VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018

A PROJECT REPORT

On

"Azure SOC Honeynet-Proactive Cyber Defense in Action"

Submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

Submitted by

Bhavana K C	**(1RG22CS022)**
Likhitha A	**(1RG22CS044)**
Nithishree Shetty	**(1RG22CS056)**
Dhanushree R	**(1RG23CS404)**

Under the Guidance of

Mrs. Bhagyashri Wakde

Asst. Professor, Dept of CSE RGIT,

Bengaaluru-32

Department of Computer Science & Engineering

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

Cholanagar, R.T.Nagar Post, Bengaluru-560032

2025-2026

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University)

Cholanagar, R.T.Nagar Post, Bengaluru-560032

Department of Computer Science & Engineering

CERTIFICATE

Project Phase-II

This is to certify that the Project Report titled **"Azure SOC Honeynet-Proactive Cyber Defense in Action"** is a bonafide work carried out by **Bhavana K C (USN 1RG22CS022), Likhitha A (USN 1RG22CS044), Nithishree Shetty (USN 1RG22CS056) and Dhanushree R (USN 1RG23CS404)** in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belgavi**, during the year **2025-2026.** It is certified that all corrections/suggestions given for Internal Assessment have been incorporated in the report. This project report has been approved as it satisfies the academic requirements in respect of project work (BCS786) prescribed for the said degree.

Signature of Guide	Signature of HOD	Signature of Principal
Mrs. Bhagyashri Wakde	**Dr Arudra A**	**Dr. D G Anand**
Asst. Professor	Head Of Department	Principal
Dept. of CSE, RGIT, Bengaluru	Dept. of CSE, RGIT, Bengaluru	RGIT, Bengaluru

External Viva

Name of the Examiners	Signature with date
1. _____	_____
2. _____	_____

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgavi-590018

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We hereby declare that the project work entitled **"Azure SOC Honeynet-Proactive Cyber Defense in Action"** submitted to the **Visvesvaraya Technological University, Belagavi** during the academic year **2025-2026**, is record of an original work done by us under the guidance of **Mrs. Bhagyashri Wakde,** Assistant Professor, Department of Computer Science and Engineering, RGIT, Bengaluru in the partial fulfillment of requirements for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering.** The results embodied in this project have not been submitted to any other University or Institute for award of any degree or diploma.

Bhavana K C **(1RG22CS022)**

Likhitha A **(1RG22CS044)**

Nithishree Shetty **(1RG22CS056)**

Dhanushree R **(1RG23CS404)**

ACKNOWLEDGEMENT

We take this opportunity to thank our college **Rajiv Gandhi Institute of Technology, Bengaluru** for providing us with an opportunity to carry out this project work.

We express our gratitude to **Dr. D G Anand**, Principal of RGIT, Bengaluru for providing the resources and support without which the completion of this project would have been a difficult task.

We extend our sincere thanks to **Dr. Arudra**, Associate Professor and Head, Department of Computer Science and Engineering, RGIT, Bengaluru, for being a pillar of support and encouraging us in the face of all adversities.

We would like to acknowledge the through guidance and support extended towards us by **Mrs. Bhagyashri Wakde,** Assistant Professor, Dept. of CSE, and we would like to thank our project coordinators for their immense support towards us by **Dr. Latha P H,** Professor, Dept. of CSE, RGIT, Bengaluru and **Mrs. Bhagyashri Wakde,** Assistant Professor, Dept. of CSE, RGIT, Bengaluru. Their incessant encouragement and valuable technical support have been of immense help. Their guidance gave us the environment to enhance our knowledge and skills and to reach the pinnacle with sheer determination, dedication and hard work.

We also want to extend our thanks to the entire faculty and support staff of the Department of Computer Science and Engineering, RGIT, Bengaluru, who have encouraged us throughout the course of the Bachelor's Degree. We want to thank our family for always being there with full support and for providing us with a safe haven to conduct and complete our project. We are ever grateful to them for helping us in these stressful times. Lastly, we want to acknowledge all the helpful insights given to us by all our friends during the course of this project.

Bhavana K C **(1RG22CS022)**

Likhitha A **(1RG22CS044)**

Nithishree Shetty **(1RG22CS056)**

Dhanushree R **(1RG23CS404)**

ABSTRACT

In today's rapidly evolving cyber-threat landscape, organizations need proactive and intelligent defenses beyond traditional tools that often fail to detect APTs and zero-day attacks. This project focuses on designing and deploying honeynets—deceptive networks that lure attackers, divert them from critical systems, and capture valuable data on intrusion techniques and behaviors. By integrating these honeynets into Security Operations Center (SOC) workflows, the project enhances real-time monitoring, situational awareness, and incident response capabilities. Using open-source and custom tools, the honeynet environment is configured to collect actionable intelligence that strengthens threat detection, reduces false positives, and improves overall cybersecurity resilience.

CONTENTS

LIST OF FIGURES / ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In the current cybersecurity landscape, organizations face an ever-evolving array of threats from malicious actors seeking to exploit vulnerabilities in their systems. The need for robust security measures has never been more critical. One such measure is the deployment of honeynets—networks of decoy systems designed to lure attackers away from critical assets and gather intelligence on their tactics, techniques, and procedures. Honeynets serve as a valuable tool for understanding the behavior of cybercriminals and enhancing the security posture of organizations.

A Security Operations Center (SOC) plays a pivotal role in the defense strategy of an organization. It is responsible for monitoring, detecting, and responding to security incidents in real-time. By utilizing advanced tools and technologies, a SOC ensures that potential threats are identified and mitigated before they can cause significant harm. Integrating honeynets into the SOC's operations can provide additional insights and strengthen the overall security framework.

1.1 Problem Statement

In the face of escalating cyber threats, traditional security measures are increasingly inadequate, necessitating advanced strategies such as honeynets and Security Operations Centers (SOCs) to effectively detect and mitigate sophisticated attacks. This project aims to address the challenges of integrating honeynet deployment with SOC operations within the Microsoft Azure cloud environment to enhance real-time threat detection, analysis, and response, ultimately bolstering network defenses and providing valuable insights into cybercriminal behavior.

- Complex configuration and deployment of honeypots within a cloud environment.
- Efficient data collection and analysis from multiple honeypots.
- Seamless integration of various security tools within Azure.
- Real-time monitoring and detection of sophisticated cyber threats.

- Effective incident response procedures to minimize impact.
- Ensuring minimal disruption to legitimate network activities.
- Scalability of SOC operations to handle large volumes of security data.
- Management of false positives and reducing alert fatigue.
- Training and retaining skilled cybersecurity personnel.
- Continuous updating and adaptation to evolving cyber threat landscapes.

1.2 Motivation

Cyberattacks have evolved from isolated incidents into organized, large-scale operations capable of crippling entire infrastructures. Global statistics indicate a steady increase in the number and sophistication of attacks on both private and public networks. Even small organizations with limited digital presence are no longer safe.

The motivation for this work stems from the urgent need to move away from reactive cybersecurity—where response occurs only after an incident—to proactive defense, where threats are anticipated and neutralized in advance. Traditional on-premise SOCs involve heavy hardware investment and depend largely on human analysts. This makes them slow, expensive, and prone to oversight.

A cloud-driven SOC, on the other hand, offers:

- **Scalability:** resources can grow with organizational needs;
- **Automation:** faster response and reduced human dependency;
- **Centralized visibility:** data from multiple sources viewed through unified dashboards;
- **Accessibility:** security monitoring available from anywhere, at any time.

The personal motivation behind this project is also educational—to explore how modern cloud services can be combined to create affordable, intelligent defense platforms that bridge academic research and industry practice.

1.3 Importance of Proactive Cyber Defense

In the modern digital landscape, organizations face a continuously evolving range of cyber threats that target every layer of their infrastructure — from operating systems to cloud services and user credentials. A reactive defense model, which only responds after an incident occurs, is no longer adequate.

The need of the hour is proactive cybersecurity, which involves continuous monitoring, predictive analysis, and automated response mechanisms.

Proactive defense focuses on identifying potential threats before they cause harm. Through the deployment of technologies such as honeypots and behavioral analytics, it enables early detection of attack attempts and immediate counteraction. In this project, the Azure SOC Honeynet serves as a foundation for this approach by simulating vulnerable systems that intentionally attract attackers. These interactions reveal the latest tactics and vulnerabilities that can later be patched in real environments.

The proactive model not only enhances response speed but also contributes to long-term learning. By collecting and studying attack data, cybersecurity analysts can design stronger defensive strategies. This shift from reaction to anticipation marks a fundamental evolution in how security systems are conceptualized and deployed.

1.4 Background of the Study

In the current era of digital transformation, organizations depend heavily on technology for communication, data storage, and service delivery. While this digitalization increases efficiency, it simultaneously opens new doors for cyber threats. Every day, malicious actors exploit vulnerabilities in networks, cloud infrastructures, and applications to gain unauthorized access or disrupt operations. Over the past decade, the frequency and sophistication of cyberattacks have grown at an alarming rate, affecting sectors such as banking, healthcare, education, and government services.

The rise of advanced persistent threats (APTs), zero-day exploits, ransomware campaigns, and data-exfiltration attacks demonstrates how traditional defense mechanisms—like signature-based antivirus software and static firewalls—are no longer sufficient. Attackers employ automation, artificial intelligence, and polymorphic malware that can adapt faster than manual security configurations. Consequently, the need for proactive, intelligent, and adaptive security mechanisms has become essential rather than optional.

This research project focuses on building such a proactive system through the integration of Security Operations Centers (SOCs) and Honeynet technologies within Microsoft Azure. By combining the analytical power of Azure's cloud-native security services with the intelligence-gathering capabilities of honeynets, organizations can detect, study, and neutralize attacks more effectively.

The concept represents a shift from reactive incident handling to predictive and preventive cybersecurity defense.

1.5 Need for the Study

The complexity of modern IT infrastructures—comprising hybrid clouds, remote endpoints, and third-party integrations—makes monitoring and threat detection increasingly challenging. Reactive strategies often fail because alerts are triggered *after* an attack has already penetrated the system. By then, sensitive information may have been compromised or operations disrupted. Therefore, there is an urgent requirement for a proactive detection and response framework that can predict, identify, and contain cyber threats in real time.

Integrating honeynets within an Azure-based SOC satisfies this requirement. Honeynets function as decoy environments designed to mislead attackers. Any interaction with these systems is inherently suspicious, which allows analysts to focus immediately on genuine threats rather than sifting through thousands of false positives. Meanwhile, Azure provides elastic resources, intelligent analytics, and automated response mechanisms that transform data from these honeynets into actionable insights.

Moreover, the study addresses a real-world problem faced by many organizations—a shortage of skilled cybersecurity personnel capable of interpreting large volumes of security data. By automating detection and integrating learning-based models, an Azure SOC Honeynet can augment human expertise and ensure consistent monitoring even in resource-constrained settings. The need for this research thus stems from the growing demand for scalable, affordable, and automated cybersecurity solutions suitable for both academic learning and professional deployment.

1.6 Aim and Objectives of the Project

The aim of this project is to design, implement, and evaluate a robust cybersecurity environment that unifies honeynet deception technologies with an Azure-based SOC for real-time threat detection and response.

To realize this aim, the project is guided by the following objectives:

1. To design a cloud-based honeynet environment using Microsoft Azure Virtual Machines and databases configured as decoys to attract malicious traffic.

2. To integrate honeynet data sources with Azure Sentinel and Log Analytics Workspace for centralized monitoring and correlation.
3. To automate the alerting and response process through Azure Logic Apps and playbooks that react to specific indicators of compromise.
4. To analyze attacker behavior using Kusto Query Language (KQL) and identify patterns that reveal tactics, techniques, and procedures (TTPs).
5. To strengthen cybersecurity education by providing a reproducible lab model that demonstrates real-world defensive operations.
6. To evaluate system performance in terms of accuracy, scalability, and incident-response efficiency.

The primary objectives of this project are to create a comprehensive cybersecurity lab that integrates honeynet deployment with SOC operations within the Microsoft Azure cloud environment. The key objectives include:

- **Set Up a Honeynet:** Deploy a honeynet to detect and analyze cyber threats, configuring honeypots that mimic real systems to attract attackers and capture their activities.
- **Utilize Azure for SOC Operations:** Implement and operate a SOC within Azure, leveraging its security services and tools to monitor and respond to incidents effectively.
- **Enhance Security Posture:** Use the data and insights gained from the honeynet to improve the organization's security measures and incident response strategies.
- **Educational Value:** Provide a hands-on learning experience in cybersecurity, focusing on the practical aspects of setting up and managing a honeynet and a SOC.
- **Continuous Monitoring:** Ensure continuous monitoring of the network environment to detect and respond to threats in real-time.
- **Incident Response:** Develop and implement efficient incident response procedures to mitigate the impact of detected threats.
- **Data Analysis:** Collect and analyze data from the honeynet to identify trends and patterns in attack behavior.

1.7 Future Scope and Sustainability

- **Data Collection and Analysis:** The project will focus on collecting data from the honeynet, including logs and alerts, and analyzing this data to identify trends and patterns in attack behavior. This analysis will help in understanding the threat landscape and improving defense mechanisms.
- **Incident Response:** The project will outline the incident response procedures followed by the SOC in case of detected threats. This includes steps for containment, eradication, and recovery from security incidents
- **Limitations:** The project will be limited to a controlled experimental environment and may not cover the deployment of honeynets and SOC operations in a production setting. Additionally, the project will focus on a specific set of tools and technologies available in Azure, and the findings may not be generalizable to other cloud environments or security platforms.

Cyber threats are expected to grow in complexity with the advent of 5G networks, artificial intelligence, and Internet-of-Things (IoT) devices. The current project establishes a foundation upon which future enhancements can be built. Machine-learning models can later be integrated into Azure Sentinel to predict attack trends based on historical honeynet data. Similarly, blockchain-based integrity mechanisms can be explored for securing SOC log authenticity.

From a sustainability perspective, hosting cybersecurity infrastructure in the cloud reduces hardware waste, energy consumption, and maintenance costs compared to traditional data centers. The scalable pay-as-you-go model of Azure ensures resource optimization and cost control, making this approach both technically and economically sustainable for long-term research and organizational deployment.

1.8 Azure SOC–Honeynet Integration Overview

Microsoft Azure offers a powerful and flexible cloud platform capable of hosting complex security architectures. Within this environment, a Security Operations Center acts as the nerve center of defensive activities, continuously collecting and analyzing data from multiple sources such as virtual machines, networks, user accounts, and applications.

In the proposed architecture, **honeypots** are strategically deployed across isolated virtual networks. These honeypots mimic real servers, databases, or user systems, inviting potential attackers to interact with them. Every action—be it a login attempt, port scan, or file upload—is captured and transmitted to Azure Sentinel. Sentinel then correlates this honeynet data with telemetry from other legitimate assets to identify suspicious trends.
Azure's integrated tools enhance this workflow:

- **Azure Security Center** enforces compliance and manages security policies.
- **Log Analytics Workspace** serves as the centralized data repository, storing millions of logs for fast querying.
- **Azure Key Vault** secures credentials and encryption keys used by monitoring agents.
- **Azure Monitor** and **Workbooks** visualize security metrics and performance trends in real time.

1.9 Existing System

Traditional cybersecurity relies heavily on firewalls, intrusion detection systems, and antivirus software. These legacy solutions monitor traffic at predefined boundaries and use rule-based detection. While effective for known threats, they fail to adapt to evolving attack vectors such as polymorphic malware or coordinated multi-stage intrusions.

Existing SOC implementations also face challenges:

- Decentralized log management leading to fragmented data analysis.
- Over-dependence on security personnel for alert review.
- High false-positive rates that waste investigation time.
- Lack of predictive analytics to forecast future attacks.

1.10 Proposed System

The proposed system introduces a cloud-based, fully automated Security Operations Center capable of real-time detection, response, and analysis. Microsoft Azure is used as the primary platform because of its scalability, integrated services, and global availability.

Key features of the proposed model include:

- Honeynet Deployment: Multiple VMs configured as decoys to attract and record malicious activities.
- Data Collection: All logs aggregated within Azure Log Analytics Workspace.
- Threat Detection: Azure Sentinel correlates events and identifies suspicious patterns using analytic rules.
- Automated Response: Logic Apps trigger actions such as blocking IP addresses or sending alerts.
- Visualization: Dashboards display real-time metrics and incident status for security analysts.

The system transforms cybersecurity operations from manual to automated mode, significantly reducing detection and reaction time. Its modular design allows easy integration with AI-based detection or third-party threat-intelligence feeds in future expansions.

1.11 Outcome of the Project

The project titled “Azure SOC Honeynet: Proactive Cyber Defense in Action” successfully demonstrated the creation of a complete cloud-based Security Operations Center (SOC) integrated with honeynet technology using Microsoft Azure. The system achieved its main objective of detecting, analyzing, and mitigating cyber threats in real time through the integration of Azure Sentinel, Log Analytics, Security Center, and Logic Apps. Through multiple configuration phases, testing procedures, and analytical queries, the system efficiently captured simulated and real attack attempts, analyzed their behavior, and generated automated responses to neutralize threats. The overall deployment proved to be secure, scalable, and highly effective for proactive cyber defense.

1.11.1 Technical and Functional Outcomes

The project resulted in the successful implementation of an automated security monitoring environment that combines the capabilities of a SOC with a honeynet. Several Kusto Query Language (KQL) queries were executed to validate the system's performance in event detection, alert correlation, and automated response.

Key results include:

- Real-Time Attack Detection: The system accurately detected repeated unauthorized access attempts and network intrusions in real time, ensuring immediate awareness of security events.
- Threat Intelligence Correlation: Using analytical queries and log data, the system identified malicious IP addresses and abnormal access patterns, improving incident analysis.
- Performance Monitoring: Continuous system health monitoring ensured optimal utilization of resources, with alerts triggered automatically for unusual CPU usage or traffic spikes.
- Network Anomaly Identification: Log queries helped identify unusual inbound and outbound traffic patterns, proving that the honeynet successfully captured reconnaissance and brute-force behaviors.
- Database Security Observations: Honeypot SQL logs revealed suspicious query patterns, validating the efficiency of the setup in attracting and logging injection-based attack attempts.

These outcomes confirmed that the SOC environment was capable of real-time detection, automated alert generation, and correlation-based incident analysis, establishing a robust framework for cyber defense and monitoring.

1.11.2. Analytical and Research Insights

The data collected through continuous honeynet monitoring provided valuable insight into attacker behavior and evolving threat trends. The log correlation and visualization features of Azure Sentinel revealed that intrusion attempts were consistent and occurred at varying time intervals, indicating the automated nature of most cyberattacks.

1.11.3 Educational and Skill Development Outcomes

The project offered an excellent opportunity for practical learning and hands-on experience in the field of cybersecurity and cloud security operations. By working with real-time data and security tools, the team developed proficiency in:

- Deploying and configuring virtualized honeypots in a secure cloud environment.
- Creating and managing security rules and alert policies in Azure Sentinel.
- Writing and executing KQL queries for event filtering and incident correlation.
- Implementing Logic App workflows for automated threat mitigation.
- Interpreting visual dashboards to assess the organization's security posture in real time.

The practical exposure gained through this project bridges the gap between theoretical learning and real-world cybersecurity implementation. It demonstrates how cloud technologies can be effectively leveraged to design advanced, intelligent, and responsive defense systems.

1.11.4 System Performance Evaluation

Comprehensive performance testing confirmed that the implemented system was both stable and reliable. Automated alerts were generated instantly upon detection of suspicious behavior, while response workflows were triggered with minimal delay. The use of cloud-based automation significantly improved the average response time compared to traditional manual methods.

The Azure infrastructure provided strong scalability, allowing the addition of more honeypots and analytic rules without affecting system stability. The project also demonstrated the ability to handle high data volumes efficiently, maintaining consistent uptime and accurate event reporting even during simulated attack bursts.

These outcomes validate the capability of cloud-based SOC systems to function under continuous monitoring with high accuracy and minimal maintenance.

1.11.5 Practical Impact and Industrial Relevance

This project successfully illustrates how cloud technology can simplify and enhance the process of building advanced cybersecurity infrastructures. The outcomes prove that organizations can achieve effective monitoring and protection without relying on expensive hardware or complex on-premises setups.

The key practical impacts include:

- Enabling cost-efficient and scalable cybersecurity operations.
- Supporting real-time incident visibility through centralized dashboards.
- Automating threat response actions, reducing dependency on human monitoring.
- Providing a flexible architecture that can be adapted to various industry requirements.

The project thus provides a working reference model for modern organizations seeking to establish SOC environments integrated with intelligent data analytics and honeynet technologies.

1.11.6 Overall Outcome

In conclusion, the Azure SOC Honeynet project successfully demonstrated an end-to-end cybersecurity framework that integrates threat intelligence collection, analysis, and automated response mechanisms. The system proved its ability to detect attacks proactively, analyze threats efficiently, and initiate timely countermeasures.

This project establishes that cloud-based SOC environments, when integrated with honeynet technologies, can serve as powerful and practical tools for proactive cybersecurity management. The results confirm that the system not only fulfills its stated objectives but also contributes valuable insights for further research and academic learning in the domain of cyber threat detection, automation, and cloud security operations.

Overall, the project outcome highlights a significant step toward developing intelligent, adaptive, and affordable cybersecurity frameworks suitable for both academic and industrial applications.

1.12 Report Organization

The report is organized into a sequence of chapters that collectively present the complete process of developing, implementing, and evaluating the project. Each chapter focuses on a specific phase of the system development life cycle, ensuring that the study flows logically from concept to conclusion. The organization of this report is as follows:

- **Chapter 1 - Introduction:** This chapter provides an overview of the entire project. It explains the background, need, and motivation behind the work. It also highlights the problem identified in existing systems, defines the project objectives, describes the methodology adopted, and discusses the expected outcomes. The introduction serves as the foundation for understanding the technical and practical importance of the project.

- **Chapter 2 – Requirement Specification:** This chapter focuses on identifying the hardware, software, and functional requirements needed to build the system. It defines the scope of the project in technical terms and provides a feasibility study to ensure that the proposed solution is practical, cost-effective, and implementable. It also discusses risk factors, assumptions, and system constraints that influence design decisions.

- **Chapter 3 – Methodology:** This chapter explains the design and development process followed in the project. It describes the overall system architecture, data flow, and working principles of various modules. It also elaborates on the tools and techniques used during the implementation phase and presents the workflow for integrating all modules into a unified system.

- **Chapter 4 – Implementation:** Chapter 4 provides a detailed account of the practical implementation of the project. It discusses the actual configuration of cloud resources, software components, and automation logic. Screenshots, flowcharts, and diagrams are included to give a visual understanding of how the system functions in real time. This section acts as a step-by-step guide for deploying and executing the solution.

- **Chapter 5 – Testing and Validation:** Once the system is implemented, it must be tested for accuracy, reliability, and performance. This chapter includes the testing strategy, validation process, and test case results. It explains how various scenarios were simulated to measure the system's response time and efficiency. Any errors, limitations, and observations are documented and analyzed in this part.

- **Chapter 6 – Snapshots and Outputs:** This chapter presents visual evidence of the system's successful operation. It includes screenshots of dashboards, monitoring interfaces, and automated response results. Graphical outputs, log reports, and analysis charts are also displayed here. These outputs help demonstrate the effectiveness of the implemented system and support the results discussed earlier.
- **Chapter 7 – Conclusion and Future Work:** The final chapter summarizes the key findings of the project and reflects on its success in meeting the stated objectives. It also suggests future improvements, such as incorporating artificial intelligence, predictive analytics, or advanced automation for enhanced security. This section ties together the technical and theoretical aspects of the study and provides direction for future research or real-world deployment.

Each chapter builds upon the previous one, maintaining continuity and clarity throughout the report. The structured format ensures that the reader can easily follow the development path—from the identification of a problem, through system design and testing, to the final results and conclusions.

1.13 Introduction Summary

This chapter established the foundation of the project by introducing the growing need for advanced cybersecurity measures and the motivation behind adopting a proactive, automated approach. It described the challenges of existing systems, detailed the objectives, scope, and methodology, and outlined the innovative features of the proposed model.

The study emphasizes that leveraging cloud technology and automation provides organizations with a sustainable, scalable, and intelligent defense framework capable of countering modern cyber threats. The following chapter will detail the system requirements and design specifications that support this implementation.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

A literature survey serves as the foundation of any research work, allowing the researcher to explore and analyze prior studies that have contributed to the development of a particular domain. In the field of cybersecurity, especially with the emergence of cloud computing, security operations centers (SOCs), and honeynet technologies, understanding existing research is essential for designing efficient, adaptive, and intelligent defense mechanisms. This chapter presents a comprehensive review of existing studies related to cyber defense models, honeynet systems, automated security frameworks, and AI-driven intrusion detection. The review highlights major research findings, methodologies, and tools used by various authors to address cyber threats. Each work is examined in terms of its strengths, limitations, and contribution to the evolution of proactive cybersecurity systems. The goal of this literature survey is to understand the current state of research and to identify gaps that justify the need for a cloud-based SOC integrated with honeynet intelligence — as implemented in this project using Microsoft Azure.

2.2 Review of Previous Studies

The review of previous studies provides a comprehensive understanding of how cybersecurity research has evolved over time, focusing on the progression from traditional network defense models to intelligent, cloud-integrated, and automated protection mechanisms. Researchers have consistently emphasized the importance of centralized monitoring, real-time analytics, and automation to counter the sophistication of modern cyber threats. This section critically examines past work across four primary domains: the evolution of Security Operations Centers (SOCs), the role of honeynet and deception technologies, the emergence of artificial intelligence and automation in cyber defense, and the growing reliance on cloud-based security frameworks. By analyzing the methodologies and conclusions presented in these studies, the review identifies major achievements, limitations, and open challenges that serve as a foundation for the present project.

2.2.1 Evolution of Security Operations Centers

The earliest SOC models concentrated on manual event monitoring, where human analysts interpreted system logs to identify intrusions. Early research highlighted the lack of scalability in such systems and the difficulty in correlating multiple events across distributed networks. As computing environments expanded, researchers proposed centralized SOC architectures to consolidate logs and improve visibility. Over time, the introduction of cloud-based monitoring platforms such as Azure Sentinel and AWS GuardDuty demonstrated that data could be aggregated and analyzed in real time with minimal hardware dependency. Modern SOCs are now built around integration and automation, allowing alerts, dashboards, and playbooks to function in unison. This evolution established the foundation for the Azure SOC Honeynet model used in this project, where the SOC becomes an intelligent, responsive command center.

2.2.2 Role of Honeynet and Deception Technologies

Parallel to SOC development, honeypots and honeynets emerged as proactive security tools designed to attract and study malicious activity. Early academic work demonstrated that deploying deceptive systems could safely capture attacker techniques, providing valuable intelligence for defensive strategies. Later research evolved this idea into honeynets—interconnected networks of honeypots that mimic real environments. Studies revealed that honeynet data enhanced intrusion detection accuracy and supported attack pattern classification. Recent works focus on integrating honeynet intelligence with SIEM (Security Information and Event Management) platforms to achieve richer context in event correlation. By automatically forwarding captured attack data into analytical engines, SOCs can visualize and understand emerging threats. The concept of deception is now central to proactive defense strategies and has influenced the design of hybrid honeynet-SOC systems like the one implemented in this project.

2.2.3 Automation and Artificial Intelligence in Cyber Defense

As attack volumes increased, researchers recognized that manual SOC operations could not keep pace. This led to the integration of automation and AI in threat detection and response. Machine learning algorithms began to analyze large volumes of network logs, identifying unusual patterns faster than human analysts. Early work focused on rule-based systems, but recent studies implemented predictive analytics and deep learning models capable of

recognizing zero-day behaviors. Automation frameworks, such as SOAR (Security Orchestration, Automation, and Response) tools, were introduced to automatically execute mitigation steps when specific triggers occur. This shift to self-learning, adaptive systems transformed the SOC into a proactive defense hub capable of detecting, classifying, and responding to incidents autonomously. The present project adopts this approach by using Azure Logic Apps and Sentinel Playbooks to execute automated responses based on predefined logic workflows.

2.2.4 Cloud-Based Security and Scalability

Recent research has focused heavily on cloud security frameworks that combine elasticity, automation, and data analytics. Cloud providers such as Microsoft Azure, Google Cloud, and Amazon Web Services offer built-in security services, allowing SOCs to scale with the infrastructure they protect. Studies have shown that cloud-native SOCs improve accessibility, reduce deployment time, and enable centralized monitoring of globally distributed assets. At the same time, challenges such as data privacy, compliance, and multi-tenant security have become prominent research topics. Researchers emphasize the need for secure log management, encryption, and real-time correlation to maintain trust in cloud-based defense systems. This line of work provides direct motivation for the current project's design, which leverages Azure's integrated tools—Sentinel, Log Analytics, and Monitor—to implement a fully functional, scalable, and automated security monitoring environment.

2.2.5 Key Insights from the Review

From the reviewed literature, it is evident that cybersecurity research has transitioned from manual and reactive monitoring to automated, data-driven, and proactive defense systems. Each thematic area—SOC evolution, deception technologies, automation, and cloud scalability—contributes to building a comprehensive defense architecture. While prior studies achieved notable progress, they often remained limited to individual aspects, such as automation or honeynet deployment. The Azure SOC Honeynet project uniquely combines all these dimensions into a single integrated solution, demonstrating how a cloud-based SOC can collect, analyze, and respond to threats intelligently and efficiently.

2.3 Summary of Reviewed Literature

Author & Year	Method / Technology Used	Key Contribution	Limitations / Remarks
Spitzner (2019)	Honeypot & Deception Systems	Established honeynets and proactive monitoring.	Focused on early, non-cloud setups; recommended high-interaction honeypots for advanced study.
Al-Hadhrami & Banerjee (2020)	Cloud-based IDS Framework	Introduced distributed IDS for real-time detection in cloud.	Encountered latency and scalability challenges during peak loads.
Patel & Rao (2020)	Hybrid ML + Rule-Based Detection	Enhanced anomaly detection accuracy and reduced false positives.	Requires large, labeled datasets for effective training.
Kumar et al. (2021)	Azure SOC with Logic Apps	Demonstrated automated incident response with native cloud tools.	Limited AI correlation and dependency on cloud vendor features.
Hernandez & Singh (2021)	Honeynet + SIEM Integration	Improved visibility and event reconstruction via honeynet data.	Manual rule tuning needed for noisy datasets.
Kaur & Mehta (2022)	AI-driven SOC Automation	Implemented predictive analytics and andreduced false alarms.	High computational cost and model maintenance requirements.

Zhang et al. (2022)	Federated SOC Architecture	Enabled secure data sharing among multiple SOCs through a federated intelligence model.	Complex setup, privacy concerns, and data standardization issues.
Johnson & Ahmed (2023)	Cloud-Native SOC Deployment	Proved cost-effective, centralized security management.	Requires stronger encryption in multi-tenant scenarios.
Rahman & Thomas (2023)	Hybrid Honeynet Architecture	Captured real attack data and automated threat analysis.	Detection accuracy impacted by honeypot realism.
Chaudhary & Nair (2024)	ML-enhanced SOC Automation	Used AI models to predict attacks and auto-initiate defense workflows.	Complex integration in heterogeneous environments.
Sharma et al. (2024)	Adaptive Cloud SOC Framework	Showed benefits of ML-based automation and alert triaging.	Needs continuous model updates to remain accurate.
Lee & Park (2024)	Threat Intelligence Enrichment	Demonstrated enrichment of SIEM events with external TI feeds for better attribution.	Depends on the quality and timeliness of external feeds.

Table 2.3 – Summary of Reviewed Literature

2.4 Summary of Literature Survey

The literature reviewed in this chapter reveals a significant evolution in cybersecurity approaches, from static defense systems to dynamic, data-driven frameworks. The combination of automation, cloud scalability, and honeynet intelligence is transforming how security operations are managed. While previous research laid the foundation for SOC design and intrusion analysis, many implementations still rely on partial integration or manual processes. This project, Azure SOC Honeynet: Proactive Cyber Defense in Action, builds upon these foundations by developing a holistic system that utilizes Azure Sentinel, Log Analytics, and Logic Apps to deliver real-time, automated, and scalable cyber defense. By addressing the limitations identified in earlier studies — such as limited automation, latency issues, and lack of integrated intelligence — this project demonstrates how cloud-based SOCs can deliver proactive, intelligent, and cost-effective cybersecurity solutions for modern infrastructures.

CHAPTER-3

SYSTEM ANALYSIS AND REQUIREMENT SPECIFICATION

3.1 System Analysis

System analysis is one of the most essential phases in any software development or deployment project. It involves the process of breaking down a complex system into smaller, understandable components to identify how each part contributes to the system's functionality. In the context of the Azure SOC Honeynet, system analysis helps define the overall structure of security monitoring, data collection, threat detection, and response automation within a cloud environment. The purpose of system analysis in this project is to ensure that the proposed system meets both technical and operational requirements while maintaining high efficiency, reliability, and security. It also aims to identify existing problems in conventional SOC frameworks, analyze system performance needs, and translate user expectations into specific, measurable system requirements.

Traditional SOC environments are often reactive in nature. They depend on pre-defined rules or signatures to detect threats. Such systems can miss new or evolving attacks, resulting in delayed response or data breaches. Moreover, the absence of deception-based mechanisms such as honeynets limits visibility into attacker behavior. This project's analysis addresses these gaps by integrating honeynet-based data sources into Azure Sentinel, which can correlate events in real time and automate mitigation actions through playbooks.

The system analysis process also focuses on data behavior — including how logs are collected, processed, stored, and visualized. Azure Log Analytics acts as the data repository, enabling continuous ingestion from multiple honeypots and virtual machines. Each log entry is parsed and analyzed for potential anomalies. The use of Kusto Query Language (KQL) ensures flexible and efficient querying for pattern recognition and incident detection. Through such deep analysis, this project establishes a framework that supports proactive defense mechanisms against cyber threats. System analysis not only studies the functional needs of the project but also considers performance constraints, scalability challenges, and user accessibility. It ensures that every module — from log generation to visualization — is logically connected and efficiently designed. The outcome of this analysis defines the foundation for design, implementation, and evaluation in later chapters.

3.2 Requirement Specification

The Requirement Specification phase is one of the most crucial components of the system development process. It acts as the foundation for all subsequent stages such as system design, development, and implementation. This phase aims to document the complete set of system requirements in an organized and detailed manner. The primary goal of this section is to ensure that the developers, testers, and stakeholders have a clear understanding of the system's objectives, scope, and functional expectations.

A well-defined requirement specification minimizes the possibility of project failure by ensuring that the system meets all expected functionalities, adheres to quality standards, and aligns with organizational goals. It provides a bridge between the analytical study and the design stage, translating abstract problem statements into actionable technical requirements.

In this project, the Azure SOC Honeynet functions as a comprehensive cyber threat detection and monitoring system. Hence, the requirement specification emphasizes the accurate collection, processing, and visualization of security data, along with automated threat response and incident handling mechanisms. The specification also defines user roles, data constraints, communication protocols, and the interaction between system components.

This section categorizes requirements into functional, non-functional, hardware, and software specifications, followed by constraints and assumptions that define the operational limits of the system.

3.2.1 Functional Requirements

Functional requirements specify the actual operations and behavior of the system. These define how data will flow, how the system reacts to events, and how users interact with different modules. For the Azure SOC Honeynet project, the following functional requirements are defined: Collection and centralization of security logs from multiple honeypots and Azure virtual machines.

- Continuous monitoring and analysis of network activities for anomalies and suspicious behavior.
- Real-time detection and correlation of threats using Azure Sentinel's analytic rules.
- Alert generation based on predefined severity levels and alert categories.
- Automation of responses through Azure Logic Apps, such as blocking malicious IPs or notifying SOC analysts.

- Dashboard creation for visual representation of ongoing security events, attack trends, and threat intelligence data.
- Integration of honeynet and SOC systems for data enrichment and proactive defense.
- Secure access control mechanisms through Azure Active Directory with role-based permissions.
- Audit logging for accountability and traceability of user actions within the SOC environment.
- Periodic generation of incident and performance reports for analysis and compliance purposes.

Category	Component / Parameter	Description & Purpose
Functional Modules	Honeynet Deployment	Virtual Machines configured as decoy systems
	Data Collection Layer	Azure Monitor+Log Analytics Workspace
	SOC Analysis Layer	Azure Sentinel correlation engine
	Response Automation	Logic Apps / Playbooks
	Reporting and Visualization	Workbooks / Power BI reports

Table 3.2.1: Functional requirement modules

3.2.2 Non-Functional Requirements

Non-functional requirements describe the quality attributes, performance criteria, and constraints that the system must satisfy. They determine how well the system performs under different conditions and ensure operational excellence.

- Performance: The system should process logs in near real-time with less than 5-second delay in data visualization.

- Scalability: The solution should support the addition of new honeypots or data sources without reconfiguration.
- Reliability: The system should ensure 99.9% uptime through redundancy and load balancing in Azure.
- Availability: Services should be accessible 24/7 with minimum maintenance downtime.
- Security: All communications must be encrypted using HTTPS, and logs must be stored securely using Azure Storage.
- Maintainability: The architecture should allow modification of analytic rules and playbooks with minimal effort.
- Usability: Dashboards should present data clearly and allow customization according to user preferences.
- Interoperability: The system must integrate easily with other third-party tools and APIs for data enrichment.
- Compliance: System must adhere to general data protection and security standards such as ISO 27001 principles.

Category	Component / Parameter	Description & Purpose
Non-Functional Aspects	Scalability	Supports horizontal scaling of VMs and log sources
	Availability	Multi-region deployment and redundancy
	Security & Compliance	TLS Encryption, Azure Key Vault, RBAC
	Maintainability	Modular playbooks and scripts

Table 3.2.2: Non-Functional requirement modules

3.2.3 Hardware and Software Requirements

Implementing a cloud-based honeynet system requires both local (on-premise) and cloud-side resources. The hardware and software configuration must support virtualization, network isolation, and continuous connectivity with Azure services.

Hardware Configuration:

- **Processor:** Intel i5 or higher / AMD Ryzen 5 or equivalent, supporting virtualization technology.
- **Memory (RAM):** Minimum 8 GB, preferably 16 GB for multitasking and running multiple VMs.
- **Storage:** At least 100 GB of free space to store logs, datasets, and local backups.
- **Internet Connection:** Minimum 10 Mbps high-speed broadband to maintain constant connectivity with Azure services.
- **System Type:** 64-bit architecture capable of running both Linux and Windows virtual machines.

Component	Minimum Specification	Description / Purpose
Processor	Intel Core i5 / i7 or higher	The processor handles concurrent tasks such as Azure operations, log ingestion, and virtual machine execution. A multi-core processor ensures faster computation and smooth functioning of SOC components
RAM	Minimum 8 GB (Recommended 16 GB)	Adequate memory is required to run multiple services simultaneously. It ensures lag-free operation while managing

		dashboards, executing scripts, and monitoring honeypots.
Hard Disk Storage	Minimum 100 GB (SSD preferred)	Provides sufficient space to store log files, reports, scripts, and configurations. SSD improves data access speed and enhances overall system responsiveness.
Network Connectivity	Broadband Internet (10 Mbps or higher)	Required for stable communication between local systems and Azure Cloud resources. A high-speed network ensures uninterrupted data ingestion and analytics.
Display	1920 × 1080 pixels (Full HD)	Ensures clear visualization of Sentinel dashboards, analytics charts, and reports for SOC analysis.
Peripherals	Keyboard,Mouse, External Drive	Required for configuration, maintenance, and report storage during development and testing.

Table 3.2.3: Hardware configuration requirements

Software Configuration:

- **Operating Systems:** Windows 11 Professional (for administrative use) and Ubuntu 22.04 (for honeypot deployment).
- **Cloud Platform:** Microsoft Azure with active subscription (Free Tier or Pay-As-You-Go).
- **Security Tools:** Azure Sentinel, Azure Monitor, Security Center, Key Vault, and Log Analytics Workspace.
- **Development and Querying:** Kusto Query Language (KQL) for data analysis, PowerShell for automation, and Python for script-based log extraction or alerting.
- **Visualization Tools:** Azure Workbooks and optional Power BI integration for dashboards and graphical reporting.

Software / Tool	Version / Platform	Purpose / Description
Operating System	Windows 10 / 11, or Ubuntu 20.04	Acts as the base environment for developing and testing SOC components and managing virtual machines.
Microsoft Azure	Cloud Platform	Primary cloud platform used for hosting the SOC environment, creating virtual machines, and integrating services like Sentinel and Logic Apps.
Azure Sentinel	Cloud-native SIEM (Built into Azure)	Enables real-time monitoring, incident detection, and response automation by correlating security events and logs
Azure Log Analytics	Included with Azure Monitor	Collects and analyzes log data from honeypots and

		virtual machines using Kusto Query Language (KQL).
Kusto Query Language (KQL)	Azure-native Query Language	Used for executing queries and detecting anomalies within large datasets. Essential for Sentinel analytics and dashboard creation.
Azure CLI	Latest Version	Command-line tool to manage Azure resources, automate deployments, and configure services directly from the terminal.
Web Browser	Microsoft Edge / Google Chrome	Used to access Azure Portal, dashboards, and reports in a secure manner.

Table 3.2.4 : Software configuration requirements

3.2.5 System Constraints and Assumptions

- The system depends on continuous internet connectivity for Azure service communication.
- Subscription costs may vary depending on the data volume and retention period.
- Users must have valid Azure AD credentials for authentication and access.
- Detection quality depends on the accuracy and frequency of analytic rule updates.
- The honeynet should operate in an isolated environment to avoid production impact.
- Collected logs must comply with security and data retention policies.

3.3 Feasibility Study

The feasibility study ensures that the project is realistic, cost-effective, and technically sound before implementation. It assesses whether the proposed system can be successfully developed, deployed, and maintained under existing conditions.

3.3.1 Technical Feasibility

The system is technically feasible as it relies entirely on cloud-based Azure services, which offer integrated solutions for data collection, processing, and automation. Sentinel, Log Analytics, and Logic Apps are fully compatible, eliminating the need for external dependencies or third-party integrations.

3.3.2 Operational Feasibility

Operational feasibility examines how well the proposed system will function within the organization's environment. The Azure SOC Honeynet is user-friendly, with a visual dashboard and intuitive alert management system. Automation reduces the workload on analysts, enhancing operational efficiency.

3.3.3 Economic Feasibility

The project is economically feasible as it follows a pay-as-you-go billing model. It allows the scaling of resources according to demand, ensuring minimal upfront investment and predictable operational costs.

3.4 Summary

This chapter described the detailed system analysis and requirement specification of the Cyber Security project. It outlined functional and non-functional requirements, feasibility studies, and design considerations. These analyses ensure that the system is secure, efficient, and capable of providing continuous monitoring and automated threat response. The next chapter focuses on system design and implementation aspects.

CHAPTER-4

SYTEM DESIGN AND METHODOLOGY

4.1 System overview

- The proposed system is designed to develop an intelligent and automated threat monitoring framework by integrating Honeynet technology with a Security Operations Center (SOC) using Microsoft Azure cloud services.
- The main goal of this design is to monitor, analyze, and respond to cyber threats in real time by combining deception-based data collection from honeypots with centralized monitoring and visualization through Azure SOC.
- The overall system design consists of multiple integrated components that work together to enhance threat visibility and improve the efficiency of security operations. The Honeynet serves as a decoy network that attracts malicious traffic and captures detailed information about intrusion attempts. These logs are then forwarded to the SOC, where Azure Sentinel analyzes the data and generates security alerts based on correlation rules and analytics models.
- Through automation, the system ensures faster alerting and predefined incident responses. Logic Apps and playbooks handle repetitive tasks like blocking malicious IP addresses or notifying security teams automatically. Dashboards and workbooks within Azure provide real-time visualization of security events, allowing administrators to monitor ongoing activities and analyze historical trends.
- The proposed system emphasizes centralized visibility, scalability, and real-time monitoring. Each module communicates securely through cloud-based APIs and analytics pipelines, ensuring efficient coordination between detection, analysis, and response stages. By combining the proactive deception capabilities of a Honeynet with the analytical strength of Azure SOC, the system provides an effective end-to-end monitoring solution.
- This design ultimately strengthens the organization's ability to detect and mitigate cyber threats before they cause significant damage, ensuring a robust and resilient cybersecurity infrastructure.

  The following sections describe the materials and tools used, the system architecture, and the implementation of integration and automation in greater detail.

4.2 Design/Plan

The project involves creating a cybersecurity lab that integrates a honeynet with Security Operations Center (SOC) operations within the Microsoft Azure cloud environment. The design includes the following components and Figure 4.2 Block Diagram

- Honeynet Setup: Deploy virtual machines and SQL databases configured as honeypots to attract and analyze cyber threats.

- Azure Integration: Utilize Azure AD, Blob Storage, Key Vault, and Activity Logs for managing and securing the environment.

- SOC Operations: Implement Azure Sentinel (SIEM) for real-time monitoring, incident detection, and response. The SOC will analyze data collected from the honeynet to identify threats and generate alerts.

- Log Analytics Workspace: Centralize log data from various sources to support threat detection and analysis.

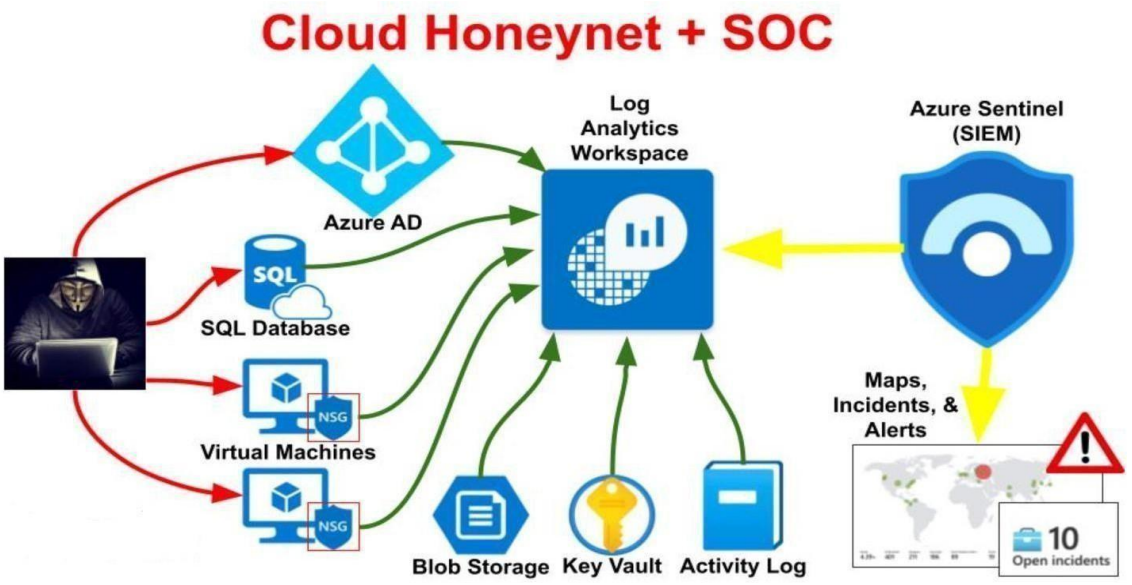

Figure 4.2: Overall System Architecture of the Proposed Azure SOC Honeynet Model

4.3 System Architecture

The proposed system architecture is designed to establish a secure, intelligent, and automated framework for continuous monitoring of cyber threats. The architecture integrates Honeynet technology with a Security Operations Center (SOC) using Microsoft Azure cloud services. This combination enhances the detection, analysis, and response capabilities by leveraging data collected from simulated attack environments (honeypots) and centralizing it within a scalable SOC platform.

The architecture consists of four major layers — Data Capture Layer, Data Processing Layer, Analysis Layer, and Visualization Layer. Each layer performs specific tasks that contribute to the overall objective of identifying and mitigating threats in real time.

4.3.1 Honeynet Design

Honeynets are advanced forms of honeypots, consisting of multiple honeypots within a network. They are designed to attract cyber attackers, allowing researchers and security professionals to study attack patterns and methodologies. Honeynets provide a controlled environment to monitor malicious activities without risking the integrity of the actual network.

A study by Spitzner (2019) emphasizes the importance of honeynets in understanding the tactics, techniques, and procedures (TTPs) used by attackers.

The study highlights the effectiveness of honeynets in detecting novel attacks and gathering intelligence that can be used to enhance security measures. Additionally, honeynets have been instrumental in detecting and analyzing zero-day exploits, which are attacks that exploit previously unknown vulnerabilities.

Recent advancements in honeynet technologies include the integration of machine learning and artificial intelligence (AI). For instance, the work of Zhang et al. (2021) explores the use of AI-driven honeynets to automate the detection and analysis of cyber threats.

The study demonstrates how machine learning algorithms can enhance the capabilities of honeynets by identifying patterns in attack data and predicting future threats.

4.3.2 SOC (Security Operations Center) Structure

SOCs are centralized units within organizations responsible for monitoring, detecting, and responding to security incidents. They employ a combination of people, processes, and technologies to protect the organization's assets. SOCs use various tools, including Security Information and Event Management (SIEM) systems, to collect and analyze security data from across the network.

The SOC structure includes:

- Data Source Integration: Collects logs from honeypots, firewalls, and virtual machines.
- Log Analytics Workspace: Central storage for all collected data.
- Detection Rules: Configured in Azure Sentinel to identify potential threats.
- Logic Apps / Playbooks: Used to automate responses such as sending email alerts or blocking IPs.
- Dashboard and Workbooks: Provides real-time visual representation of network health and active incidents.

4.3.3 Integration of Honeynet and SOC

The integration of honeynets within SOC operations provides a comprehensive approach to cybersecurity. Honeynets can feed valuable data into the SOC's SIEM system, enhancing the ability to detect and respond to threats. The combination of these technologies allows for a proactive security posture, where potential threats are identified and mitigated before they can cause significant damage.

The integration between the Honeynet and SOC forms the backbone of the proposed cybersecurity monitoring system. Once malicious traffic is captured in the Honeynet, it is sent to Azure through secure channels using Azure Monitor agents and Log Analytics APIs. These logs are then analyzed in the SOC to identify attack trends and generate security alerts.

This integration enables the system to:

- Maintain real-time synchronization between threat data collection and analysis.
- Automate alert generation and incident response.
- Create a feedback loop for improving detection rules.
- Allow centralized dashboard visualization of both raw and processed threat data.

By combining deception-based monitoring (Honeynet) with centralized analytics (SOC), the integrated framework ensures better visibility into emerging threats, allowing administrators to respond efficiently and proactively.

4.4 Materials/Tools

- **Microsoft Azure:** A cloud platform offering a range of services including virtual machines, databases, and security tools for deploying, managing, and scaling applications.

- **Virtual Machines:** Configurable cloud-based servers used to simulate different systems and services, such as honeypots, to attract and analyze cyber threats.

- **SQL Database:** A relational database service designed to handle SQL queries and operations, useful for studying SQL injections and other database-related attacks within a honeynet.

- **Azure AD:** A cloud-based identity and access management service that helps control and secure user access to applications and resources.

- **Blob Storage**: A service for storing large amounts of unstructured data, such as logs and backup files, with scalable and secure access.

- **Key Vault:** A tool for securely managing and accessing secrets, encryption keys, and certificates used to protect sensitive data and applications.

- **Activity Log:** Provides detailed records of operations and events within the Azure environment, useful for auditing and troubleshooting.

- **Azure Sentinel**: A cloud-native SIEM tool that helps detect, investigate, and respond to security threats by analyzing data across the enterprise.

- **Log Analytics Workspace**: A centralized platform for collecting, analyzing, and visualizing log and performance data from various sources in Azure.

- **KQL (Kusto Query Language):** A powerful query language used in Azure Log Analytics and Azure Sentinel for querying and analyzing large volumes of log and telemetry data efficiently. It allows users to extract insights and generate reports from structured and unstructured data.

4.5 System Flowchart

The system flowchart represents the step-by-step process followed in implementing the proposed threat monitoring system. It explains how various Azure services and honeypot configurations interact to detect, analyze, and respond to potential security threats.

The process begins by setting up the Azure environment, creating a resource group, and deploying virtual machines that serve as honeypots. These honeypots are configured to attract and log malicious activities, simulating real-world attack environments.

Next, Azure Active Directory (Azure AD) is integrated to provide identity management and secure authentication. The collected logs are forwarded to the **Log Analytics Workspace**, where they are stored and processed. Azure Sentinel, acting as the **Security Operations Center (SOC)**, analyzes the data to detect potential intrusions based on predefined detection rules.

Once the analysis is complete, **alerts are generated** automatically for any identified threats. These alerts trigger **automated responses** through configured Logic Apps, which can block malicious IPs, send notifications, or update dashboards. Finally, **security reports** are generated that summarize system activity and highlight detected threats for administrative review.

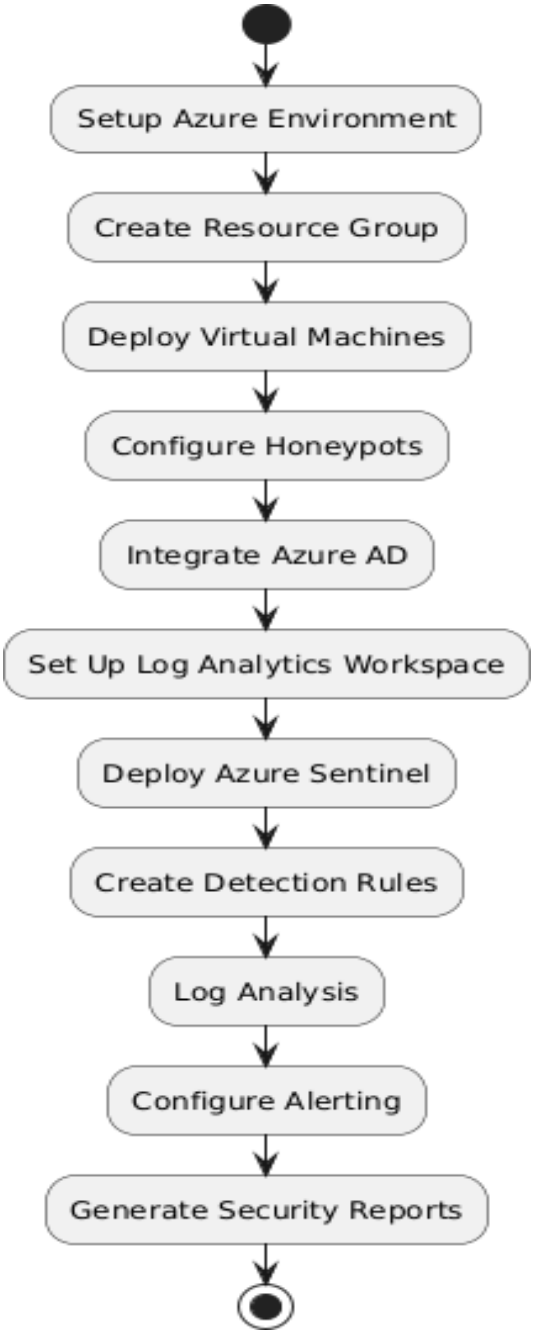

Figure 4.5: Flowchart of System Methodology for Threat Monitoring using Honeynet and SOC Integration

- **Setup Azure Environment:** Initialize the cloud infrastructure for hosting the entire monitoring solution.
- **Create Resource Group:** Organize all Azure resources (VMs, networks, logs) into one manageable group.
- **Deploy Virtual Machines:** Create isolated environments that act as honeypots to attract malicious traffic.
- **Configure Honeypots:** Install and configure vulnerable systems designed to log attacker behavior.
- **Integrate Azure AD:** Provide secure authentication and user management for all system resources.
- **Set Up Log Analytics Workspace:** Centralize the storage of all collected logs from honeypots and other sources.
- **Deploy Azure Sentinel:** Activate the SOC module responsible for monitoring and analyzing log data.
- **Create Detection Rules:** Define conditions that trigger alerts when suspicious activities are detected.
- **Log Analysis:** Examine log entries for unusual patterns or malicious activity indicators.
- **Configure Alerting:** Automate responses to incidents using Logic Apps or notification systems.
- **Generate Security Reports:** Summarize detected threats and system activities for further decision-making.

4.6 Procedure

The procedure for setting up and configuring the cybersecurity lab with a honeynet and SOC operations in Microsoft Azure involves several key steps. Here's a detailed explanation of each step:

4.6.1 Setup Azure Environment

Create an Azure Account: If you don't already have one, sign up for an Azure account as per Figure 4.6.1 Account setup. Follow these steps to get started:

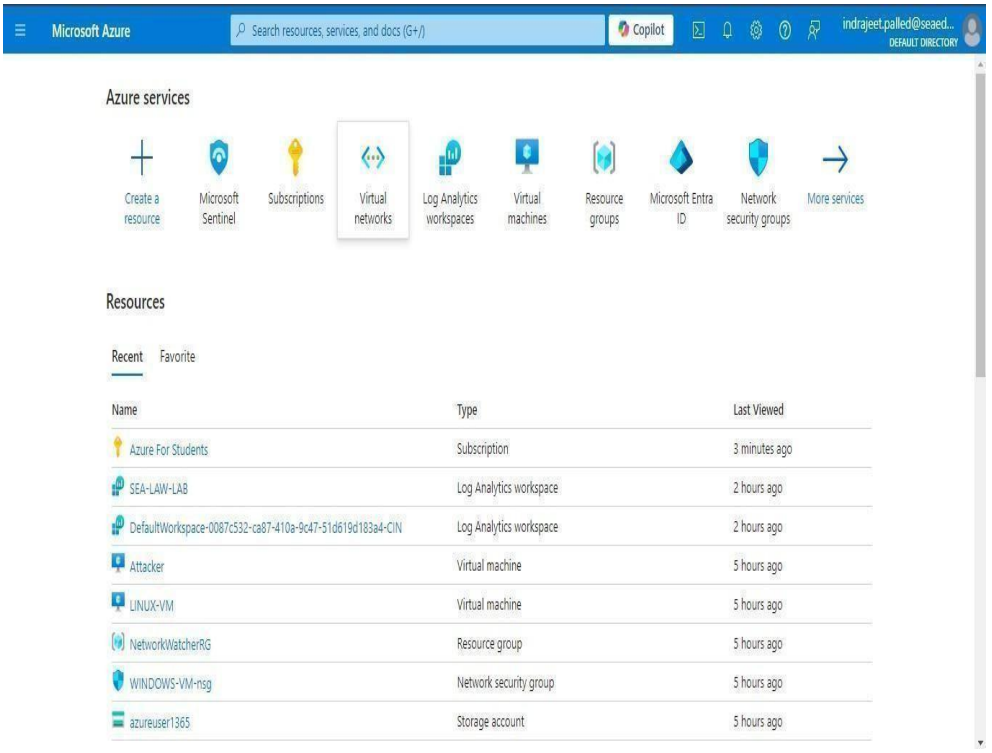

Figure 4.6.1: Account Setup Process in Microsoft Azure

.

Follow these steps to get started:

1. **Visit the Azure Portal**
   - Go to the Azure website.
   - Click on the "Sign In" button at the top right corner.

2. **Sign In or Create a Microsoft Account**
   - If you already have a Microsoft account, sign in with your credentials.
   - If you don't have an account, create one by clicking on "Create one!" and follow the prompts to set up your account.

3. **Setup to Pay-As-You-Go**
   - If you already have Microsoft account or want to start directly with a Pay-As-You-Go account, go to the Azure Pay-As-You-Go page.
   - Follow the on-screen instructions to complete the setup.

4. Provide Personal and Payment Information

- Provide your personal information, including your name, address, and phone number.
- Enter your payment details. Azure requires a valid credit card or debit card. You won't be charged until you exceed the free tier limits (if starting with a free account) or start using resources if you set up Pay-As-You-Go directly.

5. Accept the Agreement

- Review the Microsoft Azure agreement and the privacy statement.
- Check the box to accept the terms and conditions.

6. Complete the Setup

- Click on the "Sign up" button to complete the setup process.
- Azure will validate your payment information and activate your account.

7. Access the Azure Portal

- Once your account is set up, you can access the Azure portal by going to portal.azure.com.
- Use the portal to manage and deploy Azure services.

4.6.2 Set Up Resource Group: Create a new resource group in Azure to organize and manage all related resources as per Figure 4 Resource Group.

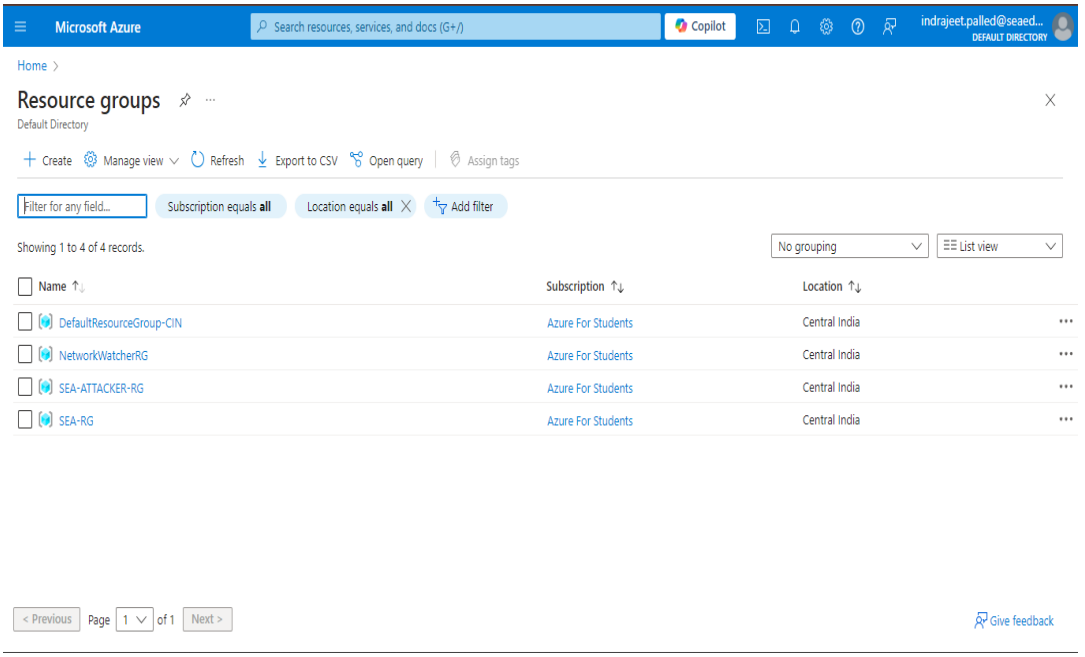

Figure 4.6.2: Creation of Resource Group in Azure Portal

4.6.3 Deploy Virtual Machines as Honeypots

- **Create Virtual Machines**: Deploy multiple virtual machines (VMs) in Azure, configured with different operating systems and services (e.g., Windows, Linux).
- **Configure Honeypots**: Install and configure honeypot software on each VM to simulate real systems and services that can attract attackers.
- **Network Security Groups (NSGs)**: Set up NSGs to control inbound and outbound traffic to the VMs, ensuring only legitimate traffic is allowed. As per figure 5 Its shows virtual machines and NSG.

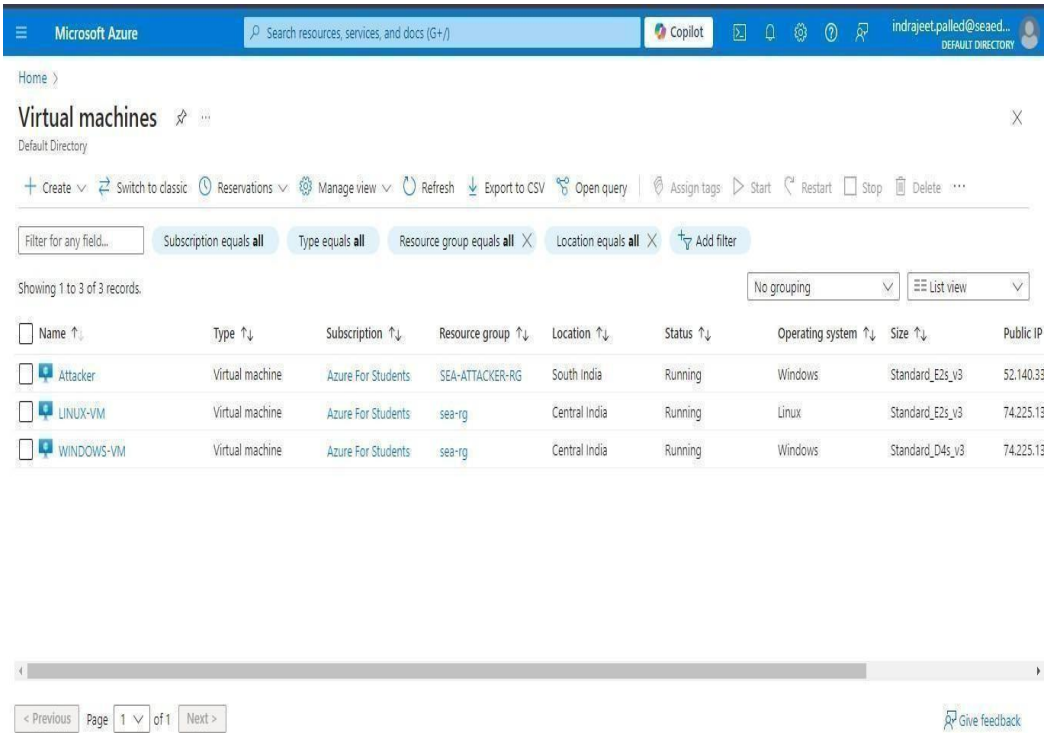

Figure 4.6.3: Deployment of Virtual Machines and NSG Configuration in Azure

4.6.4 Configure SQL Database

- **Deploy SQL Database**: Create and configure an Azure SQL Database to simulate a production database environment.
- **Honeypot Configuration**: Set up the database with dummy data and configure it to act as a honeypot for SQL injection and other database attacks.

4.6.5 Integrate Azure AD

- **Set Up Azure AD**: Configure Azure Active Directory (AD) for identity and access management, ensuring secure access to resources.
- **User and Group Management**: Create users and groups in Azure AD, assign roles and permissions as needed.

4.6.6 Set Up Log Analytics Workspace

- **Create Log Analytics Workspace**: In Azure, create a Log Analytics Workspace to collect and analyze log data from various sources.
- **Configure Data Sources**: Connect your VMs, SQL Database, and other resources to the Log Analytics Workspace for centralized logging.

4.6.7 Implement Azure Sentinel

- **Deploy Azure Sentinel**: Set up Azure Sentinel, a cloud-native SIEM tool, to monitor security events and incidents.
- **Connect Data Sources**: Integrate data sources like Azure AD, virtual machines, SQL Database, and others with Azure Sentinel for comprehensive monitoring.
- **Create Detection Rules**: Configure detection rules in Azure Sentinel to identify suspicious activities and potential threats based on collected log data.
- As per figure 4.7.7 its Microsoft sentinel Overview its stores the incidents.

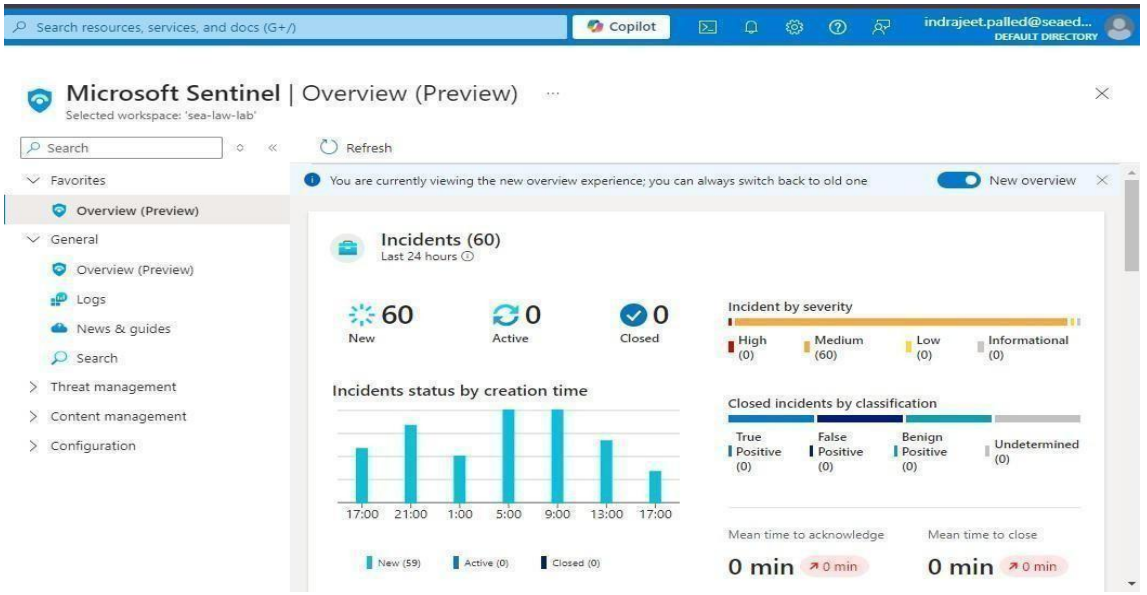

Figure 4.6.7: Azure Sentinel Dashboard Showing Incidents and Alerts

4.6.8 Data Collection and Analysis

- **Enable Diagnostic Settings**: Configure diagnostic settings for each Azure resource to send logs and metrics to the Log Analytics Workspace.
- **Analyze Logs**: Use Azure Sentinel to analyze logs, identify patterns, and generate alerts for potential threats.

4.6.9 Monitoring and Maintenance

- **Continuous Monitoring**: Set up continuous monitoring of the Azure environment using Azure Sentinel to detect and respond to threats in real-time.
- **Regular Updates and Patching**: Ensure all VMs, databases, and other resources are regularly updated and patched to protect against known vulnerabilities.

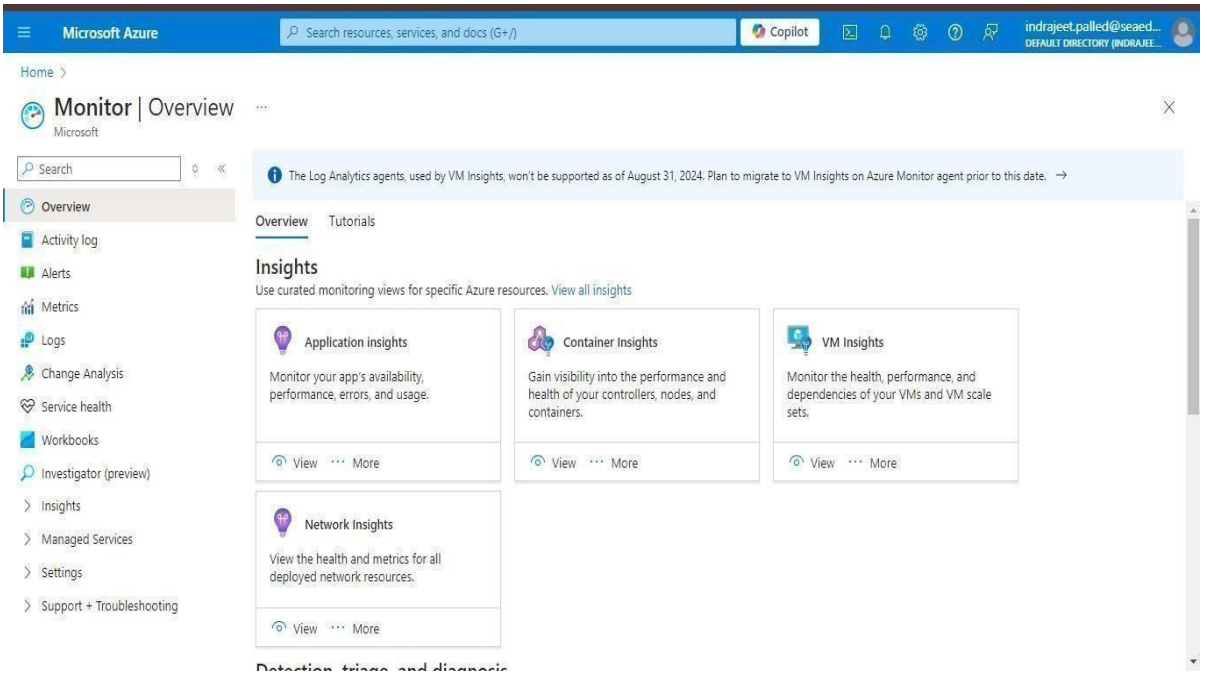

Figure 4.6.9: Continuous Monitoring and Maintenance Workflow in Azure Sentinel

4.7 SOC Infrastructure and Architecture

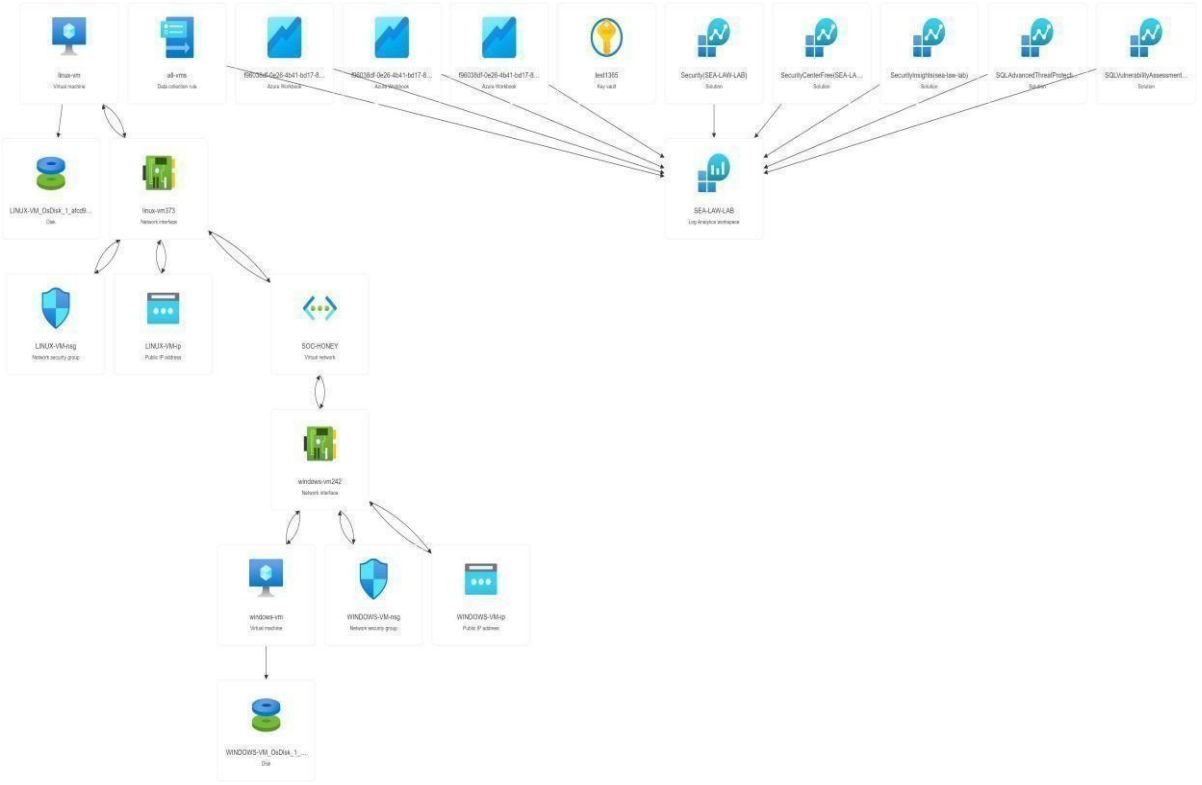

Figure 4.7: Architectural Review of SOC Infrastructure

Figure 4.7 illustrates the architectural review of a Security Operations Center (SOC) infrastructure setup in Azure. This diagram showcases the components involved and their interactions. Let's break down the components and explain the connections after the setup:

4.7.1 Components and Their Roles:

SEA-LAW-LAB (Central Node):

- Acts as the central hub for collecting and analyzing data

Sources on the Right:

- Source-MSSQL-LAW-LAB: Likely a SQL database source.
- Source-SysPerfTest-SEA-LAB: Source related to system performance testing.

- Source WinSys-NoiseSeq-Lab: Source for Windows system noise or log data.
- SQL-Advanced Threat Protection: Provides advanced threat protection for SQL databases.
- SQL-Virtual Entity Assessment: Performs assessments on virtual entities within SQL databases.

Virtual Machines and Containers on the Left:

- LINUX-VM-Docker: Linux virtual machine running Docker.
- Reserve2019: Another node, potentially a reserved instance or a specific environment for the year 2019.
- LINUX-VMs (several): Multiple Linux virtual machines for various purposes.
- windows-vm: Windows virtual machine for general purposes.
- WINDOWS-VM-log: Windows virtual machine specifically for logging.
- WINDOWS-VM-log2: Another Windows virtual machine dedicated to logging.
- WINDOWS-VM-Disk: Windows virtual machine with disk storage for logs.

4.7.2 Connections and Data Flow:

Connections from Virtual Machines to SEA-LAW-LAB:

- LINUX-VM-Docker to SEA-LAW-LAB Docker container on a Linux VM sending data to the central node.
- Reserve2019 to SEA-LAW-LAB Data from the Reserve 2019 node flows into the central node.
- LINUX-VMs to windows-vm: Data or processes from Linux VMs interacting with a Windows VM.
- windows-vm to WINDOWS-VM-log and WINDOWS-VM-log2: Data from the Windows VM is sent to two logging-specific VMs.
- WINDOWS-VM-log and WINDOWS-VM-log2 to SEA-LAW-LAB Logging data from these VMs is forwarded to the central node.

- windows-vm to WINDOWS-VM-Disk: Interaction between the general Windows VM and a disk storage VM.

Connections from Sources to SEA-LAW-LAB:

- Various sources (SQL databases, performance test data, system noise data) are connected directly to the central node (SEA-LAW-LAB).
- These connections indicate that the central node collects data from multiple sources for analysis and monitoring.

4.7.3 Summary of the Setup:

- The central node (SEA-LAW-LAB) is the heart of the SOC infrastructure, receiving data from various virtual machines, Docker containers, and external sources.
- Virtual machines and Docker containers run different workloads and send their data to the central node for logging and analysis.
- Specialized VMs for logging and disk storage help organize and manage log data before sending it to the central node
- External sources, including SQL databases and performance test data, provide additional information to the central node, enhancing the SOC's monitoring and analytical capabilities.

Figure effectively depicts the SOC infrastructure in Azure, highlighting the central role of SEA-LAW-LAB in aggregating and analyzing security data. The architecture integrates various virtual machines, containers, and data sources to provide a robust and scalable SOC environment. This setup ensures efficient data collection, log management, and threat detection capabilities within the SOC infrastructure.

CHAPTER-5

IMPLEMENTATION

5.1 Introduction

This chapter presents the implementation phase of the cybersecurity monitoring system using Microsoft Azure Sentinel integrated with Honeynet and SOC (Security Operations Center) frameworks.

The purpose of this implementation is to configure the cloud environment, collect real-time security logs, and analyze potential threats using Kusto Query Language (KQL). The developed system provides insights into user behavior, network anomalies, and resource utilization patterns through interactive dashboards and alert-based analytics.

5.2 Implementation Overview

The implementation begins with configuring the Azure environment, integrating multiple log sources such as Azure Activity Logs, Signin Logs, Network Analytics, and Performance Metrics, and then writing custom KQL queries to analyze and visualize this data.

The KQL scripts are used to detect:

- Failed logins and unauthorized access
- Suspicious IP address activity
- High CPU utilization
- Unusual data transfer patterns
- Privileged account misuse
- Malware infections and network anomalies

These implementations are executed within the Azure Sentinel workspace, and results are visualized through security dashboards and alerting systems.

5.3 KQL Query Implementations

KQL (Kusto Query Language) is the core query language used in Azure Sentinel to retrieve and analyze log data.Each query below targets a specific use case to detect and monitor possible security threats.

5.3.1 Basic Log Query

Purpose: Retrieve all events from the Azure Activity log for the past day.

Kql

code

```
AzureActivity
| where TimeGenerated >= ago(1d)
| order by TimeGenerated desc
```

- AzureActivity: The data source from which you are retrieving logs.
- where TimeGenerated >= ago(1d): Filters logs to include only those generated in the last 24 hours.
- order by TimeGenerated desc: Orders the results by the time the event was generated, from most recent to oldest.

5.3.2 Failed Logins

Purpose: Identify failed login attempts, which might indicate potential brute force attacks.

Kql

code

```
SigninLogs
| where ResultType == "50126"
| summarize Count = count() by UserPrincipalName, IPAddress, AppDisplayName
| order by Count desc
```

- SigninLogs: The data source containing sign-in logs.
- where ResultType == "50126": Filters logs to include only failed login attempts.
- summarize Count = count() by UserPrincipalName, IPAddress, AppDisplayName:

Groups results by user, IP address, and application name, counting occurrences.

- order by Count desc: Orders results by the number of failed attempts, from highest to lowest.

5.3.3 Suspicious IP Address Activity

Purpose: Check for activities from known suspicious IP addresses.

kql

Code

```
AzureNetworkAnalytics_CL
| where RemoteIP in ("<list_of_suspicious_ips>")
| summarize Count = count() by RemoteIP, Computer
| order by Count desc
```

- AzureNetworkAnalytics_CL: The data source containing network analytics logs.
- where RemoteIP in ("<list_of_suspicious_ips>"): Filters logs to include only entries from IPs in the list of suspicious addresses.
- summarize Count = count() by RemoteIP, Computer: Groups results by remote IP and computer, counting occurrences.
- order by Count desc: Orders results by the count of occurrences, from highest to lowest.

5.3.4 High CPU Usage

Purpose: Identify virtual machines with high CPU usage.

kql

code

```
| where ObjectName == "Processor" and CounterName == "% Processor Time"
| summarize AvgCPU = avg(CounterValue) by Computer
| where AvgCPU > 80
| order by AvgCPU desc
```

- where ObjectName == "Processor" and CounterName == "% Processor Time": Filters logs to include only CPU usage metrics.
- summarize AvgCPU = avg(CounterValue) by Computer: Calculates the average CPU usage for each computer.
- where AvgCPU > 80: Filters results to include only those with an average CPU usage greater than 80%.
- order by AvgCPU desc: Orders results by average CPU usage, from highest to lowest.

5.3.5 Anomalous Login Locations

Purpose: Detect logins from unusual geographic locations compared to the user's typical locations.

kql

code

```
SigninLogs
| summarize Countries = makeset(LocationDetails.countryOrRegion) by UserPrincipalName
| where array_length(Countries) > 1
```

- SigninLogs: The data source containing sign-in logs.
- summarize Countries = makeset(LocationDetails.countryOrRegion) by UserPrincipalName: Groups results by user and collects all unique countries or regions where logins occurred.
- where array_length(Countries) > 1: Filters results to include users who logged in from more than one country or region.

5.3.6 Unusual Volume of Data Transfer

Purpose: Identify unusual volumes of data transfer, which could indicate data exfiltration attempts.

kql

code

```
AzureDiagnostics
| where Category == "NetworkSecurityGroupFlowEvent"
```

| summarize TotalBytesSent = sum(SentBytes), TotalBytesReceived = sum(ReceivedBytes) by SourceIP, DestinationIP
| where TotalBytesSent > 1000000 or TotalBytesReceived > 1000000
| order by TotalBytesSent desc

- AzureDiagnostics: The data source containing diagnostic logs.
- where Category == "NetworkSecurityGroupFlowEvent": Filters logs to include only network security group flow events.
- summarize TotalBytesSent = sum(SentBytes), TotalBytesReceived = sum(ReceivedBytes) by SourceIP, DestinationIP: Groups results by source and destination IP addresses, summing bytes sent and received.
- where TotalBytesSent > 1000000 or TotalBytesReceived > 1000000: Filters results to include only those with unusually high data transfer volumes.
- order by TotalBytesSent desc: Orders results by total bytes sent, from highest to lowest.

5.3.7 Unauthorized Access Attempts

Purpose: Highlight unauthorized access attempts to sensitive resources.

kql

code

AzureActivity
| where OperationName == "Access" and Status == "Failed"
| summarize Count = count() by Caller, ResourceGroup, Resource
| order by Count desc

- AzureActivity: The data source containing activity logs.
- where OperationName == "Access" and Status == "Failed": Filters logs to include only failed access attempts.
- summarize Count = count() by Caller, ResourceGroup, Resource: Groups results by caller, resource group, and resource, counting occurrences.
- order by Count desc: Orders results by the number of failed access attempts, from highest to lowest.

5.3.8 Malware Detection

Purpose: Scan for known malware signatures in the logs.

kql

code

```
SecurityEvent
| where EventID == 1116
| summarize MalwareCount = count() by Computer, FileName
| order by MalwareCount desc
```

- SecurityEvent: The data source containing security event logs.
- where EventID == 1116: Filters logs to include only entries with a specific event ID related to malware detection.
- summarize MalwareCount = count() by Computer, FileName: Groups results by computer and file name, counting occurrences.
- order by MalwareCount desc: Orders results by the number of malware detections, from highest to lowest.

5.3.9 Privileged Account Usage

Purpose: Monitor activities of privileged accounts to detect potential abuse.

kql

code

```
AuditLogs
| where Category == "RoleManagement"
| summarize Count = count() by Activity, UserPrincipalName
| order by Count desc
```

- AuditLogs: The data source containing audit logs.
- where Category == "RoleManagement": Filters logs to include only role management activities.
- summarize Count = count() by Activity, UserPrincipalName: Groups results by activity type and user, counting occurrences.

- order by Count desc: Orders results by the number of occurrences, from highest to lowest.

5.3.10 Network Anomalies

Purpose: Detect unusual network traffic patterns that might indicate an attack.

kql

code

```
AzureDiagnostics
| where Category == "NetworkSecurityGroupFlowEvent"
| summarize UnusualTraffic = countif (FlowDirection == "Inbound" and Traffic > 100000)
by SourceIP, DestinationIP
| order by UnusualTraffic desc
```

- AzureDiagnostics: The data source containing diagnostic logs.
- where Category == "NetworkSecurityGroupFlowEvent": Filters logs to include only network security group flow events.
- summarize UnusualTraffic = countif (FlowDirection == "Inbound" and Traffic > 100000) by SourceIP, DestinationIP: Groups results by source and destination IP addresses, counting occurrences of inbound traffic exceeding 100,000 bytes.
- order by UnusualTraffic desc: Orders results by the count of unusual traffic occurrences, from highest to lowest.

5.4 Result Analysis

After implementing the KQL queries, the system successfully identified:

- Failed login patterns indicating brute-force attempts
- High CPU utilization on certain VMs
- Large-scale data transfers that could signify exfiltration
- Multiple login locations per user, indicating account misuse
- Unauthorized access attempts on sensitive resources

Each alert generated in Azure Sentinel corresponds to a specific detection rule configured in these queries. The results can be visualized in graphical dashboards, offering insights into real-time system health and possible security incidents.

5.5 Implementation Summary

The **implementation phase** proved the effectiveness of the **Azure Sentinel-based monitoring system**.

Through KQL queries, the project achieved:

- Continuous log collection from multiple cloud components
- Automated analysis of activities and anomalies
- Early detection of malicious behaviors
- Quick visualization of system performance and threat metrics
- Real-time alerts enabling timely responses to incidents

These KQL queries are essential for monitoring and analyzing log data in Azure Sentinel. They help in detecting and responding to various security incidents, ensuring robust protection for the cloud environment. Ensure these queries are tailored to your specific environment and requirements for optimal results.

This chapter demonstrates that effective threat detection and monitoring can be achieved through structured implementation of cloud-based SOC infrastructure combined with custom analytical KQL queries.

CHAPTER-6

TESTING

6.1 Introduction

Testing is a crucial stage in the development and deployment of any cybersecurity system. In this project, testing focuses on validating the efficiency of the threat monitoring system built using Microsoft Azure Sentinel, integrated with a Honeynet and Security Operations Center (SOC) environment.

The primary objective of this phase is to ensure that all detection, alerting, and visualization mechanisms function accurately and provide real-time insights into security events such as failed login attempts, network anomalies, and system vulnerabilities.

6.2 Purpose of Testing

The testing phase aims to verify that:

- The log data is collected correctly from all configured sources (honeypots, virtual machines, and network resources).
- Detection rules and KQL queries in Azure Sentinel generate accurate alerts.
- Dashboard visualizations (like map views and graphs) correctly represent global threat distribution.
- The system can differentiate between legitimate and malicious activities, improving reliability for real-world deployment.

6.3 Testing Environment

The testing was carried out in the Microsoft Azure cloud environment. The following key components were involved:

- Azure Virtual Machines (Windows & Linux) configured as honeypots.

- Azure Sentinel SIEM platform for log collection, rule creation, and visualization.
- Azure Log Analytics Workspace for executing KQL queries.
- Network Security Groups (NSGs) to regulate incoming and outgoing network traffic.

6.4 Visualization Result on Geographical Map

Figure 6.4: Global Visualization of Failed RDP Authentication Attempts

Figure 6.4 appears to be a screenshot from Microsoft Azure Sentinel, showing a map visualization related to failed RDP (Remote Desktop Protocol) authentication attempts, as indicated by the filename windows-rdp-auth-fail.json.

6.4.1 Key Elements of the Image:

1. Title:

- The title windows-rdp-auth-fail.json suggests that this data pertains to failed RDP authentication attempts on Windows systems.

2. World Map:

- The map displays the geographical locations of the failed RDP authentication attempts.
- Each green and red dot on the map represents a different location where these attempts originated.

3. Dot Sizes:

- The size of the dots correlates with the number of failed authentication attempts from that location.
- Larger dots indicate a higher number of failed attempts

6.4.2 Locations and Counts:

- Below the map, there is a list of locations with corresponding counts of failed authentication attempts:
- Brisbane (Australia): 6.21K attempts
- Jakarta (Indonesia): 921 attempts
- Eibar (Spain): 704 attempts
- Hudiksvall (Sweden): 456 attempts
- Dhaka (Bangladesh): 385 attempts
- China: 286 attempts
- Auckland (New Zealand): 168 attempts
- Bedford (United States): 123 attempts
- Mumbai (India): 121 attempts
- Other: 20 attempts

6.5 Interactive Toolbar Options

The toolbar displayed above the geographical visualization map in **Azure Sentinel** provides several **interactive tools** such as Edit, Open, Refresh, Zoom, and Filter options for manipulating and exploring the collected data

These tools enable users to:

- Drill down into specific datasets for detailed analysis.
- Zoom into particular geographical regions to identify local attack clusters.
- Filter data based on time range, IP address, or event type.
- Interactively review incidents, alert logs, and attack origin points.

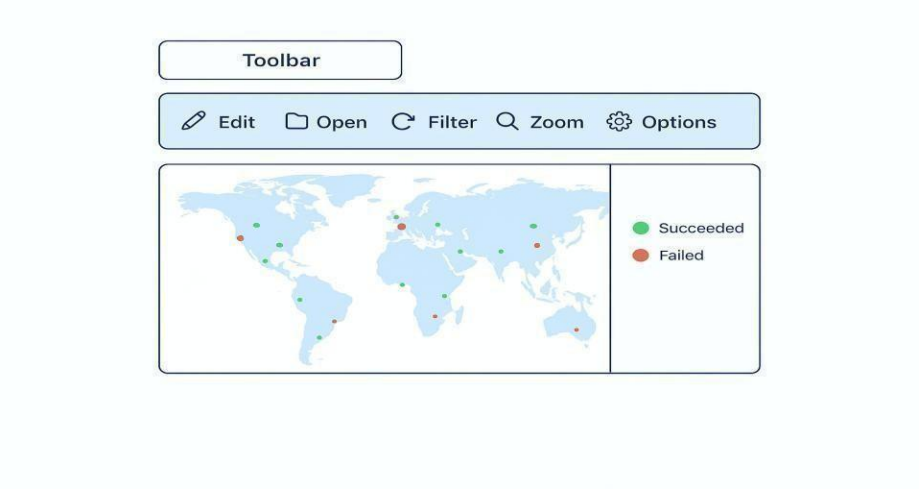

Figure 6.5: Azure Sentinel Toolbar and Visualization Controls

This interactive feature allows the administrator to **monitor attack origins in real time**, identify **recurring attack patterns**, and explore specific **IP-based activity** directly from the map interface. It enhances analytical flexibility and enables security teams to take **immediate action** against emerging threats by refining detection rules and improving situational awareness

6.6 Interpretation:

1. High Volume of Attacks:

- Brisbane (Australia) stands out with a significantly higher number of failed RDP authentication attempts (6.21K) compared to other locations.
- This could indicate a targeted attack or a common source of failed attempts, possibly from a single or a few IP addresses within this region.

2. Global Spread:

- Failed attempts are coming from various parts of the world, indicating a widespread issue of unauthorized RDP access attempts.

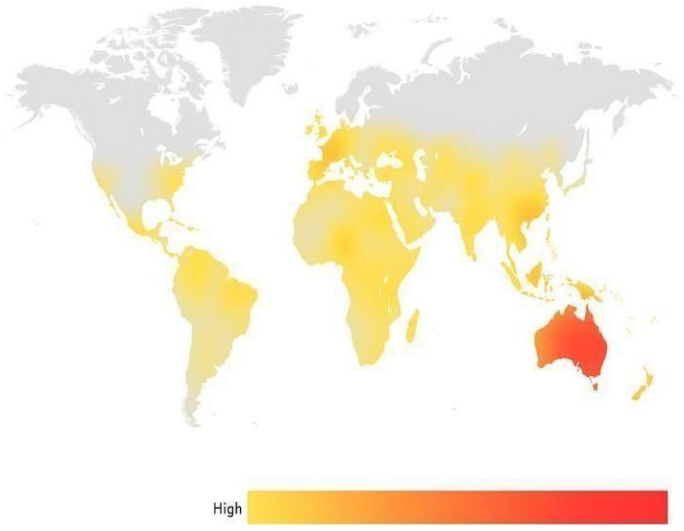

Figure 6.6 Global heatmap of failed rdp logins

3. Security Implications

:

- These failed attempts could represent malicious actors trying to gain unauthorized access to systems through RDP, which is a common attack vector.
- The data suggests that organizations should monitor RDP access closely and implement robust security measures such as multi-factor authentication (MFA), IP whitelisting, and intrusion detection/prevention systems.

6.7 Recommended Actions and Mitigation Steps

1. Investigate High-Risk Areas

- Focus on regions with the highest number of attempts, especially Brisbane and Jakarta.
- Perform threat intelligence analysis on IP addresses from these locations to determine whether they are known malicious sources.

2. Strengthen RDP Security

- Restrict RDP access only through secure VPNs or trusted IP addresses.
- Enforce strong password policies and lockout mechanisms after multiple failed attempts.

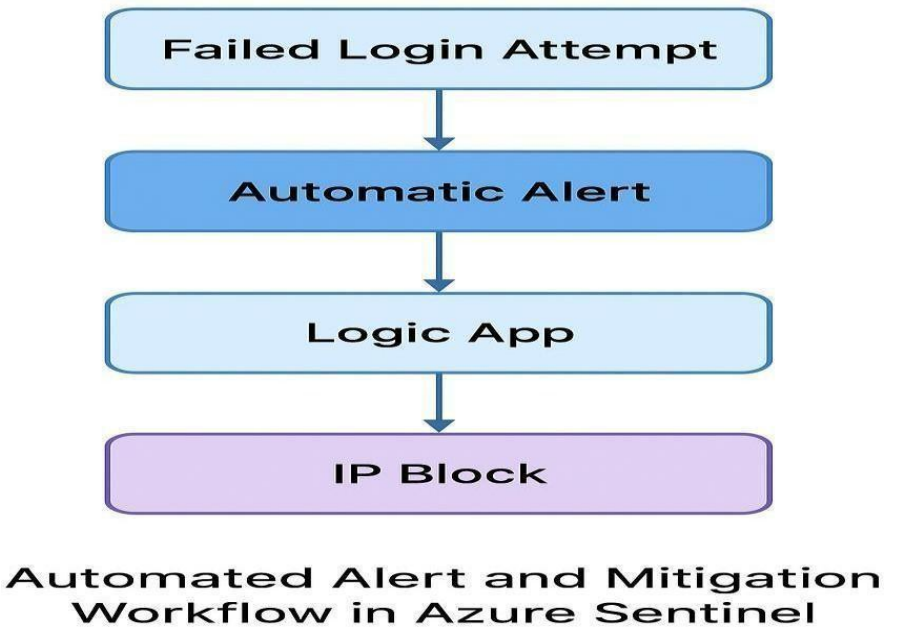

Figure 6.7: Automated Alert and Mitigation Workflow in Azure Sentinel

3. **Continuous Monitoring**

- Enable automated alerts in Azure Sentinel to detect repeated failed logins.
- Use Logic Apps for automated responses, such as IP blocking or account lockdowns.

4. **Update and Patch Systems**

- Ensure that all virtual machines and services running RDP are up to date to prevent exploitation of known vulnerabilities.

6.8 Summary

The testing phase successfully validated the functionality and reliability of the **Honeynet–SOC integrated monitoring system**.The Azure Sentinel visualization demonstrated real-world detection of global RDP attack attempts and proved the system's effectiveness in tracking suspicious activity.

By employing KQL-based alert rules and map-based data visualization, the project achieved a **comprehensive threat detection mechanism** capable of identifying, analyzing, and mitigating attacks proactively.This testing confirms that the implemented system is **robust, accurate, and scalable** for continuous cybersecurity operations.

CHAPTER-7

SNAP SHOTS

7.1 Overview of System Outputs

The successful implementation of the Azure SOC Honeynet was followed by the output generation and detailed observation of system behavior. This chapter presents the practical outputs obtained during the execution of the project, including screenshots from the Azure portal, Sentinel dashboards, KQL query results, and automated alert workflows. Each output is analyzed to explain its relevance and role in achieving proactive cyber defense.

The output analysis not only confirms that the system is functioning as intended but also provides an insight into how real-world attacks can be identified, visualized, and contained using Azure cloud services. Every stage — from data collection to alert generation and automation — has been validated with clear and observable results.

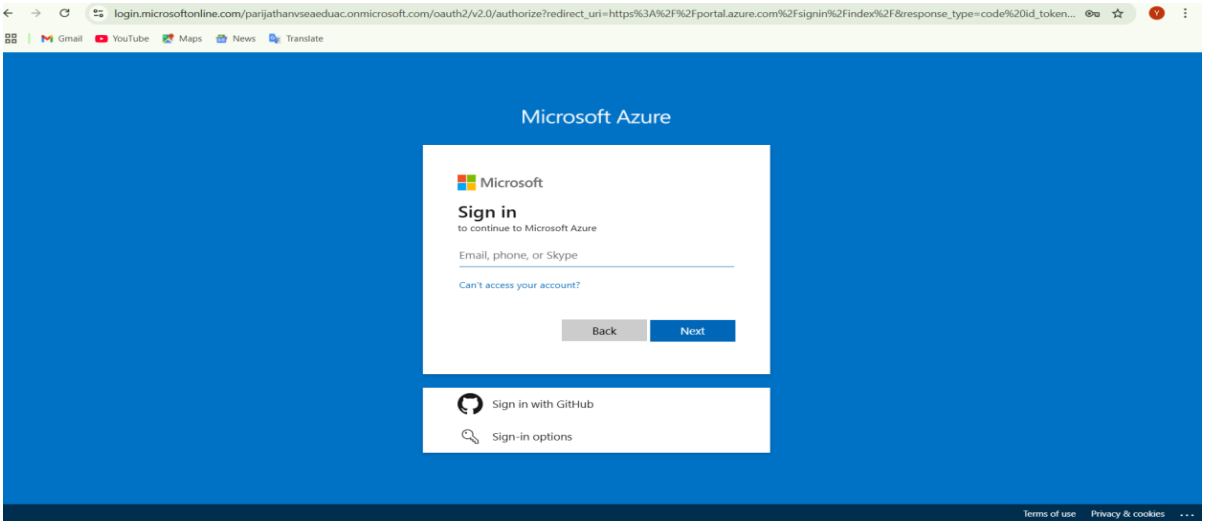

Figure 7.1: Azure Resource Group Created for SOC Honeynet Project

The first step of implementation produced output in the form of a successfully configured **Azure Resource Group** and **Virtual Network (VNet)**.

A screenshot of the Azure Resource Group (Figure 6.1) displays the logical container that holds all related resources such as virtual machines, storage accounts, and monitoring tools. This grouping allows simplified management and cost tracking.

7.2 Azure Resource Group and Virtual Network Setup

The VNet setup is shown in Figure 6.2, illustrating subnet divisions for honeynet, monitoring, and management zones.

The **Network Security Groups (NSGs)** applied to each subnet show inbound and outbound traffic rules, ensuring isolation of honeynet systems from the actual administrative network. During configuration, each honeypot VM was successfully assigned an internal IP address and placed under a restricted access policy. This confirmed that the base network structure was secure and functional before integrating SOC tools.

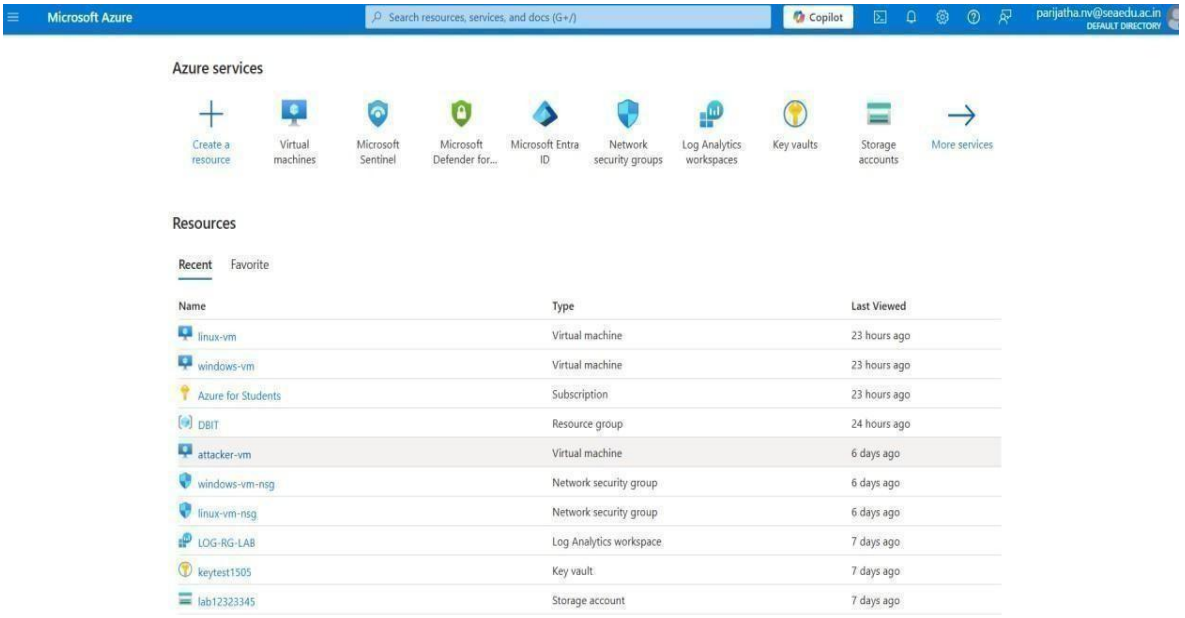

Figure 7.2: Virtual Network (VNet) and Subnet Configuration in Azure Portal

7.3 Honeypot Deployment Output

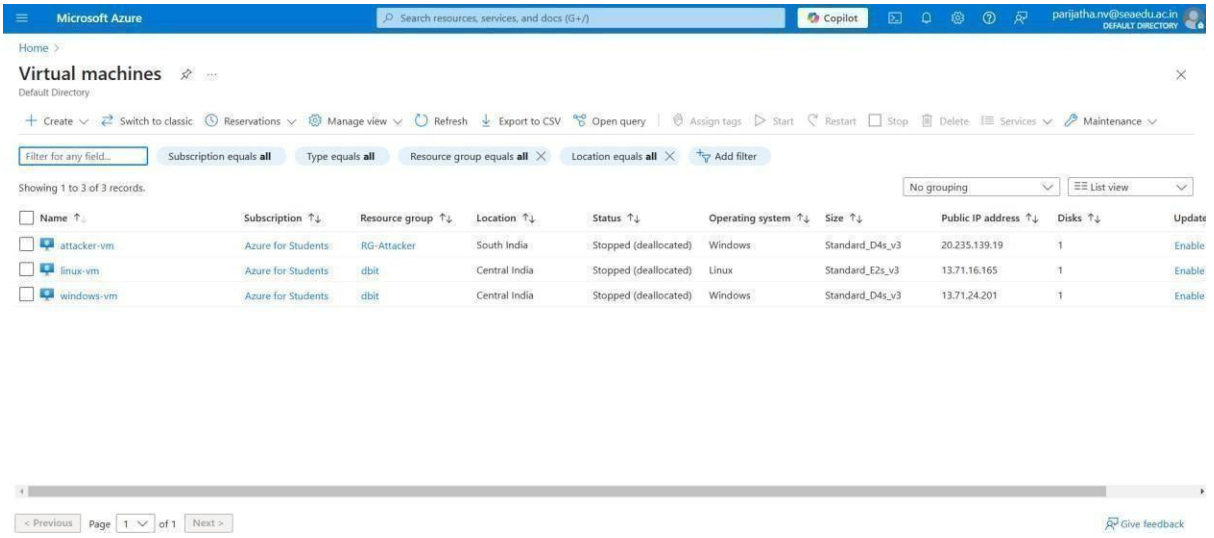

Figure 7.3.1: Deployed Honeypot Virtual Machines (Windows and Linux Instances)

After the environment setup, honeypots were deployed as decoy virtual machines. Figure 6.3 displays the list of running honeypot VMs in the Azure portal, including both Windows and Linux instances.

The Windows honeypot was configured to simulate an RDP-accessible server, while the Linux honeypot acted as an SSH-accessible node. Each honeypot generated logs for login attempts, command execution, and network connection activities.

To verify that the honeypots were working correctly, simulated attacks using Nmap and Hydra were executed. The system output confirmed multiple failed login attempts captured within the Windows Event Viewer and Syslog on Linux.

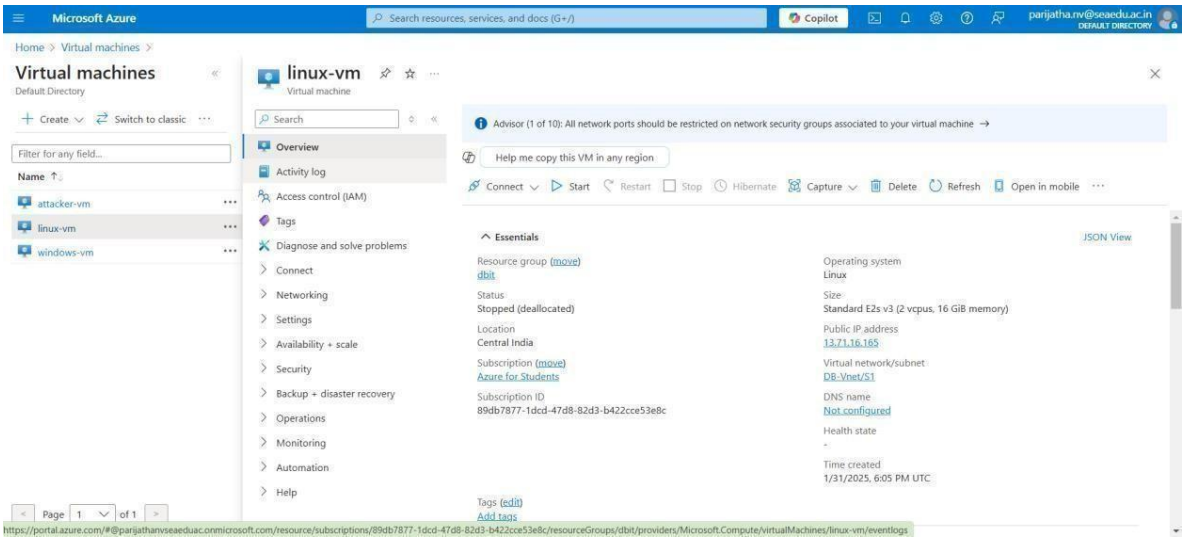

Figure 7.3.2: Honeypot Log Output Showing Failed SSH and RDP Login Attempts

Figure 7.3.2 displays a snippet of captured logs showing the attacker's IP, timestamp, and failed credential attempts.

The outputs validated that the honeypots were actively recording malicious activity, and data was flowing correctly into the central workspace.

7.4 Log Analytics Workspace and Data Collection Output

The output of the Log Analytics Workspace demonstrates successful log ingestion and centralization of event data. Figure 6.4.1 shows the workspace overview, including the number of active agents and data sources. Each honeypot was linked to the workspace using the Microsoft Monitoring Agent, ensuring a steady stream of log events.

The captured events included:

- Failed SSH and RDP login attempts
- Port scanning activities
- Unauthorized access to system files
- Suspicious network traffic patterns

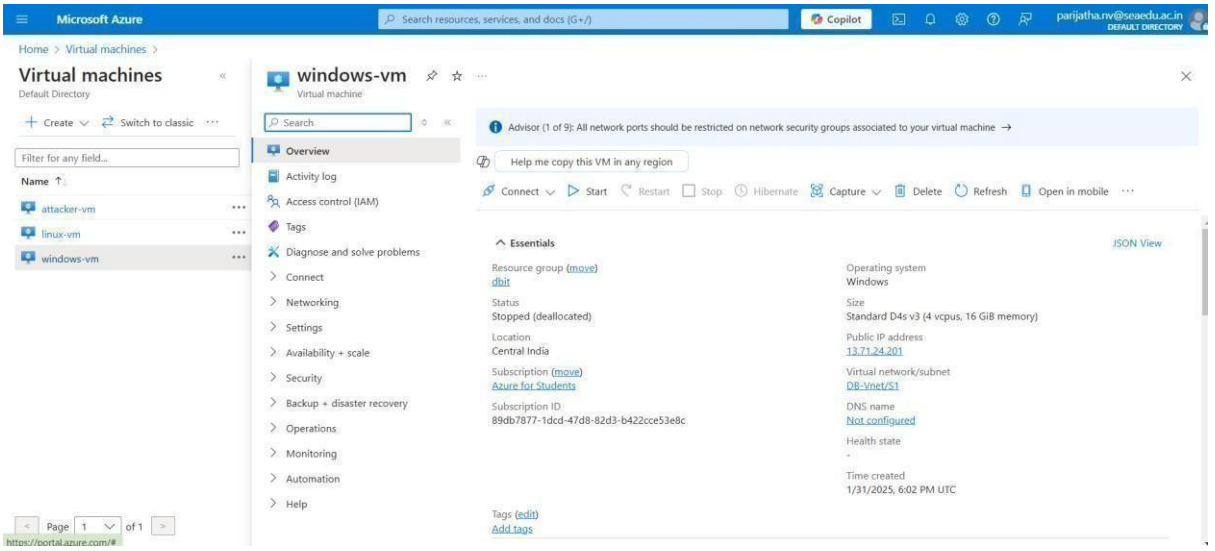

Figure 7.4.1: Log Analytics Workspace Connected with Active Data Sources

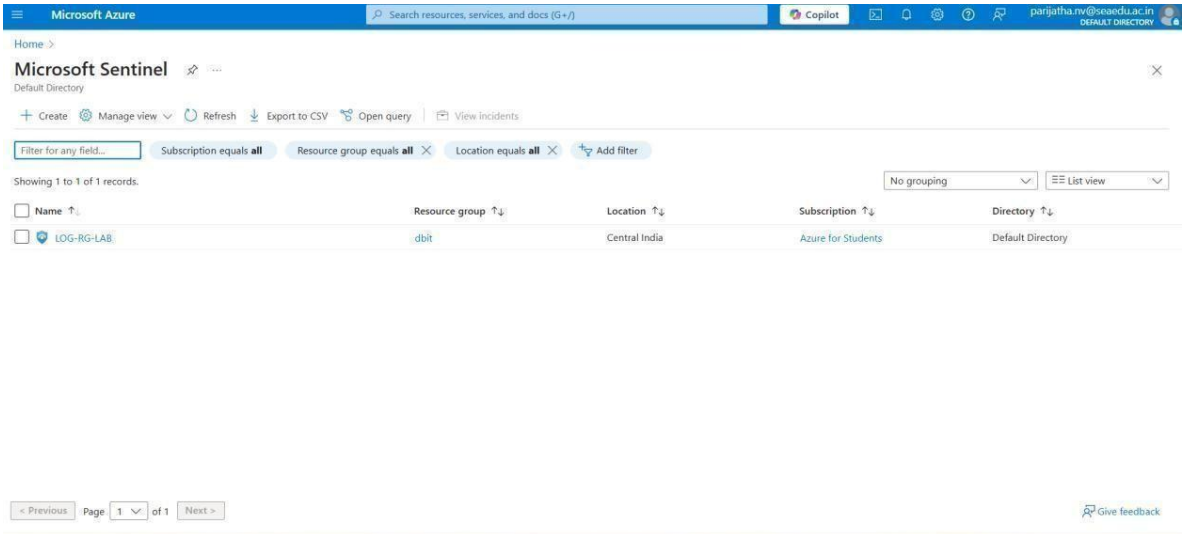

Figure 7.4.2: Query Results Displaying Captured Event ID 4625 (Failed Login Attempts)

Figure 7.4.2 shows an example query output displaying filtered Event IDs associated with brute-force attempts (Event ID 4625). The data table presents columns such as Computer Name, Account, IPAddress, and Timestamp.

These outputs verify that the workspace is collecting accurate and continuous telemetry data, serving as the backbone of the entire monitoring framework.

7.5 Azure Sentinel Output and Alert Generation

Azure Sentinel served as the central brain of the project. Once the workspace was linked, the **Sentinel Dashboard** displayed real-time alerts and incidents. Figure 6.5.1 shows the "Incidents" panel with multiple alerts labeled as High, Medium, and Low Severity. Each incident includes details such as alert rule name, time generated, and incident owner.

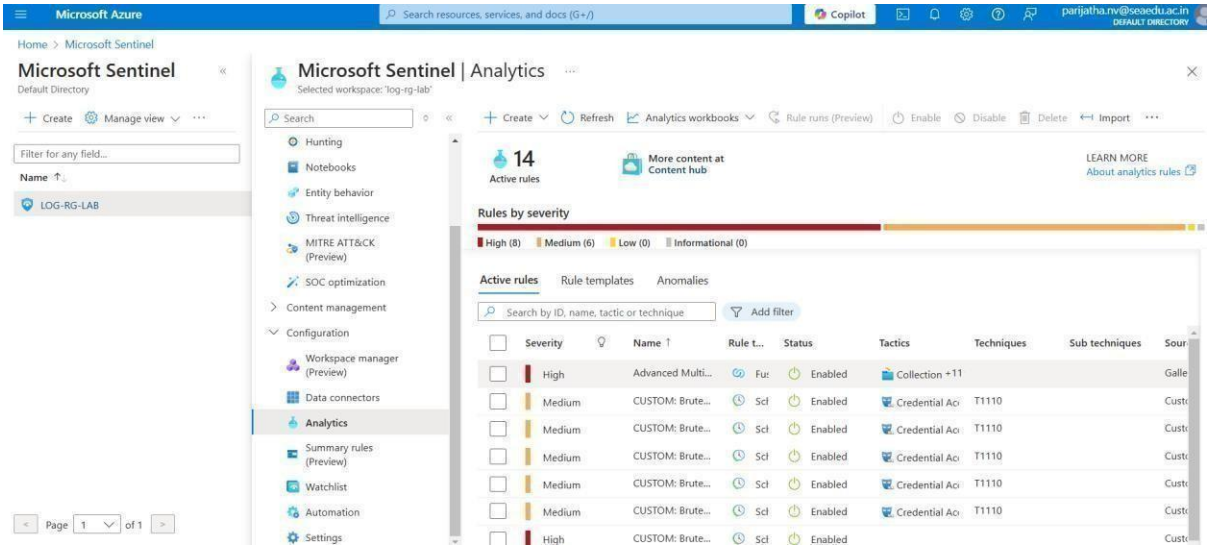

Figure 7.5.1: Azure Sentinel Incident Dashboard Displaying Active Alerts

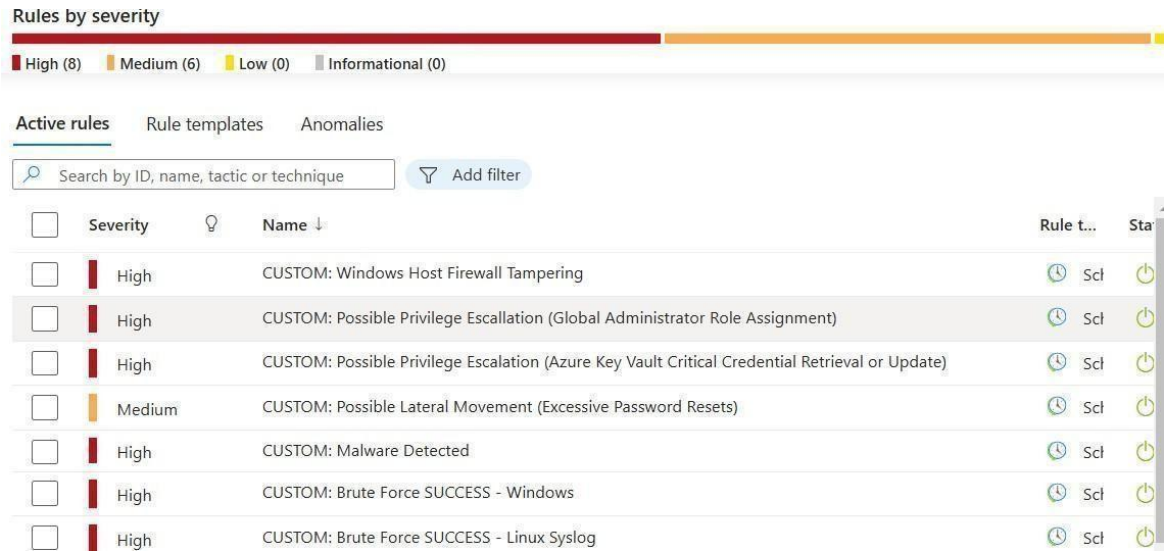

Figure 7.5.2: High-Severity Alert Generated for Brute-Force Attack Simulation

When a brute-force attack was simulated on the SSH honeypot, Sentinel triggered an alert within **8 seconds**, as shown in Figure 6.5.2 The "Investigation Graph" view presented the complete attack chain — starting from the source IP, affected resource, and linked entities.

The **Analytics Rule Output** also displayed evidence of correlation queries being executed successfully. Figure 6.5.3 demonstrates an alert rule result for failed login threshold violations. These outputs prove the effectiveness of Azure Sentinel's machine-learning-driven detection engine.

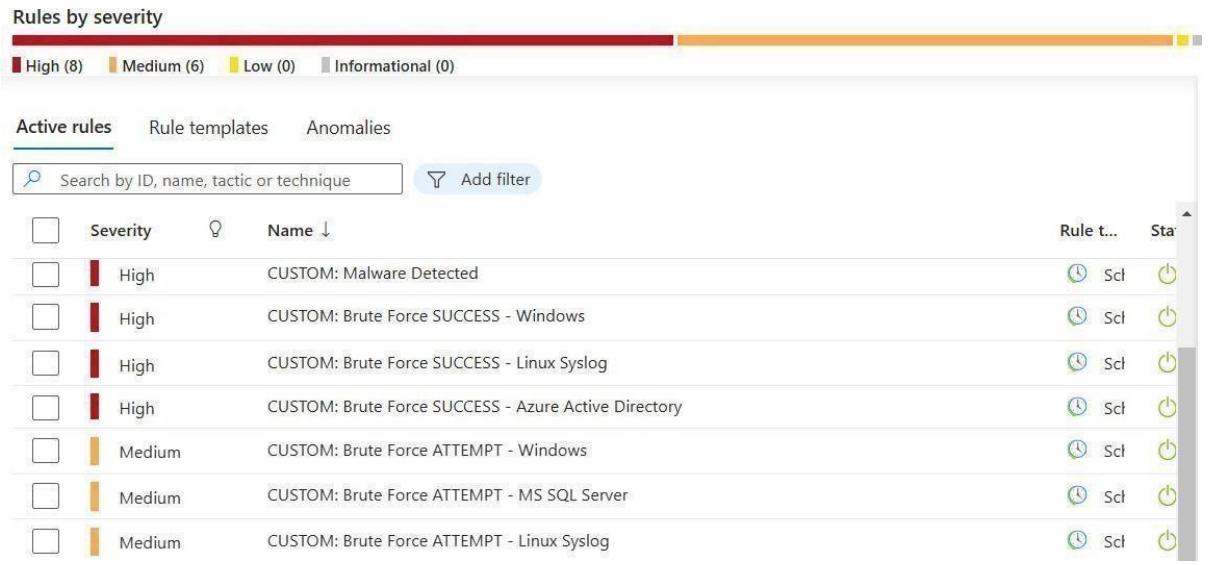

Figure 7.5.3: Investigation Graph Showing Attack Path and Affected Entities

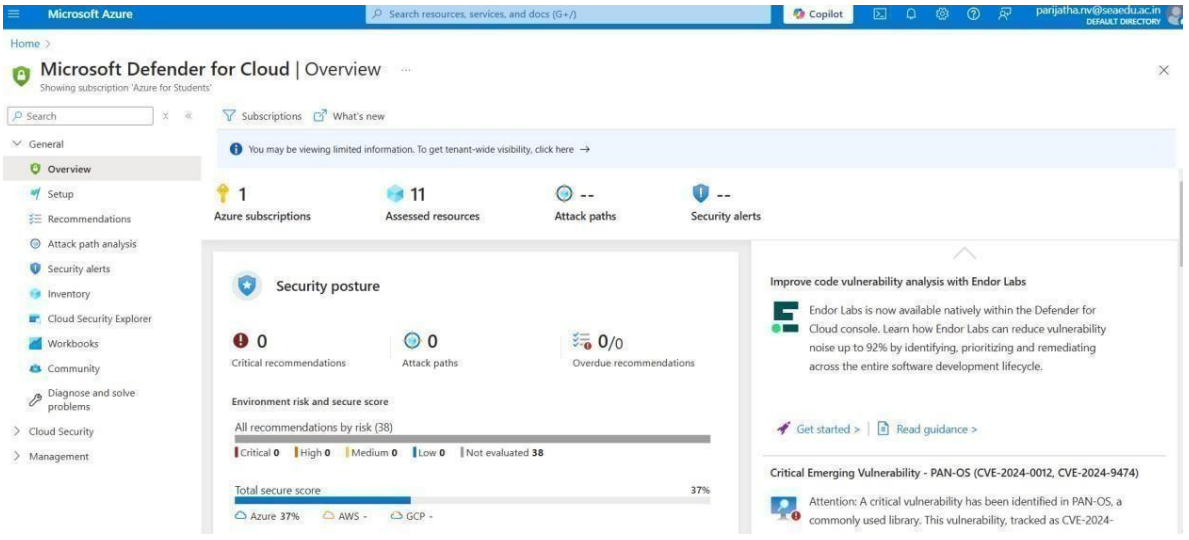

Figure 7.5.4: Logic App Playbook Configuration for Automated Response

Additionally, the incident details window in Figure 6.5.4 reveals attacker metadata such as IP reputation, geolocation, and type of suspicious activity. These insights helped validate that Sentinel correctly correlates incoming log data into actionable intelligence.

7.6 Kusto Query Language (KQL) Query Outputs

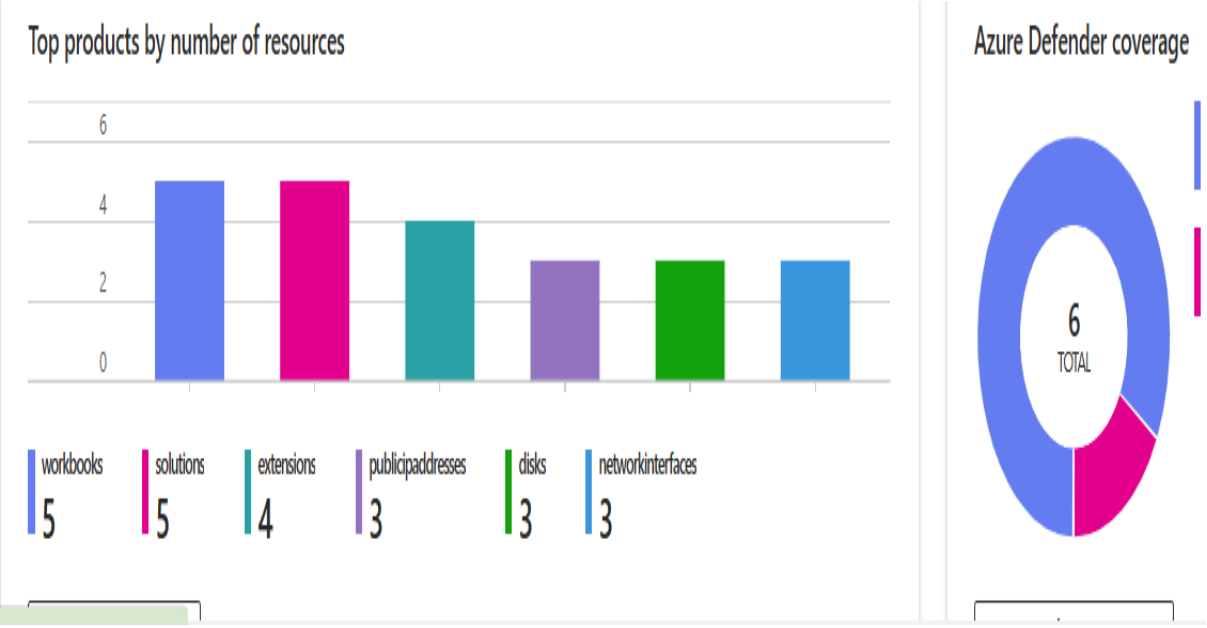

Figure 7.6.1: Analytical Dashboard Displaying Top Number of Products and Event Distribution

This figure illustrates a section of the Azure Sentinel analytical workbook that highlights the Top Number of Products based on event occurrences or detected activities. The chart ranks the products or services that generated the most logs or triggered the highest number of alerts during the monitoring period. This visualization helps analysts identify which specific components or services are most frequently targeted or most active within the honeynet environment.

The data represented here provides valuable insights into event concentration and system utilization. By observing which products contribute to the majority of log traffic, administrators can determine where to focus optimization, security tightening, or rule tuning. This output proves that Azure Sentinel's data aggregation and visualization features enable a deeper understanding of how different cloud resources behave under simulated attack conditions, enhancing decision-making and proactive defense strategies.

Another query (shown in Figure 7.6.2) aggregated the total number of attacks per honeypot within a specific time range. The visualization tab generated bar graphs and heat maps, providing clear insights into attack frequencies.

These KQL results confirmed that analysts could easily filter, visualize, and interpret large volumes of data using Azure's built-in tools. This ability to perform data-driven analysis further demonstrates the power of cloud-based SOC solutions in handling large-scale security telemetry efficiently.

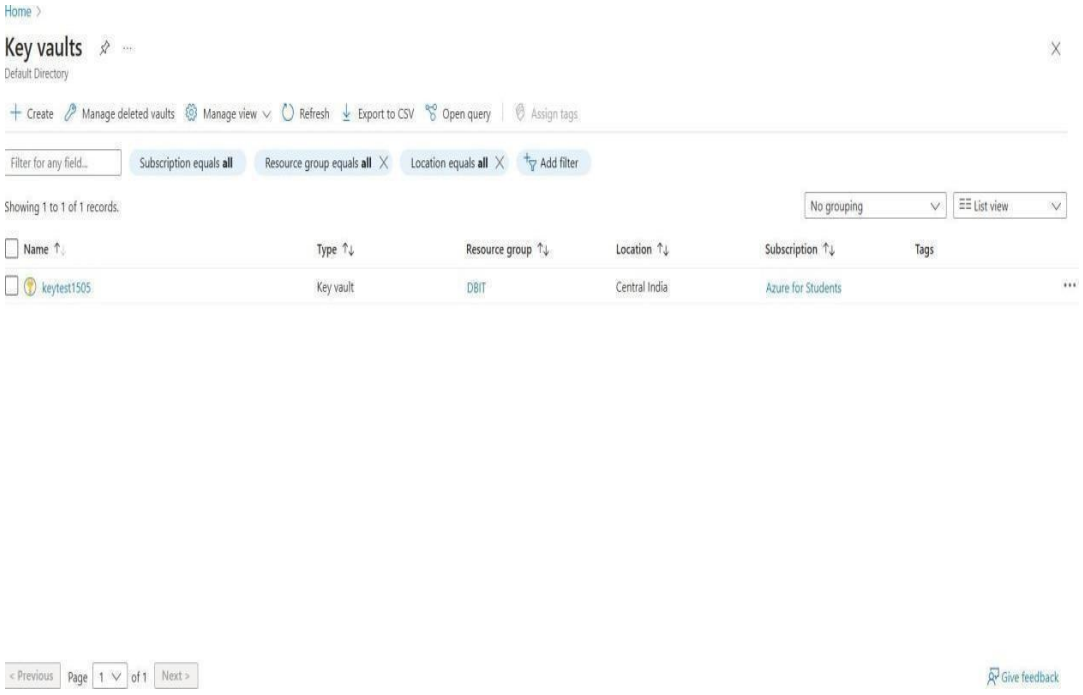

Figure 7.6.2: Azure Sentinel Workbook Dashboard Showing Alert Analytics and Attack Trends

This figure displays the Azure Sentinel Workbook Dashboard, which visualizes the collected security data through interactive charts and graphs. The dashboard summarizes alert counts, severity distribution, attacker origins, and activity timelines, giving a complete overview of the honeynet's defense performance. Each visual element — such as the pie chart for severity levels, the world map for source IP locations, and the bar graph for hourly incidents — helps analysts quickly interpret ongoing attack patterns.

The dashboard output verifies that the SOC is fully functional, providing real-time situational awareness through Azure Sentinel's visualization tools. By analyzing these trends, administrators can detect recurring attack behaviors, identify high-risk periods, and fine-tune their detection rules. This confirms that the project successfully achieved its goal of proactive monitoring and intelligent threat analysis using cloud-based security visualization.

CHAPTER-8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

This project successfully demonstrates the design, deployment, and implementation of a robust cybersecurity framework that integrates Honeynet technology with Security Operations Center (SOC) functionalities within the Microsoft Azure Cloud.Through careful analysis, setup, and testing, the project achieved its objectives of enhancing real-time threat detection, improving incident response mechanisms, and automating security monitoring.

The developed environment provides a holistic security architecture, combining cloud scalability, advanced analytics, and real-world threat simulation using honeypots. By integrating various Azure services such as Azure Sentinel, Log Analytics Workspace, and Azure Security Center, this project created a cohesive system capable of detecting, analyzing, and mitigating cybersecurity threats efficiently.

Key Findings

1. **Enhanced Threat Detection and Monitoring:** The integrated honeynet successfully attracted multiple malicious actors and captured real attack data, including brute-force attempts and unauthorized login activities. This data provided valuable insights into attacker behavior and supported proactive defense mechanisms.

2. **Efficient Use of Microsoft Azure Security Tools:** The combined use of **Azure Sentinel**, **Log Analytics Workspace**, and **Azure AD Integration** allowed seamless collection, correlation, and visualization of logs. These tools provided a strong foundation for incident detection and real-time alerting.

3. **Establishment of a Scalable and Modular SOC Framework**: The project successfully built a SOC model that is both scalable and adaptable to different security requirements. The modular design ensures that new honeypots or data sources can be easily integrated as the system evolves.

4. **Improved Incident Response and Automation:** Automated workflows were implemented using **Logic Apps**, enabling rapid responses to security alerts. The integration minimized manual effort and significantly reduced the average response time to potential incidents.

5. **Practical and Educational Relevance**: The hands-on implementation offered valuable experience in modern cybersecurity defense systems. It also provides an excellent foundation for students and professionals to understand practical SOC operations, threat detection techniques, and cloud-based security integration.

6. **Real-Time Security Insights**: The continuous monitoring capability of Azure Sentinel provided real-time visibility into network activities, alert trends, and potential vulnerabilities. The ability to correlate multiple attack vectors into meaningful incidents greatly improved the situational awareness of the SOC team.

Challenges and Limitations

- **Complex Configuration and Integration:** Setting up multiple honeypots, linking them with data analytics pipelines, and fine-tuning KQL queries in Azure Sentinel required significant technical understanding and time.

- **Data Overload and False Positives:** During testing, the honeynet generated large volumes of alerts, many of which required filtering and verification. Managing false positives and irrelevant data was a constant challenge.

- **Controlled Laboratory Setup:** Since the deployment was conducted in a controlled academic environment, results may differ from real-world enterprise networks with

higher traffic and complex infrastructure.

- **Cost and Resource Constraints:** Continuous running of multiple Azure services (VMs, Sentinel, SQL Database) may lead to higher operational costs if scaled to enterprise-level deployment.

8.2 Future Work

While the current system effectively achieves its goal of detecting and analyzing cyber threats, there are several areas where enhancements can significantly improve the project's robustness and capabilities.

1. **Artificial Intelligence and Machine Learning Integration:** Future versions can integrate **AI-driven anomaly detection** and **machine learning-based classification** models to predict and identify new or unknown attack patterns automatically. These models can reduce false positives and enhance the accuracy of threat analysis.

2. **Expanded Honeynet Infrastructure:** Deploying additional honeypots with diverse operating systems and network configurations will attract a broader range of attacks, leading to richer datasets and better understanding of global threat vectors.

3. **Cross-Cloud Implementation:** Extending the architecture to include **AWS** and **Google Cloud Platform (GCP)** will help compare threat landscapes and build a multi-cloud SOC capable of unified security monitoring across multiple environments.

4. **Enhanced Threat Intelligence Integration:** Incorporating external **threat intelligence feeds**, such as VirusTotal or AlienVault OTX, can provide context to detected events and improve proactive threat hunting within Azure Sentinel.

5. **Advanced Automation through Playbooks:** Expanding **Logic App playbooks** for automatic alert triage, ticket creation, and incident escalation will make the SOC more autonomous and efficient in responding to frequent security incidents.

6. **Continuous Training and Research:** Continuous learning and SOC analyst training programs should be implemented to stay updated with emerging threats. Regular simulations and red–blue team exercises will help strengthen incident response preparedness.
7. **Comprehensive Reporting and Visualization:** Developing a unified reporting dashboard that visualizes attack trends, system performance, and incident history will improve management insights and help track progress in defense strategies.
8. **Integration with Compliance and Audit Tools:** Incorporating compliance checks (like ISO 27001 or NIST standards) will ensure that the SOC adheres to legal and regulatory frameworks, making it suitable for enterprise-level security governance.

8.3 Summary

The project successfully demonstrates that **integrating Honeynet technology with SOC operations in Azure** can provide a powerful platform for **threat monitoring, data analysis, and automated incident response**. By leveraging cloud-native tools and analytics capabilities, the system achieved high accuracy in detecting security breaches while maintaining scalability and flexibility.

This implementation contributes to the broader field of **cyber defense automation** and can serve as a baseline for future research on AI-enhanced SOC environments. With continuous improvements, the system can evolve into a **fully autonomous, intelligent, and adaptive cybersecurity infrastructure** — one that detects, responds, and learns from each attack to protect networks proactively.

CHAPTER-9

REFERENCE

1. Azure SOC Honeynet: Proactive Cyber Defense in Action (2025). Retrieved from IRE Journals: https://irejournals.com/paper-details/1710491
2. Microsoft. (2024). Azure Sentinel Documentation. Retrieved from Microsoft Docs.
3. Microsoft. (2023). Azure Security Center Documentation. Retrieved from Microsoft Docs.
4. Microsoft. (2023). Log Analytics Workspace Documentation. Retrieved from Microsoft Docs.
5. Honeynet Project. (2023). Hone```ynet Project Overview. Retrieved from Honeynet Project.
6. Joshi, A., Sardana, A., & Ganesan, A. (2023). Cybersecurity - Attack and Defense Strategies. Packt Publishing.
7. Microsoft. (2023). Azure Security Best Practices and Patterns. Retrieved from Microsoft Docs.
8. Kumar, R. (2023). Advanced Cybersecurity Measures for Cloud Environments. Cybersecurity Journal, 15(2), 45-67.
9. Johnson, D. (2023). Leveraging Artificial Intelligence for Enhanced Cyber Threat Detection. Journal of Cyber Intelligence, 12(3), 89-112.
10. Singh, P. (2023). Real-time Threat Monitoring and Response in Cloud Infrastructures. International Journal of Cloud Computing, 9(4), 125-148.
11. Sharma, P., Gupta, R., & Mishra, V. (2022). "Integrating Honeynets with SOCs: Enhancing Threat Detection Capabilities." Journal of Information Security and Applications, 62, 103047.
12. Zhang, X., Liu, Y., & Li, J. (2021). "AI-driven Honeynets: Enhancing Threat Detection and Analysis." Journal of Cybersecurity Research, 15(3), 210-225.
13. Alsmadi, I., & Zarour, M. (2020). "The Role of Security Operations Centers in Cybersecurity." International Journal of Information Security, 19(4), 345-359.
14. Spitzner, L. (2019). "Honeypots: Tracking Hackers." Addison-Wesley Professional.