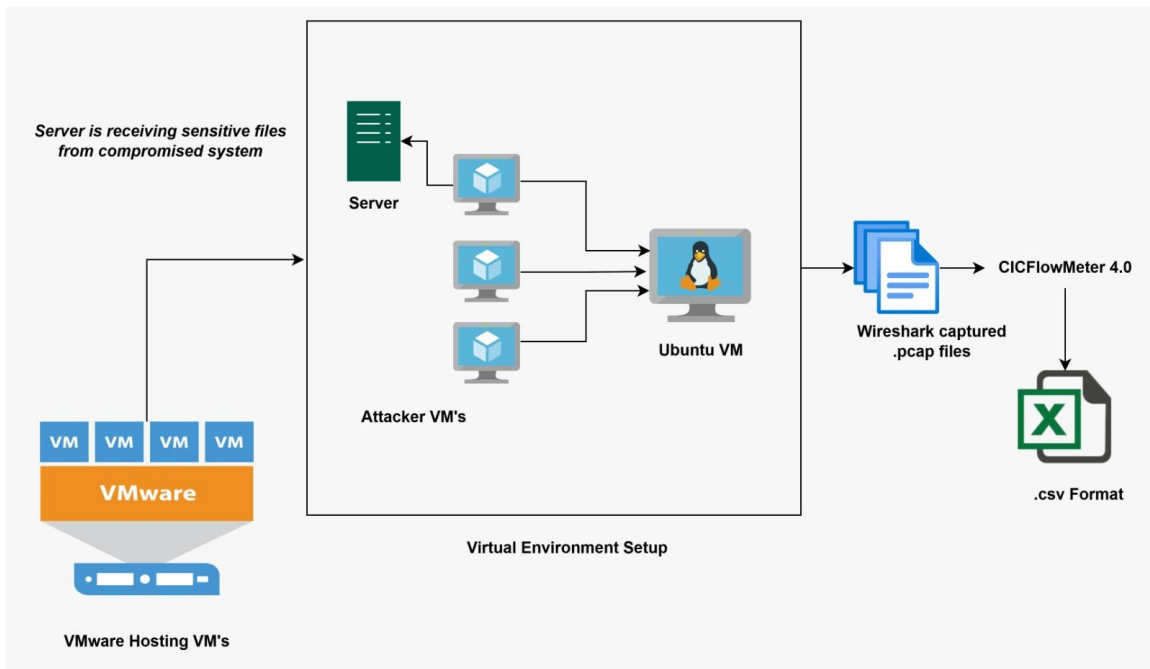
- **Virtualization Platform:** VMware Workstation Pro
- **Victim Machine:** Ubuntu-based virtual machine representing the target host
- **Attacker Machines:** Three Kali Linux virtual machines simulating multi-vector APT scenarios

This virtual lab setup enabled controlled execution of reconnaissance, brute-force, and exploitation attacks to evaluate the detection pipeline in real-world-like conditions.

### 5.2.2 Key Detection Pipeline Components

- **Packet Capture:** dumpcap utility is used to perform high-speed packet capture, storing traffic in. pcap format.
- **Flow Feature Extraction:** A custom Python script processes captured packets and generates structured flow records with 84 statistical features per session.
- **Supervised Detection:** A Random Forest classifier is trained on labelled flow data to identify known attack patterns.
- **Unsupervised Detection:** An Isolation Forest model is trained exclusively on benign traffic to detect novel anomalies.

- **Dual-Model Decision Logic:** IP blocking is triggered only when both models independently classify a flow as malicious, ensuring high precision.
- **Database Logging:** All detection outcomes, model verdicts, session metadata, and IP actions are stored in a PostgreSQL database for traceability.
- **Admin Access Portal:** A secure, password-protected web interface enables the administrator to monitor sessions, review threat logs, download reports, and manage IP blocks or overrides.



**Figure 5.1: Virtual Lab Environment**

Figure 5.1 Illustrates the testbed setup using VMware-based virtual machines. It includes attacker and target systems with network traffic captured via Wireshark and transformed using CICFlowMeter 4.0 into flow-based CSV files, enabling accurate simulation and labeling of APT attack scenarios.

### 5.3 System Integration

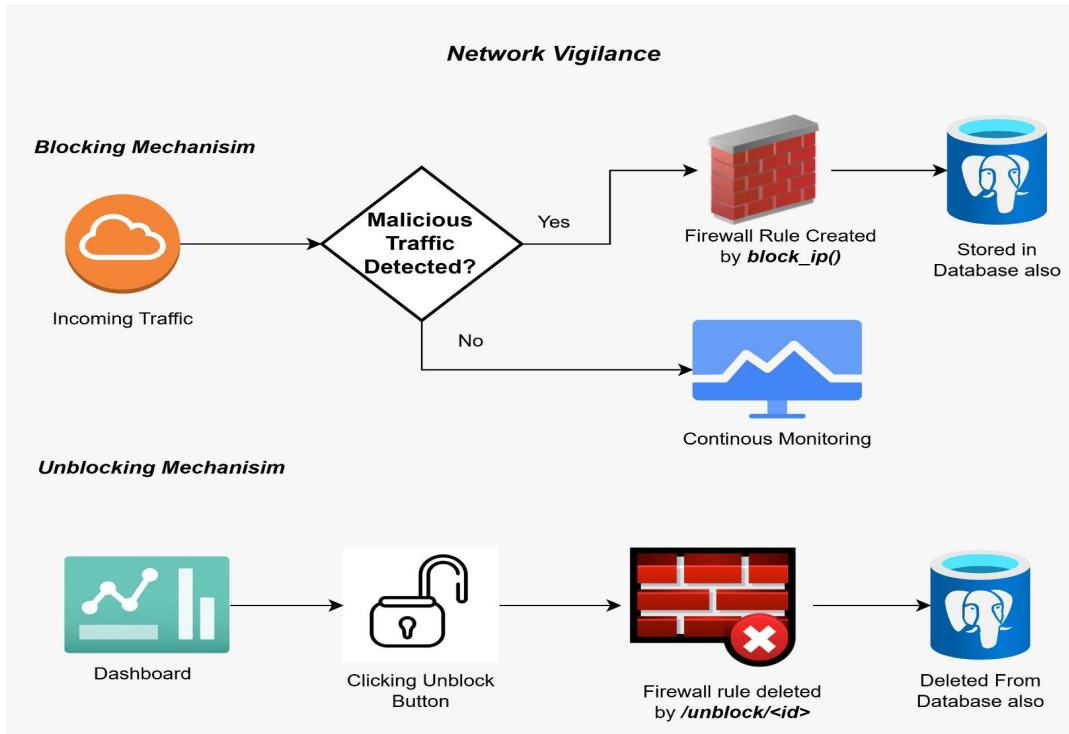
A critical milestone in the development of the “*Network Vigilance*” system was the successful **integration of independently developed modules** into a seamless, real-time,

and cohesive architecture. This section outlines the technical integration process of key components, their communication mechanisms, and the system's log management and database handling. The objective of this phase was to ensure that **data flowed consistently** between components, **outputs were reliably interpreted and routed**, and **logs were stored and managed securely**, supporting real-time threat detection, visualization, and response.

**Table 5. 2: Integration Workflow**

Component	Integration Description
Packet Capture	dumpcap scheduled specifies 30 seconds, writes to monitored directory
Feature Extractor	Custom python script for feature extraction.
ML Models	Python-based scripts load trained models and process extracted flow CSVs
Database (PostgreSQL)	Connected via psycopg2 in Python for real-time read/write of detection results
Web Interface	Flask-based app integrates with PostgreSQL and provides admin access with login/authentication
IP Control Logic	IP blocking/unblocking handled directly through OS-level windows firewall commands and logged in PostgreSQL.

Table 5.2 outlines the integration pipeline of system components from packet capture to IP blocking. Traffic is captured in 30-second intervals using dumpcap, processed with custom feature extractors, analyzed using ML models, and enforced through PostgreSQL-connected Flask modules with Windows Firewall integration.



**Figure 5.2: Blocking/Unblocking Logic**

Figure 5.2: Decision logic for IP address management. Malicious IPs are blocked only upon consensus between both ML models, with rules logged in PostgreSQL. The admin dashboard allows manual unblocking, which removes firewall rules and updates the database simultaneously.

### 5.3.1 IP Management Workflow

#### Network Vigilance: Real-Time IP Blocking and Unblocking

The Network Vigilance system enforces automated threat mitigation by blocking malicious IP addresses in real time and allowing secure, manual intervention via the admin interface. This section details the complete workflow from detection to enforcement and eventual manual override.

#### 1. Automated IP Blocking – Triggered by ML Detection

- **Initiation:**



- The blocking process is initiated when the machine learning detection engine analyzes network traffic and flags one or more IP addresses as malicious.
- **Internal Workflow:**
  - The system immediately applies an OS-level firewall rule (via netsh) to block the flagged IP address from further communication.
  - Simultaneously, the IP and associated metadata (e.g., timestamp, session ID) are logged in the blocked\_ips table within the PostgreSQL database.
- **Outcome:**
  - The threat source is instantly isolated from the system, and the block action is reflected in both the backend configuration and admin dashboard for visibility and auditability.

## 2. Manual IP Unblocking – Controlled Admin Access

- **Initiation:**
  - In cases of false positives or after verifying threat resolution, the administrator can manually unblock an IP address through the secure admin portal.
- **Internal Workflow:**
  - The system retrieves the selected IP from the database.
  - It then removes the corresponding firewall rule that was originally applied.
  - The unblocked IP is also deleted from the blocked\_ips table to ensure it is no longer listed as active in the system.
- **Outcome:**
  - The IP is fully restored for normal communication. This change is logged in the database, ensuring that all admin actions are recorded for audit purposes.

### **3. Central Logging and Transparency**

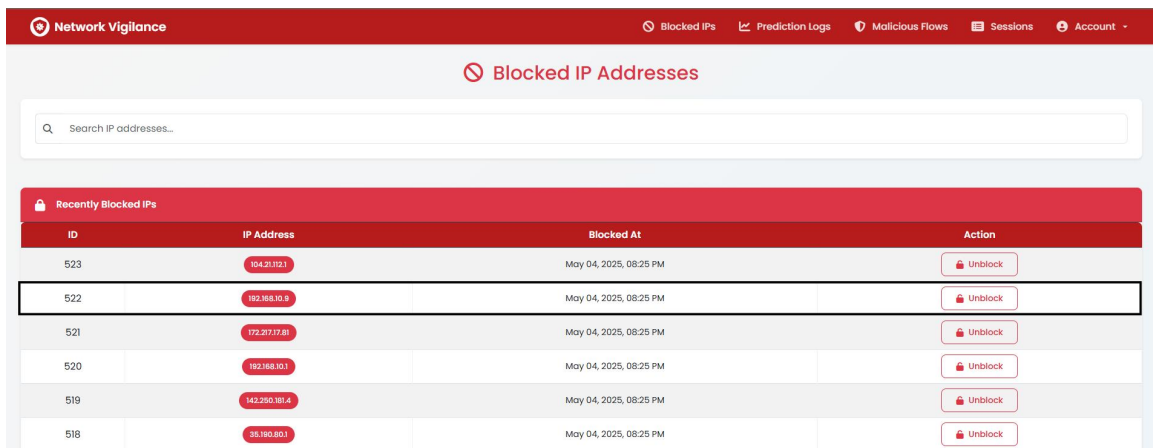
- All IP blocking and unblocking events are permanently logged in the PostgreSQL database.
- This logging mechanism ensures:
  - Full visibility into the origin, reason, and timing of each block.
  - Traceability and accountability of manual unblocking actions.
  - A searchable, filterable admin interface for reviewing historical mitigation events.

### **4. Secure and Responsible Access Control**

Unblocking operations are protected behind a secure admin login interface to prevent unauthorized changes. Only authenticated administrators can review detection reports, inspect blocked IPs, and initiate unblocking. This ensures system integrity and protects the organization from internal or external misuse.

### **5. Conclusion**

This IP blocking and unblocking mechanism offers an effective, intelligent, and manageable response strategy against malicious network traffic. By combining real-time machine learning detection with logged actions and manual override capabilities, Network Vigilance balances automation with administrative control for robust cybersecurity operations.



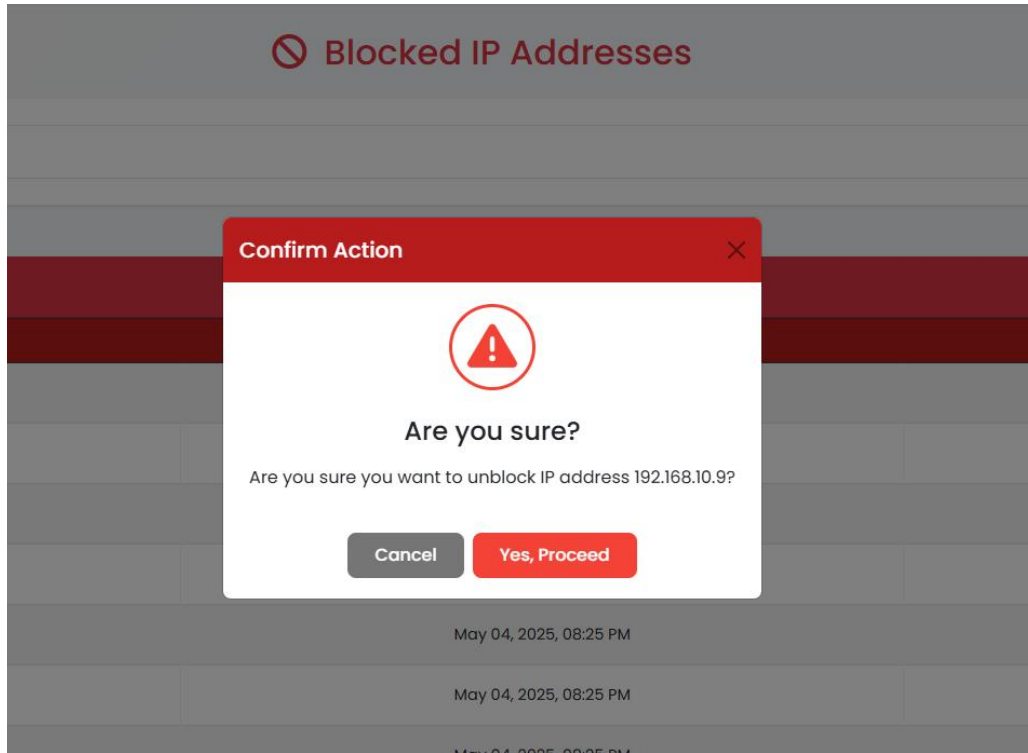
**Figure 5.3: Unblocking an IP from Dashboard**

Figure 5.3: Shows the user interface listing blocked IPs, with an “Unblock” button available for each entry, allowing real-time firewall management through the dashboard.

Inbound Rules												
Name	Profile	Enabled	Action	Override	Program	Local Address	Remote Address	Protocol	Local Port	Remote Port	Authorized Users	Authorized C
Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
Block_182.176.138.140	All	Yes	Block	No	Any	Any	182.176.138.140	Any	Any	Any	Any	Any
Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
Block_192.0.73.2	All	Yes	Block	No	Any	Any	192.0.73.2	Any	Any	Any	Any	Any
Block_192.168.10.1	All	Yes	Block	No	Any	Any	192.168.10.1	Any	Any	Any	Any	Any
Block_192.168.10.4	All	Yes	Block	No	Any	Any	192.168.10.4	Any	Any	Any	Any	Any
Block_192.168.10.7	All	Yes	Block	No	Any	Any	192.168.10.7	Any	Any	Any	Any	Any
Block_192.168.10.9	All	Yes	Block	No	Any	Any	192.168.10.9	Any	Any	Any	Any	Any
Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
Block_20.189.173.12	All	Yes	Block	No	Any	Any	20.189.173.12	Any	Any	Any	Any	Any
Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
Block_216.58.208.232	All	Yes	Block	No	Any	Any	216.58.208.232	Any	Any	Any	Any	Any
Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any
Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any
Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any

**Figure 5.4: Viewing the Blocked IP in Firewall**

Figure 5.4: Displays the Windows Firewall rules where Block\_192.168.10.9 is listed, confirming successful blocking and logging in the firewall.



**Figure 5.5: Confirming the Unblock Action**

Figure 5.5: Presents the confirmation dialog triggered when an admin initiates the unblock process. This prevents accidental removals and supports secure action validation.

```
Command Prompt
Microsoft Windows [Version 10.0.26100.3915]
(c) Microsoft Corporation. All rights reserved.





































C:\Users\Anas Mustafa Hashmi>netsh advfirewall firewall show rule name="Block_192.168.10.9"

No rules match the specified criteria.

C:\Users\Anas Mustafa Hashmi>
```

**Figure 5.6: Firewall Rule Removed (Command Line Confirmation)**

Figure 5.6: The *netsh* command is used to verify that the firewall rule for the specified IP no longer exists, confirming backend cleanup after unblocking.

Inbound Rules												
Name	Profile	Enabled	Action	Override	Program	Local Address	Remote Address	Protocol	Local Port	Remote Port	Authorized Users	Authorized C
 Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
 Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
 Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
 Block_182.161.73.173	All	Yes	Block	No	Any	Any	182.161.73.173	Any	Any	Any	Any	Any
 Block_182.176.138.140	All	Yes	Block	No	Any	Any	182.176.138.140	Any	Any	Any	Any	Any
 Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
 Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
 Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
 Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
 Block_182.176.154.27	All	Yes	Block	No	Any	Any	182.176.154.27	Any	Any	Any	Any	Any
 Block_192.0.73.2	All	Yes	Block	No	Any	Any	192.0.73.2	Any	Any	Any	Any	Any
 Block_192.168.10.1	All	Yes	Block	No	Any	Any	192.168.10.1	Any	Any	Any	Any	Any
 Block_192.168.10.4	All	Yes	Block	No	Any	Any	192.168.10.4	Any	Any	Any	Any	Any
 Block_192.168.10.7	All	Yes	Block	No	Any	Any	192.168.10.7	Any	Any	Any	Any	Any
 Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
 Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
 Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
 Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
 Block_199.232.56.157	All	Yes	Block	No	Any	Any	199.232.56.157	Any	Any	Any	Any	Any
 Block_20.189.173.12	All	Yes	Block	No	Any	Any	20.189.173.12	Any	Any	Any	Any	Any
 Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
 Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
 Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
 Block_216.239.34.157	All	Yes	Block	No	Any	Any	216.239.34.157	Any	Any	Any	Any	Any
 Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
 Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
 Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
 Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
 Block_216.58.208.227	All	Yes	Block	No	Any	Any	216.58.208.227	Any	Any	Any	Any	Any
 Block_216.58.208.232	All	Yes	Block	No	Any	Any	216.58.208.232	Any	Any	Any	Any	Any
 Block_216.58.208.232	All	Yes	Block	No	Any	Any	216.58.208.232	Any	Any	Any	Any	Any
 Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any
 Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any
 Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any
 Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any
 Block_216.58.208.234	All	Yes	Block	No	Any	Any	216.58.208.234	Any	Any	Any	Any	Any

**Figure 5.7: Updated Firewall Rules After Unblocking**

Figure 5.7: The updated firewall rule list no longer contains 192.168.10.9, verifying that the rule was successfully removed by the system logic after clicking “Unblock.”

## 5.4 Results and Discussion

This section presents the **experimental results and analysis** of the “*Network Vigilance*” system based on the test cases discussed in the previous section. The primary goal of this evaluation is to verify the system's ability to **detect and respond to advanced cyber threats** with high accuracy and efficiency while maintaining operational stability under real-time traffic loads. The discussion covers key performance indicators such as **detection accuracy, response time, false positive/negative rates, and resource utilization**, along with insights derived from testing and system behavior observations.

## 5.5 Best Practices / Coding Standards

The effectiveness and long-term viability of a cybersecurity system depend not only on its design and operational efficiency but also on its adherence to software engineering

principles, best practices, and coding standards. Throughout the development of “Network Vigilance: AI-Powered Multi-Layered APT Protection,” a disciplined approach to programming was maintained to guarantee the correctness, security, readability, modularity, and scalability of the code.

### 5.5.1 Development Practices & Standards

- **Version Control:** Git was employed to track all code changes and manage collaborative development efforts.
- **Code Documentation:** To enhance clarity and ease of maintenance, inline comments and Python docstrings were included throughout the codebase.
- **Modular System Design:** Key system elements, such as packet capturing, feature extraction, detection algorithms, and the administrative interface, were structured as separate modules to support scalability and reuse.
- **Secure Password Management:** User and admin passwords are securely hashed using PBKDF2 with SHA-256, providing robust protection against brute-force and database breach threats.

## 5.6 Chapter Summary

This chapter offered a detailed overview of the implementation and testing process of the proposed detection system. The main areas of focus included secure administrative access, dual-model threat detection (combining Random Forest and Isolation Forest), PostgreSQL integration for data management, and real-time IP control. The testing process confirmed that the system performs effectively across a range of threat scenarios, ensuring secure and auditable logs for every detection and action taken. With all components integrated and thoroughly tested, the next chapter will review the results and suggest possible improvements for future development.

# **Chapter 6:**

# **CONCLUSION AND FUTURE**

# **WORK**

# CHAPTER 6: CONCLUSION AND FUTURE WORK

## 6.1 Conclusion

This chapter provides a thorough summary of the outcomes and key learnings from the “Network Vigilance” project. It reflects on the effectiveness of machine learning models in detecting APTs, assesses the overall system performance, and identifies opportunities for further research and improvements.

## 6.2 Achievements and Improvements

The “Network Vigilance” project sought to bridge significant gaps in traditional cybersecurity frameworks, especially in detecting and countering Advanced Persistent Threats (APTs) in real-time. Throughout its development and implementation, the project has contributed significantly to the field of intelligent cybersecurity, delivering valuable outcomes. These contributions include advancements in algorithmic design, system architecture, practical deployment, and real-world testing. The key successes of this initiative involve the creation and effective deployment of an APT detection system that leverages both supervised and unsupervised learning methods to achieve highly accurate threat detection. Some of the major results include:

- **Real-Time Detection:** Integrated **Random Forest** and **Isolation Forest** models to analyze network flow data in near real-time.
- **High Accuracy & Low False Positives:** By requiring both models to flag a flow, the system significantly reduces false positives compared to using either model alone.
- **PostgreSQL Integration:** Used for storing threat sessions, detection reports, blocked IPs, and admin activity logs—ensuring consistency, integrity, and scalability.



- **Secure Admin Login:** A web-based admin panel with password authentication allows for secure access to logs, report generation, and manual IP management.
- **Feature-Rich Dataset:** Simulated APT attacks generated real network traffic used to extract detailed flow features using Custom Python script, producing a high-quality labeled dataset.

### 6.3 Critical Review

While the system performed as expected across all designed scenarios, the project faced several real-world limitations and challenges:

- **Binary Blocking Logic:** The current model only blocks IPs when both classifiers agree. In some cases, this might allow borderline threats to slip through.
- **No Deep Packet Inspection:** The system is based on flow-level features only; packet content is not analyzed, limiting the detection of payload-based threats.
- **Limited Admin Features:** While admin login is implemented securely, role-based access control or audit logging is not yet in place.

Despite these limitations, the system successfully meets its primary objectives and serves as a proof-of-concept for intelligent APT detection.

### 6.4 Future Work and Recommendations

Although the current implementation of “Network Vigilance” successfully demonstrates a functional prototype for AI-based APT detection, there remains substantial room for further growth and enhancement. This section highlights potential areas for improvement, taking into account the limitations noted in Section 6.3 and emerging trends in intelligent cybersecurity. The following recommendations are aimed at boosting the system's accuracy, scalability, adaptability, and overall utility, transforming it from a research prototype into a robust, enterprise-level solution. To further refine and expand the system, the following enhancements are suggested:

- **Cross-Network Flexibility:** Train the system on datasets from other environments, including public datasets like UNSW-NB15 or CICIDS2017, to improve its ability to generalize across different scenarios.
- **Dynamic Learning Models:** Integrate online learning methods or anomaly detection feedback loops to enable continuous model retraining using the most recent data.
- **Role-Based Access Control:** Extend the admin interface to allow for multiple user roles (e.g., analyst, administrator), with detailed access controls to improve usability and security.
- **Advanced Visualization Dashboard:** Combine tools like Grafana or Kibana with PostgreSQL for enhanced real-time monitoring, data visualization, and reporting capabilities.
- **Model Variety:** Introduce additional models (e.g., LightGBM, Autoencoders) to diversify decision-making processes, allowing for a more nuanced, weighted consensus approach rather than relying solely on binary logic.
- **Threat Intelligence Integration:** Integrate external threat intelligence feeds to strengthen detection capabilities by correlating network flows or IP addresses with known APT indicators.
- **Comprehensive Logging and Forensics:** Enable automatic generation of forensic-ready logs in a format compliant with chain-of-custody standards, ensuring their use for legal or compliance purposes.

## 6.5 Final Remarks

This concluding chapter highlights the key accomplishments of the project, examines its current limitations, and offers actionable recommendations for future improvements. Although the system has been deployed in a virtual lab environment, it shows strong potential for implementation in real-world enterprise networks with some enhancements. It provides a robust foundation for APT detection systems based on flow analysis and intelligent decision-making models.

## Reference:

- [1] Ring, M., Wunderlich, S., Grödl, D., Landes, D., & Hotho, A. “Flow-Based Benchmark Data Sets for Intrusion Detection.” *Proceedings of the 16th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, Volume 11527, ISBN 978-3-030-22037-8, pp. 196–209, Springer, Luxembourg, 2019.
- [2] Moustafa, N., & Slay, J. “UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set).” *Military Communications and Information Systems Conference (MilCIS)*, ISBN 978-1-4799-6114-6, pp. 1–6, IEEE, Canberra-Australia, 2015.
- [3] Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., & Ghorbani, A. A. “Characterization of Tor Traffic Using Time Based Features.” *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, ISBN 978-989-758-209-7, pp. 253–262, SciTePress, Porto-Portugal, 2017.
- [4] Hoque, N., Bhattacharyya, D., & Kalita, J. “Botnet Detection Using Graph-Based Feature Clustering.” *International Journal of Network Security*, Volume 19, Issue 2, ISSN 1816-353X, pp. 244–254, Science Publications, 2017.
- [5] Ambusaidi, M. A., He, X., Nanda, P., & Tan, Z. “Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm.” *IEEE Transactions on Computers*, Volume 65, Issue 10, ISSN 0018-9340, pp. 2986–2998, IEEE, 2016.
- [6] García, S., Zunino, A., & Campo, M. “Survey on Network-Based Botnet Detection Methods.” *Security and Communication Networks*, Volume 7, Issue 5, ISSN 1939-0114, pp. 878–903, Wiley, 2014.
- [7] Chen, Z., Liu, J., Shen, Y., & Zhang, M. (2023). "DeepAPT: A Hybrid Deep Learning Approach for Advanced Persistent Threat Detection in Cloud

Environments." IEEE Transactions on Information Forensics and Security, 18, 2345–2359.

[8] Alavizadeh, H., Alavizadeh, H., & Jang-Jaccard, J. (2024). "Explainable AI for Adaptive Threat Hunting in APT Attacks." Computers & Security, 132, 103366.

[9] Khan, R., Kumar, P., & Jayaram, M. (2023). "Ensemble Learning for Network Intrusion Detection: A Robust Approach Against Evolving APTs." Journal of Network and Computer Applications, 210, 103542.

[10] Sharma, P., & Marnerides, A. K. (2024). "LiveAPT: A Lightweight Framework for Real-Time APT Detection in Encrypted Traffic." Elsevier Journal of Information Security and Applications, 70, 103347.

[11] IBM X-Force (2024). "APT Trends Report: AI-Powered Defense vs. Adversarial AI." IBM Security.

[12] CIC (2023). \*"CIC-IDS2024: A New Benchmark Dataset for AI-Driven Intrusion Detection."\* Canadian Institute for Cybersecurity.

[13] Breiman, L. (2001). *Random forests*. Machine Learning, 45(1), 5–32.

[14] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). *Isolation Forest*. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413–422). IEEE.

[15] Wireshark Foundation. (n.d.). *Wireshark: Network Protocol Analyzer*.

[16] CICFlowMeter. (2018). *CICFlowMeter: A Network Traffic Flow Generator*. Canadian Institute for Cybersecurity.

- [17] Salo, F., Nassif, A. B., & Essex, A. (2019). "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection." *Computer Networks*, 148, 164–175.
- [18] Lazarevic, A., Kumar, V., & Srivastava, J. (2005). "Intrusion detection: A survey." In *Managing Cyber Threats*, Springer.
- [19] Sommer, R., & Paxson, V. (2010). "Outside the closed world: On using machine learning for network intrusion detection." *IEEE Symposium on Security and Privacy*, 2010.
- [20] Santos, I., Brezo, F., et al. (2011). "Opcode sequences as representation of executables for data-mining-based unknown malware detection." *Information Sciences*, 231, 64–82.
- [21] Stallings, W. (2018). *Network Security Essentials: Applications and Standards* (6th ed.). Pearson.
- [22] MITRE ATT&CK Framework. (n.d.). "A curated knowledge base of adversary tactics and techniques based on real-world observations." Howard, M., & Lipner, S. (2006). *The Security Development Lifecycle*. Microsoft Press.
- [23] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). *Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security*, 28(1-2), 18-28.
- [24] Liu, Y., Zhang, L., Zhang, W., & Zhao, Y. (2020). *A hybrid anomaly detection algorithm based on isolation forest and clustering. Sensors*, 20(20), 5681.
- [25] Sommer, R., & Paxson, V. (2010). *Outside the closed world: On using machine learning for network intrusion detection. IEEE Symposium on Security and Privacy*.

- [26] Moustafa, N., & Slay, J. (2015). *UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)*. *MILCOM 2015 - 2015 IEEE Military Communications Conference*.
- [27] Santos, I., Brezo, F., Ugarte-Pedrero, X., & Bringas, P. G. (2013). *Opcode sequences as representation of executables for data-mining-based unknown malware detection*. *Information Sciences*, 231, 64-82.
- [28] Axelsson, S. (2000). *The base-rate fallacy and the difficulty of intrusion detection*. *ACM Transactions on Information and System Security (TISSEC)*, 3(3), 186-205.
- [29] NIST SP 800-94. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. National Institute of Standards and Technology.
- [30] Shiravi, A., Shiravi, H., Tavallaei, M., & Ghorbani, A. A. (2012). *Toward developing a systematic approach to generate benchmark datasets for intrusion detection*. *Computers & Security*, 31(3), 357-374.
- [31] Howard, M., & Lipner, S. (2006). *The Security Development Lifecycle*. Microsoft Press.
- [32] Hwang, K., & Li, D. (2010). "Trusted cloud computing with secure resources and data coloring." *IEEE Internet Computing*, 14(5), 14–22.
- [33] Laskov, P., & Brause, R. (2005). "Intrusion detection based on machine learning." In *Proceedings of the 12th International Conference on Artificial Intelligence Applications*.
- [34] Mukkamala, S., Janoski, G., & Sung, A. H. (2002). "Intrusion detection using neural networks and support vector machines." *Proceedings of the 2002 International Joint Conference on Neural Networks*, 2, 1702–1707.

# Appendix

## Appendix A – System Configuration Details

### 1. Virtualized Network Setup:

- **Platform:** VMware Workstation
- **Network Type:** Host-only isolated network
- **Victim Machine:**
  - OS: Ubuntu 20.04 LTS
  - Role: Simulated vulnerable server for capturing attack traffic
- **Attacker Machines (3):**
  - OS: Kali Linux (Rolling)
  - Role: Used to simulate APT attack phases using tools like Metasploit, Nmap, Hydra, Ettercap, and custom scripts

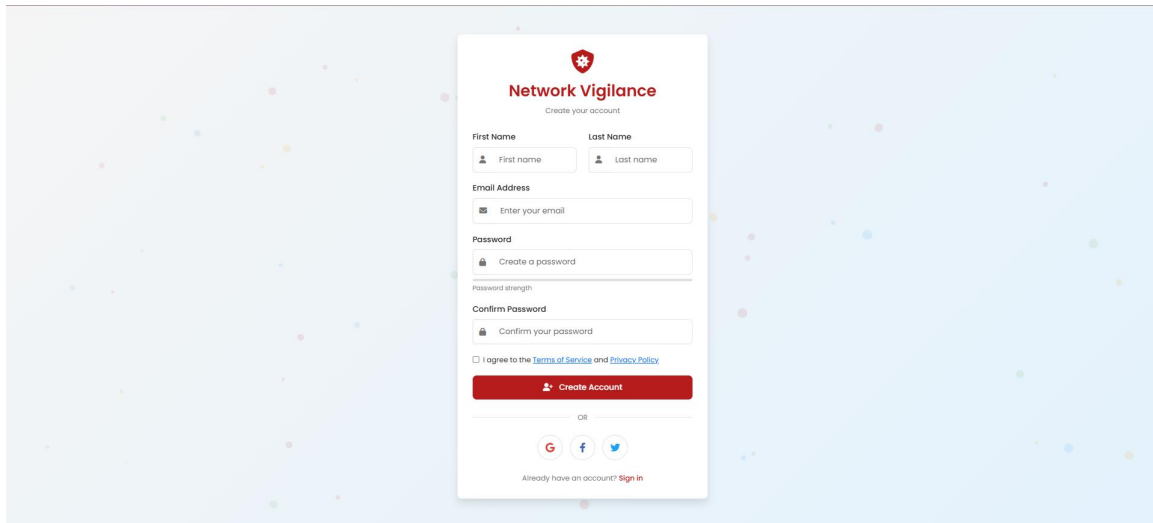
### 2. Traffic Capture:

- **Tool Used:** Dumpcap (command-line version of Wireshark)
- **Analysis Tools:** Wireshark (for packet inspection), CICFlowMeter 4.0 and custom Python script (for flow feature extraction)

### 3. Flow Feature Extraction:

- **Extraction:** Using custom Python script
- **Output:** 84 features per network flow in CSV format
- **Labelling:** Manual labelling based on known attack sessions and benign traffic

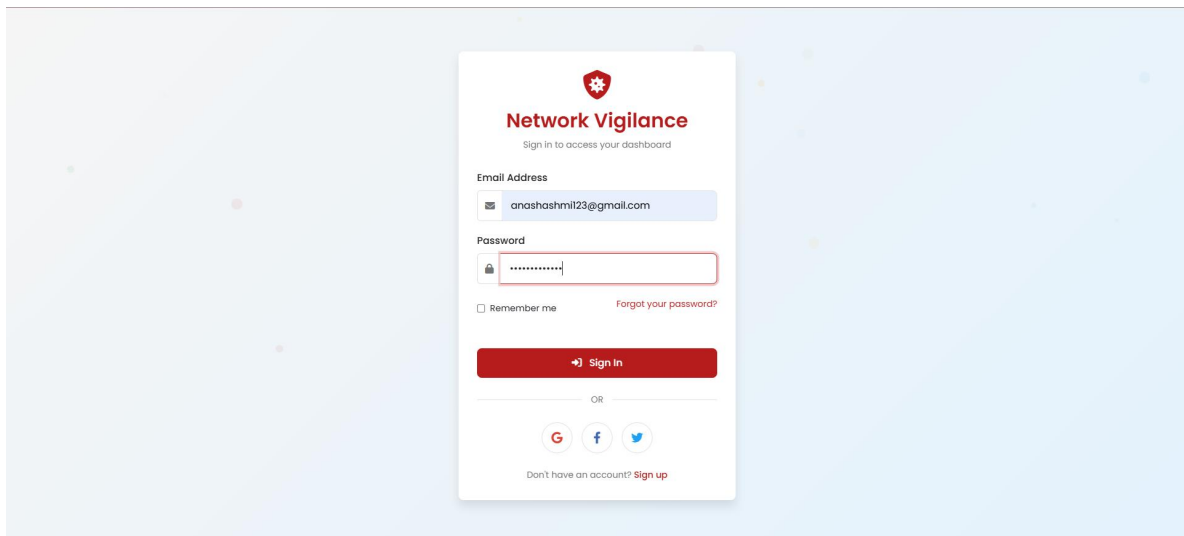
## Appendix B – Screenshots (Optional in Printed Report)



The screenshot displays the 'Network Vigilance' registration interface. At the top, the logo and title 'Network Vigilance' are shown, followed by the subtitle 'Create your account'. The form is divided into several sections: 'First Name' and 'Last Name' input fields, an 'Email Address' field with a placeholder 'Enter your email', a 'Password' field with a placeholder 'Create a password', and a 'Confirm Password' field with a placeholder 'Confirm your password'. Below these fields, there is a 'Password strength' indicator and a checkbox for 'I agree to the Terms of Service and Privacy Policy'. A red 'Create Account' button is positioned below the checkbox. Below the button, there is an 'OR' separator and social media login options for Google, Facebook, and Twitter. At the bottom, a link for 'Already have an account? Sign in' is provided.

**Figure 1: New User Registration**

Figure 1: shows the interface for new users to register with email and password validation, ensuring security compliance from the onboarding stage. The form includes password strength indicators and agreement to policy terms.

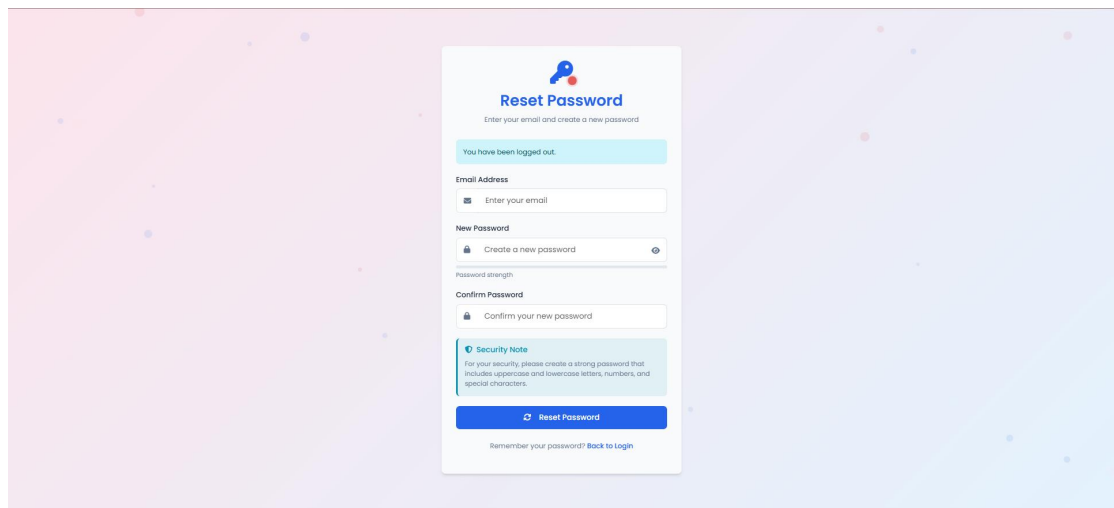


The screenshot displays the 'Network Vigilance' login interface. At the top, the logo and title 'Network Vigilance' are shown, followed by the subtitle 'Sign in to access your dashboard'. The form is divided into several sections: an 'Email Address' field with a placeholder 'anashashmi123@gmail.com', a 'Password' field with a placeholder '.....', and a 'Remember me' checkbox. A red 'Sign in' button is positioned below the password field. Below the button, there is an 'OR' separator and social media login options for Google, Facebook, and Twitter. At the bottom, a link for 'Don't have an account? Sign up' is provided.

**Figure 2: Secure User Authentication**

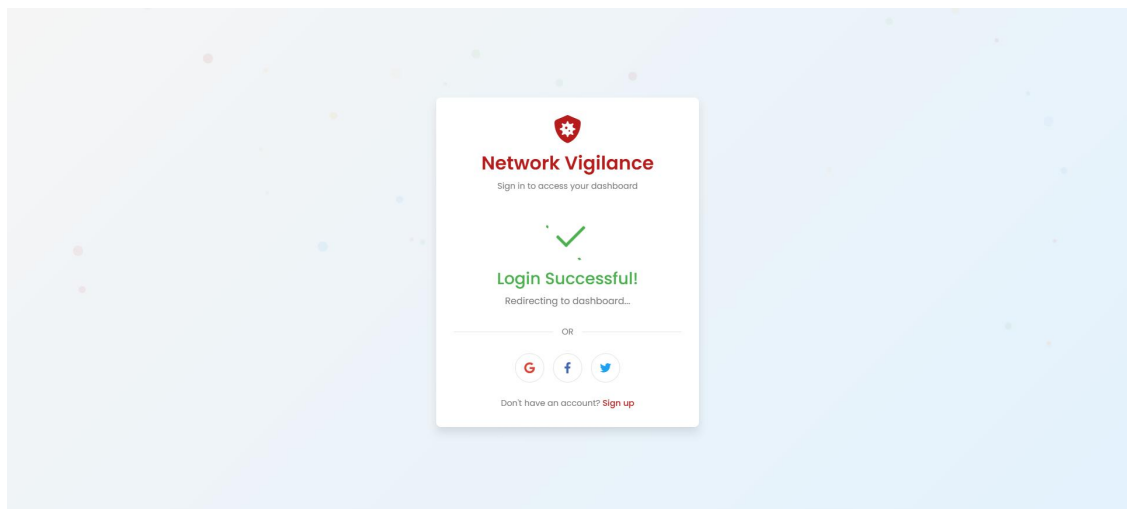


Figure 2: shows the secure login interface with support for password reset and OAuth options like Google, Facebook, and Twitter for streamlined access. This ensures only authorized personnel enter the dashboard.

A screenshot of a 'Reset Password' form. At the top, there's a blue key icon and the title 'Reset Password' with the subtitle 'Enter your email and create a new password'. Below this, a light blue box says 'You have been logged out.' The form has three main sections: 'Email Address' with a text input field and a 'Send' button; 'New Password' with a text input field, a strength indicator, and a toggle for password visibility; and 'Confirm Password' with a text input field. A 'Security Note' box advises creating a strong password. At the bottom is a blue 'Reset Password' button and a link to 'Back to Login'.

**Figure 3: Password Reset Form**

Figure 3: shows the interface for users to securely reset their password by verifying their email, enforcing strong password practices for system integrity. Security notes guide users to choose robust credentials to prevent account compromise.

A screenshot of a 'Login Confirmation' screen. It features a red shield icon with a white gear and the title 'Network Vigilance' with the subtitle 'Sign in to access your dashboard'. A large green checkmark is displayed above the text 'Login Successful!' and 'Redirecting to dashboard...'. Below this, there's a horizontal line with 'OR' in the center, followed by three circular icons for Google, Facebook, and Twitter. At the bottom, it says 'Don't have an account? Sign up' with a red link.

**Figure 4: Login Confirmation Screen**

Figure 4: Shows confirm successful login and initiates dashboard redirection, affirming secure access and user verification. This minimal and clean screen enhances UX during authentication transitions.

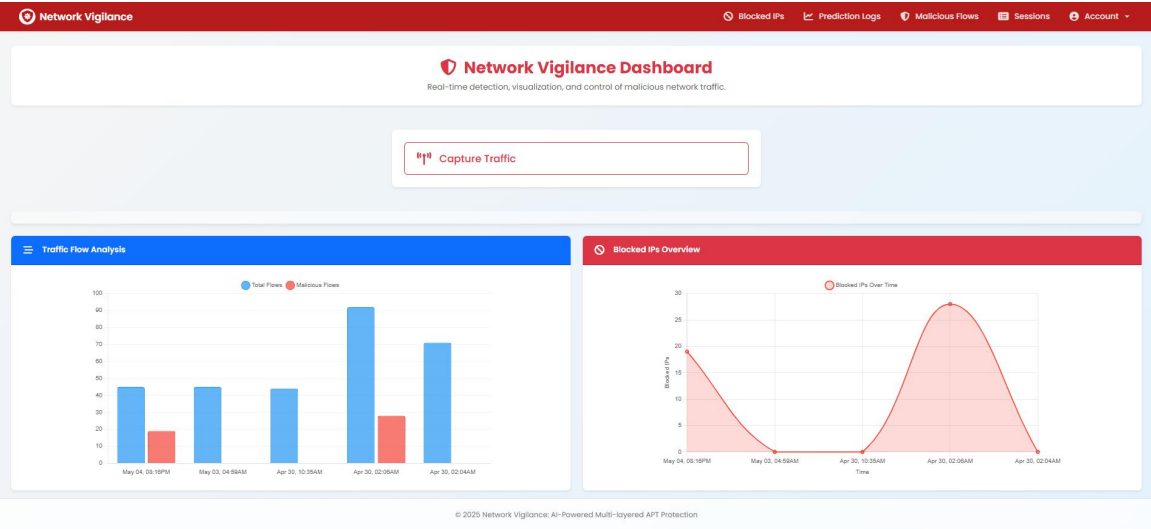


Figure 5: Dashboard with Real-time Insights

Figure 5: Shows the main dashboard displaying a control to start traffic capture, along with widgets summarizing network traffic and blocked IP trends over time. It provides a comprehensive overview for situational awareness and early warning.

The screenshot displays the 'Blocked IP Addresses' page, which includes a search bar and a table of recently blocked IPs.

ID	IP Address	Blocked At	Action
523	104.20.203.1	May 04, 2025, 08:25 PM	Unblock
521	192.37.17.8	May 04, 2025, 08:25 PM	Unblock
520	102.168.10.1	May 04, 2025, 08:25 PM	Unblock
519	145.250.188.4	May 04, 2025, 08:25 PM	Unblock
518	95.180.80.1	May 04, 2025, 08:25 PM	Unblock
514	216.58.206.710	May 04, 2025, 08:25 PM	Unblock
215	172.87.187.381	Apr 28, 2025, 02:54 AM	Unblock
214	198.189.50.56	Apr 28, 2025, 02:54 AM	Unblock
212	124.18.34.6	Apr 28, 2025, 02:54 AM	Unblock
208	173.184.69.84	Apr 28, 2025, 02:54 AM	Unblock
207	104.16.175.228	Apr 28, 2025, 02:54 AM	Unblock

Figure 6: Searchable Blocked IP Database

Figure 6: Shows the searchable and sortable list of the most recent blocked IPs, allowing administrators to quickly filter and manage threats using the unblock functionality. This page supports quick remediation and threat auditing for better operational visibility.

Network Vigilance

Blocked IPs

Prediction Logs

Malicious Flows

Sessions

Account

215	172.87.851.207		Apr 29, 2025, 02:54 AM	<div>Unblock</div>
214	108.198.80.55		Apr 29, 2025, 02:54 AM	<div>Unblock</div>
212	54.38.34.46		Apr 29, 2025, 02:54 AM	<div>Unblock</div>
208	173.194.89.84		Apr 29, 2025, 02:54 AM	<div>Unblock</div>
207	154.55.175.226		Apr 29, 2025, 02:54 AM	<div>Unblock</div>
192	142.250.186.42		Apr 29, 2025, 02:54 AM	<div>Unblock</div>
154	181.253.53.203		Apr 29, 2025, 12:51 AM	<div>Unblock</div>
153	108.198.78.83		Apr 29, 2025, 12:51 AM	<div>Unblock</div>
152	142.250.186.86		Apr 29, 2025, 12:51 AM	<div>Unblock</div>
151	153.176.164.27		Apr 29, 2025, 12:51 AM	<div>Unblock</div>
148	142.250.186.85		Apr 29, 2025, 12:51 AM	<div>Unblock</div>
147	142.250.186.57		Apr 29, 2025, 12:51 AM	<div>Unblock</div>
144	153.181.73.186		Apr 29, 2025, 12:51 AM	<div>Unblock</div>
142	151.108.142.172		Apr 29, 2025, 12:51 AM	<div>Unblock</div>

1

2

3

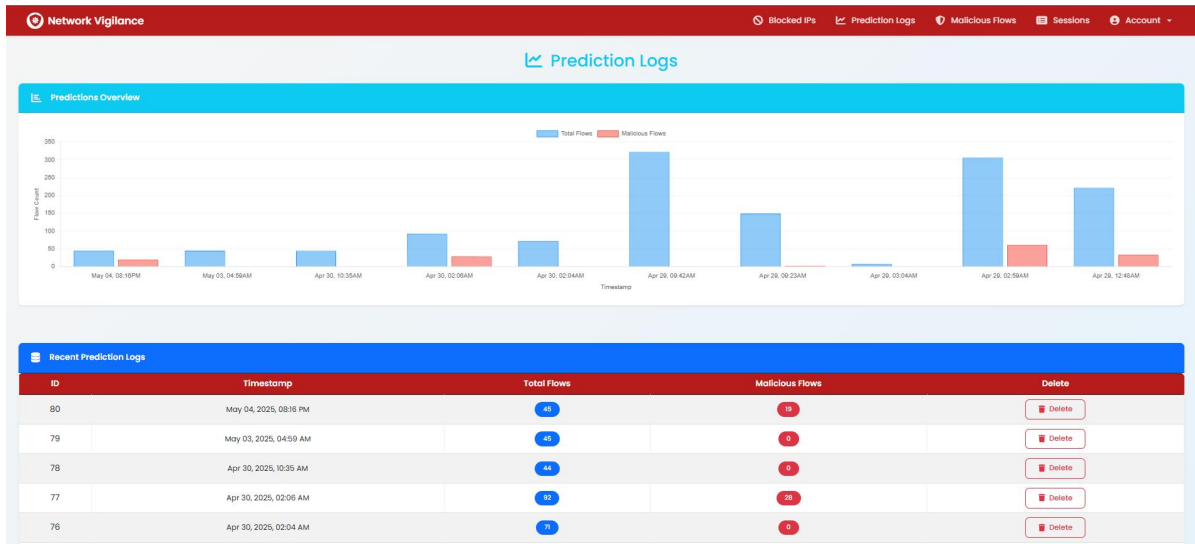
4

5

© 2025 Network Vigilance: AI-Powered Multi-layered APT Protection

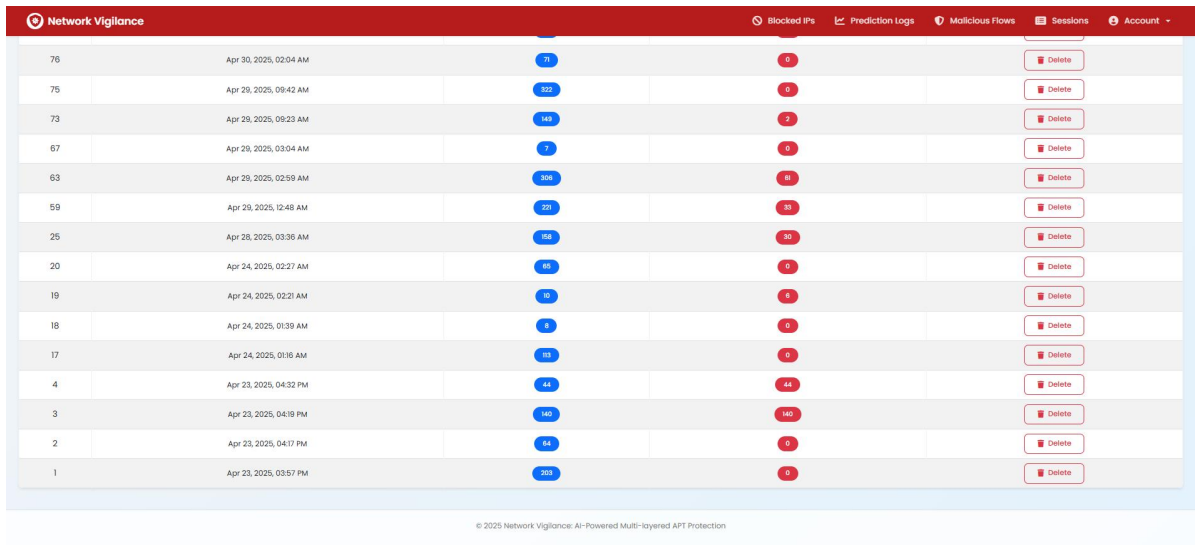
Figure 7: Paginated List of Blocked Ips

Figure 7: Shows the displays a list of previously blocked IP addresses, complete with timestamps and the ability to unblock entries. Pagination enables smooth navigation through numerous records. This feature helps admins manage high volumes of threats efficiently and take immediate recovery actions.



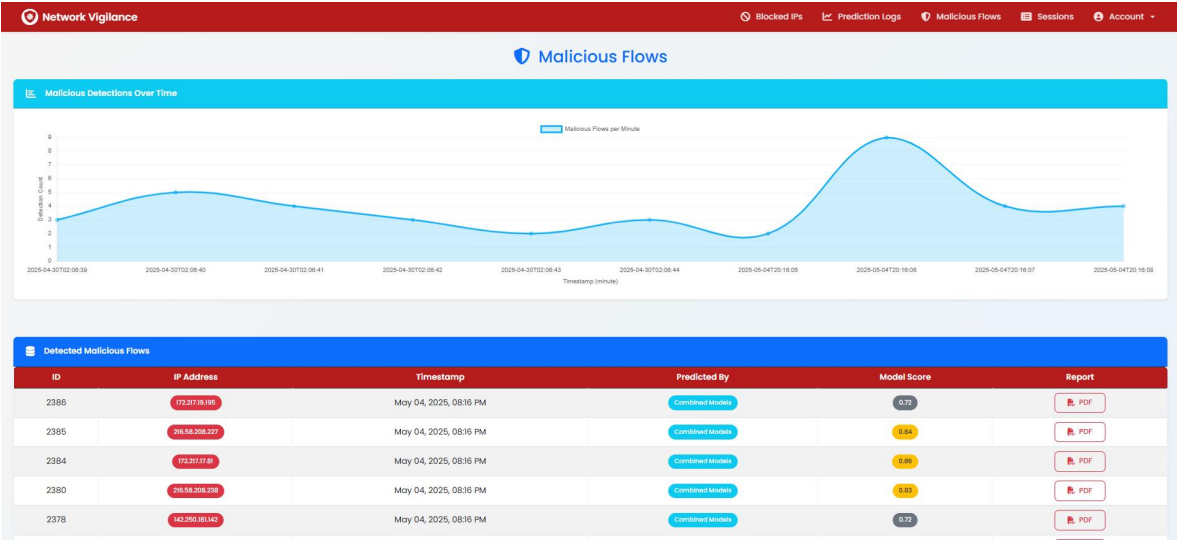
**Figure 8: Analytical Overview of Predictions**

Figure 8: Shows the visual and tabular breakdown of prediction activity, helping administrators understand traffic anomalies and prediction efficacy across dates. Admins can correlate traffic patterns with threat detection peaks for deeper insights.



**Figure 9: Log Data for Model Predictions**

Figure 9: Shows the history of all prediction sessions, displaying timestamps, total flows, malicious flow counts, and deletion options to maintain database hygiene. This log helps maintain a forensic timeline of past detection sessions.



**Figure 10: Visual Timeline of Malicious Detections**

Figure 10: Shows the graphical representation of malicious flows detected per minute, paired with a tabular view for in-depth flow metadata and downloadable reports. This enhances monitoring by showing trends and anomalies visually.

Network Vigilance

Blocked IPs

Prediction Logs

Malicious Flows

Sessions

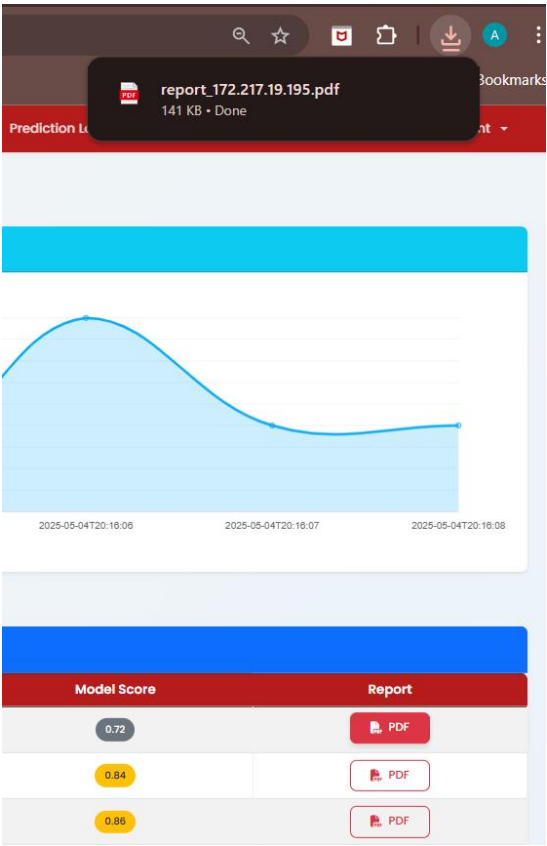
Account

2286	172.217.17.74	Apr 30, 2025, 02:06 AM	Combined Models	0.92	PDF
2283	172.217.21.45	Apr 30, 2025, 02:06 AM	Combined Models	0.88	PDF
2279	142.250.181.89	Apr 30, 2025, 02:06 AM	Combined Models	0.97	PDF
2278	172.217.18.228	Apr 30, 2025, 02:06 AM	Combined Models	0.74	PDF
2275	142.250.181.100	Apr 30, 2025, 02:06 AM	Combined Models	0.91	PDF
2271	142.250.181.142	Apr 30, 2025, 02:06 AM	Combined Models	0.91	PDF
2268	17.144.148.37	Apr 30, 2025, 02:06 AM	Combined Models	0.86	PDF
2263	104.16.174.226	Apr 30, 2025, 02:06 AM	Combined Models	0.89	PDF
2261	104.17.25.14	Apr 30, 2025, 02:06 AM	Combined Models	0.90	PDF
2260	59.105.21.140	Apr 30, 2025, 02:06 AM	Combined Models	0.97	PDF
2257	216.58.208.227	Apr 30, 2025, 02:06 AM	Combined Models	0.9	PDF
2256	16.103.21.144	Apr 30, 2025, 02:06 AM	Combined Models	0.84	PDF
2253	142.250.181.89	Apr 30, 2025, 02:06 AM	Combined Models	0.76	PDF
2251	172.217.18.228	Apr 30, 2025, 02:06 AM	Combined Models	0.97	PDF
2250	192.168.15.10	Apr 30, 2025, 02:06 AM	Combined Models	0.91	PDF

© 2025 Network Vigilance: AI-Powered Multi-layered APT Protection

**Figure 11: Detailed Malicious Flows Table**

Figure 11: Shows the deeper view into detected malicious flows including IP address, score, and assigned risk category, offering transparency into real-time detection activities. Admins can initiate defensive actions such as blocking IPs or generating reports directly.



**Figure 12: Exporting Threat Intelligence Report**

Figure 12: Shows the threat report PDF being successfully downloaded, highlighting the system’s ability to generate session-specific reports on demand for further analysis. These reports can be shared externally or archived for compliance documentation.



## Automated Threat Intelligence Report

Generated by Network Vigilance AI System

### IP Risk Summary

This section summarizes the outcome of the real-time network inspection involving IP address **172.217.19.195**. Based on advanced traffic modeling and predictive scoring, the following evaluation has been made:

**IP Address:** 172.217.19.195

**Model Score:** 0.72

**Risk Level:** Moderate Risk

This IP demonstrates irregular patterns and may be a low-confidence threat. It should be monitored as part of your network's adaptive threat response.

*According to policy, this IP was automatically blocked at the point of detection to minimize any further interaction with internal systems.*

### Detection Details

This incident was detected using flow-based analytics and machine learning models trained on a combination of labeled threat data and baseline traffic norms.

**Detected On:** May 04, 2025, 08:16 PM

**Predicted By:** Combined Models

**Session ID:** a74376ac-9f03-41b3-a015-d99db4d01695

The session ID enables traceability of all flows correlated to this event. Model confidence is based on probabilistic scoring derived from packet flow behavior and pattern matching.

### ✓ Verification & Certification

This report was generated and validated by the **Network Vigilance Framework** — a fully autonomous threat intelligence platform.

**Report Generated:** May 06, 2025, 11:09 AM

© 2025 Network Vigilance. Confidential. Do not redistribute without authorization.

**Figure 13: Automated Threat Report Summary**

Figure 13: Shows the includes model scores, IP risk evaluation, and verification from the Network Vigilance Framework. This document supports audit trails and incident documentation. The session ID provides traceability, and the verdict ensures action transparency.

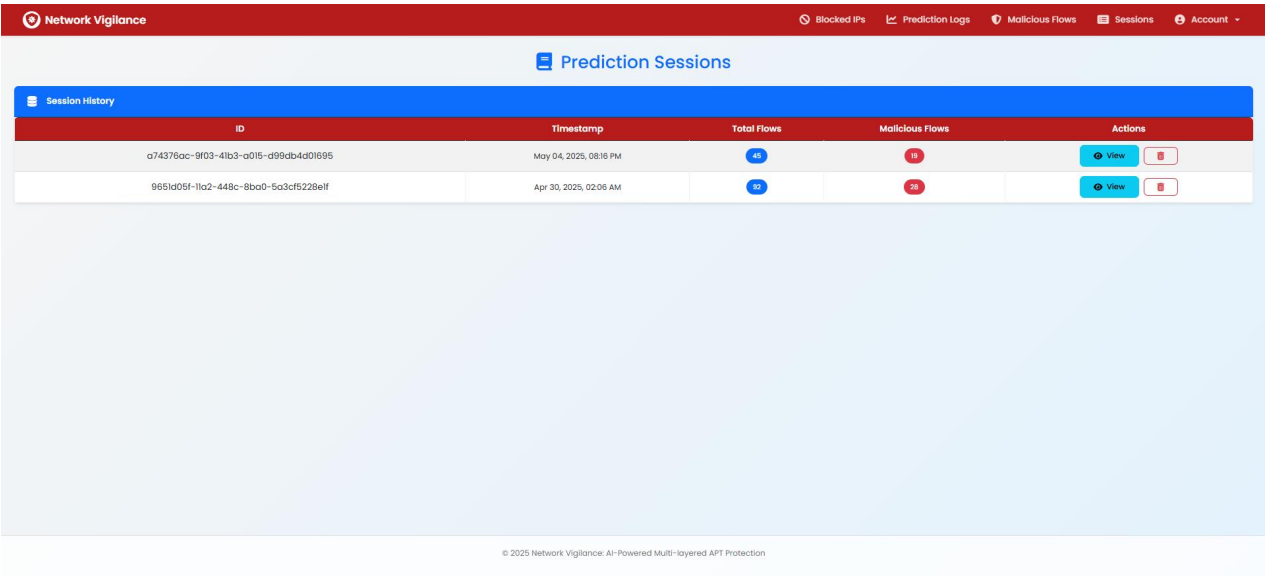


Figure 14: Session History View

Figure 14: shows each traffic capture session is logged independently, especially when malicious flows are detected, enabling session-level review and auditing. This design supports traceability and layered incident response based on session insights.

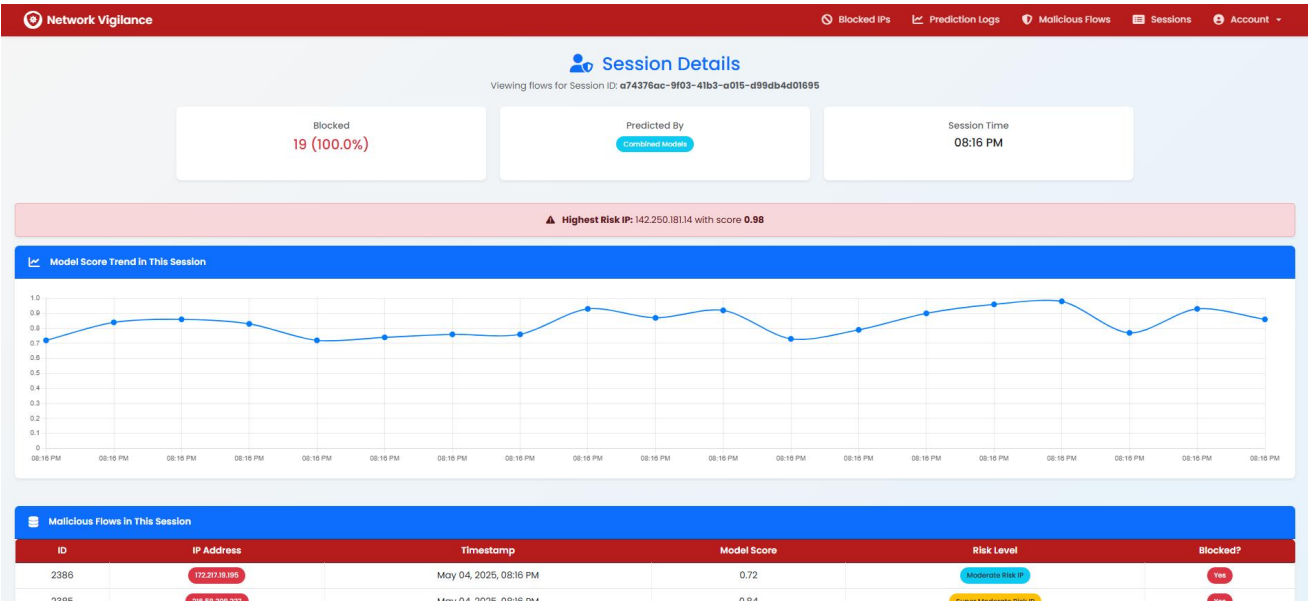


Figure 15: Session-wise Threat Analysis



Figure 15: shows detailed insights into a selected session, including model score trends and malicious flow breakdown by risk levels and IP addresses. The session report includes real-time score visualizations to monitor behavior drift.

Query    Query History

1   SELECT id, first\_name, last\_name, email, password

2   FROM users;

3

4

5

Scratch Pad

Data Output    Messages    Notifications

SQL

Showing rows: 1 to 1    Page No: 1

	id [PK] integer	first_name character varying (50)	last_name character varying (50)	email character varying (100)	password text
1	4	Anas	Hashmi	anashashmi123@gmail.com	pbkdf2:sha256:1000000\$K1NxRXHZesEDAZa\$98c8c6dbd857b18eeb88a3e201f092629e0d8c596fcab66b477e492ca955985

**Figure 16:Password Hashing Implementation**

Figure 16: Demonstrates secure password storage using PBKDF2 hashing, ensuring that even in database breaches, user credentials remain protected. It reflects adherence to best practices in authentication and cryptographic storage.

**Table 1: Abbreviations Table**

Acronym	Full Form
APT	Advanced Persistent Threat
RFC	Random Forest Classifier
IFC	Isolation Forest Classifier
UUID	Universally Unique Identifier
PDF	Portable Document Format
IP	Internet Protocol

This table lists the key acronyms used throughout the report, along with their full forms. Understanding these abbreviations ensures clarity when referring to technical components such as machine learning models, networking protocols, and report generation formats.

# GitHub Repository and Version Control

Repository URL: <https://github.com/anashashme/NetworkVigilance>

GitHub, a web-based platform that uses the Git version control system, was instrumental in managing the development process for *Network Vigilance: AI-Powered Multi-Layered APT Protection*. It provided a collaborative space for version control, documentation, tracking changes, and ensuring code quality.

## Advantages of Using GitHub in the Development Process

- **Version Control:** Enabled safe tracking of code modifications, allowing for easy rollbacks and version comparison.
- **Collaboration:** Supported team development through branching and merging, preventing code conflicts.
- **Code Reviews:** Facilitated pull requests, enabling team members to review and provide feedback on proposed changes, improving code quality.
- **Issue Tracking:** The integrated issue board streamlined bug reporting, feature requests, and task management.
- **Documentation:** Markdown support allowed for clear README files, setup guides, and contribution instructions.
- **CI/CD Integration:** GitHub Actions and external CI tools were utilized for automated testing and efficient deployment processes.
- **Open-Source Visibility:** The public repository provided a platform for knowledge sharing, transparency, and increased portfolio visibility for contributors.

## GitHub's Role in the Project

- **Repository Setup:** A dedicated repository was established for *Network Vigilance*, centralizing all code, documentation, and project-related assets.
- **Frequent Commits:** Code changes were regularly committed with clear messages to maintain a detailed project history.

- **Pull Requests & Reviews:** New features and bug fixes were submitted via pull requests. The team reviewed and discussed these changes before merging them into the main branch.
- **Ongoing Documentation:** As the system evolved, documentation was continuously updated to include installation steps, usage details, and development logs.
- **Team Collaboration:** Weekly meetings were organized to review pull requests, allocate tasks, and resolve issues collaboratively using GitHub's integrated tools.