Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2025), B.Sc. in CSE (Day)

AI-Driven Zero-Day Threat Indication Anomaly Detection

Course Title:Artificial Intelligence Lab
Course Code: CSE-316
Section: 231-D3

Students Details

Name	ID
Promod Chandra Das	231002005
Chinmoy Debnath	231902029

Submission Date: 27/12/2025
Course Teacher's Name: Mr. Mozdaher Abdul Quader

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

1 Introduction **2**

1.1 Overview 2

1.2 Motivation 2

1.3 Problem Definition 2

1.3.1 Problem Statement 2

1.3.2 Complex Engineering Problem 3

1.4 Design Goals/Objectives 3

1.5 Application 3

2 Design/Development/Implementation of the Project **5**

2.1 Introduction 5

2.2 Implementation 5

2.2.1 Implementation and Workflow 6

2.3 Algorithms 8

3 Performance Evaluation **10**

3.1 Simulation Environment/ Simulation Procedure 10

3.2 Results Analysis/Testing 12

3.3 Results Overall Discussion 12

3.3.1 Complex Engineering Problem Discussion 13

4 Conclusion **14**

4.1 Discussion 14

4.2 Limitations 14

4.3 Scope of Future Work 15

Chapter 1

Introduction

1.1 Overview

AI-Driven Zero-Day Threat Indication Anomaly Detection is an artificial intelligence–based cybersecurity system designed to identify previously unknown or zero-day cyber threats. Traditional security systems rely heavily on predefined signatures and known attack patterns, which makes them ineffective against new and evolving attacks. This project adopts an unsupervised anomaly detection approach that learns normal system and network behavior and flags significant deviations as potential security threats. The system is capable of analyzing system and network telemetry data, assigning risk levels, and generating analyst-friendly alerts and reports to support security monitoring and investigation.

1.2 Motivation

Zero-day cyber attacks pose a serious challenge to modern cybersecurity systems because they exploit vulnerabilities that are unknown at the time of attack. Signature-based intrusion detection systems and rule-based security mechanisms often fail to detect such threats due to the absence of predefined patterns. With the rapid growth of networked systems and cloud infrastructure, there is a growing need for intelligent security solutions that can adapt to new attack behaviors. This project is motivated by the need to develop an AI-driven, signature-less detection system that can automatically learn normal behavior and identify suspicious activities indicative of zero-day threats.

1.3 Problem Definition

1.3.1 Problem Statement

Conventional cybersecurity systems are ineffective in detecting zero-day attacks because they depend on known signatures and predefined rules. As attackers continuously evolve their techniques, unknown attack behaviors often remain undetected until dam-

age has already occurred. The challenge is to design a system that can detect abnormal behavior without relying on labeled attack data or prior knowledge of attack types.

1.3.2 Complex Engineering Problem

The development of a zero-day threat detection system represents a complex engineering problem due to several factors. The system must analyze high-dimensional and heterogeneous data generated by system and network activities. It must operate without labeled attack data while maintaining accuracy and minimizing false positives. Additionally, the system must balance detection sensitivity with practical usability, ensuring that alerts are meaningful and interpretable. Integrating machine learning, data preprocessing, risk classification, explainability, and real-time simulation further increases system complexity.

1.4 Design Goals/Objectives

The main objectives of this project are:

To detect potential zero-day cyber threats using unsupervised machine learning techniques.

To learn normal system and network behavior automatically.

To identify abnormal deviations that may indicate security threats.

To classify detected anomalies into different risk levels.

To generate explainable alerts and detailed analysis reports.

To support real-time, SOC-style monitoring and demonstration.

1.5 Application

The proposed system can be applied in enterprise networks, cloud infrastructures, and data centers to enhance security monitoring. It can assist Security Operation Centers (SOC) by highlighting suspicious activities that require further investigation. The system is also suitable for academic research and cybersecurity training, where it can be used to demonstrate AI-based anomaly detection techniques. Additionally, it can serve as a foundational framework for developing advanced intrusion detection and threat intelligence systems.

Table 1.1: Summary of the complex engineering attributes addressed by the project

Name of the P Attributes	How the project addresses the attribute
P1: Depth of knowledge required	Requires knowledge of cybersecurity fundamentals, anomaly detection concepts, unsupervised machine learning (Isolation Forest), feature engineering, data preprocessing (encoding/scaling), and Python-based data analysis.
P2: Range of conflicting requirements	Balances detection sensitivity vs. false positives, interpretability vs. model complexity, and real-time alerting vs. computational efficiency. It must detect unknown threats without labeled attack data while producing usable alerts.
P3: Depth of analysis required	Performs multi-stage analysis including data cleaning, normalization, baseline learning, anomaly scoring, risk classification, and explainability (identifying top contributing features) for each detected event.
P4: Familiarity of issues	Addresses real-world security challenges such as zero-day attacks, unknown intrusion behaviors, abnormal login patterns, port-scan-like activity, and unusual network traffic patterns where signatures are unavailable.
P5: Extent of applicable codes and standards	Applies machine learning engineering practices (train/test split for baseline learning), structured logging and reporting, reproducible experimentation, and SOC-oriented alert formatting. Can be aligned with security monitoring practices used in IDS/SIEM workflows.
P6: Stakeholder involvement and conflicting needs	Supports stakeholders such as SOC analysts, system administrators, and security teams who require explainable alerts, prioritized risk levels, and evidence reports, while maintaining usability for academic demonstration and evaluation.
P7: Interdependence	Integrates multiple components: dataset handling (CSV/XLSX), preprocessing pipeline, ML model training, anomaly detection, risk scoring, explainability, visualization, real-time simulation, and output report/log generation, where each module depends on others for correct results.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This project is designed as a modular AI-driven cybersecurity system focused on zero-day threat indication through anomaly detection. The system integrates multiple stages including data ingestion, preprocessing, unsupervised machine learning, anomaly scoring, risk classification, and alert generation. Python is used as the core implementation language due to its extensive support for data science and machine learning libraries, which enables rapid experimentation, scalability, and maintainability.

The system is intentionally designed to operate without relying on predefined attack signatures or labeled datasets. Instead, it learns a baseline representation of normal system and network behavior and identifies deviations from this baseline as potential threats. This design makes the system suitable for detecting previously unseen or evolving cyber attacks, commonly referred to as zero-day threats. Additionally, the modular structure allows individual components such as data preprocessing, model training, detection logic, and reporting to be modified or extended independently.

2.2 Implementation

The implementation follows a structured and systematic pipeline where raw telemetry data is progressively transformed into actionable security insights. The system supports both real datasets and a built-in demo simulation mode, ensuring functionality even in the absence of external data during testing or demonstrations.

Initially, input data is loaded from CSV or Excel files containing system or network telemetry features. The preprocessing module then removes irrelevant or label-like columns, handles missing values, and converts categorical attributes into numerical representations. Feature scaling and normalization are applied to ensure that all attributes contribute proportionately during model training.

Once preprocessing is complete, the system trains an unsupervised anomaly detection model on baseline data representing normal behavior. The trained model is then

used to evaluate new or unseen data, generating anomaly scores that indicate the degree of deviation from normal patterns. Based on these scores, events are classified into different risk levels and presented through structured alerts, summary tables, and visual plots. The implementation emphasizes clarity, reproducibility, and explainability to support both academic evaluation and practical cybersecurity analysis.

2.2.1 Implementation and Workflow

The workflow of the system begins with loading system or network telemetry data from CSV or Excel files supplied by the user. If a real dataset is unavailable, the system automatically switches to a demo mode that generates synthetic data resembling real-world cybersecurity scenarios. This ensures uninterrupted operation for testing and presentation purposes.

Following data loading, a comprehensive preprocessing phase is executed. This includes removing irrelevant columns, handling missing or inconsistent values, encoding categorical features, and normalizing numerical attributes. These steps are essential to prepare the data for effective machine learning analysis and to prevent biased or unstable model behavior.

After preprocessing, the anomaly detection model is trained using unsupervised learning techniques on baseline data representing normal system behavior. The trained model then evaluates incoming data points and assigns an anomaly score to each event. Detected anomalies are further analyzed and categorized into risk levels such as LOW, MEDIUM, HIGH, or CRITICAL based on their severity. Finally, the results are presented through alert logs, CSV reports, graphical visualizations, and a real-time SOC-style simulation interface, enabling analysts to quickly understand and respond to potential threats.

The workflow

The workflow of the AI-Driven Zero-Day Threat Indication Anomaly Detection system follows a structured and modular pipeline to ensure accurate detection, interpretability, and safe analysis of anomalous behavior. The system is designed to operate in both real dataset mode and demo simulation mode, allowing flexibility in testing and evaluation.

First, the system loads input data from a CSV or Excel file containing system and network telemetry information. If a real dataset is not available, a synthetic dataset is generated to demonstrate the functionality of the system. This ensures uninterrupted operation and ease of demonstration.

Next, the data preprocessing stage is applied. During this stage, irrelevant or label-like columns are removed to prevent bias. Categorical features are encoded using one-hot encoding, and numerical features are normalized using Min-Max scaling. Missing or invalid values are handled to ensure data consistency and model stability.

After preprocessing, the dataset is divided into two parts. A majority portion of the data is used to train the anomaly detection model, representing normal system behavior. The remaining portion is reserved for detecting anomalous behavior. This baseline-learning approach enables the system to understand what constitutes normal activity

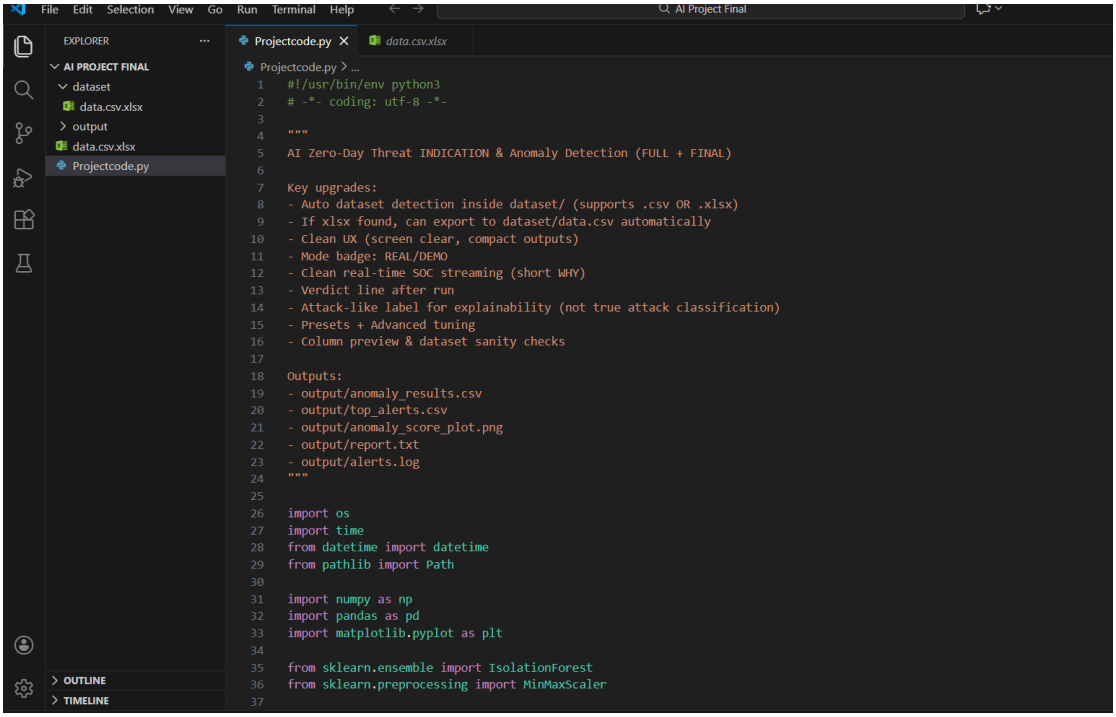

Figure 2.1: Implemenation Code

Figure 2.2: Implemenation Code

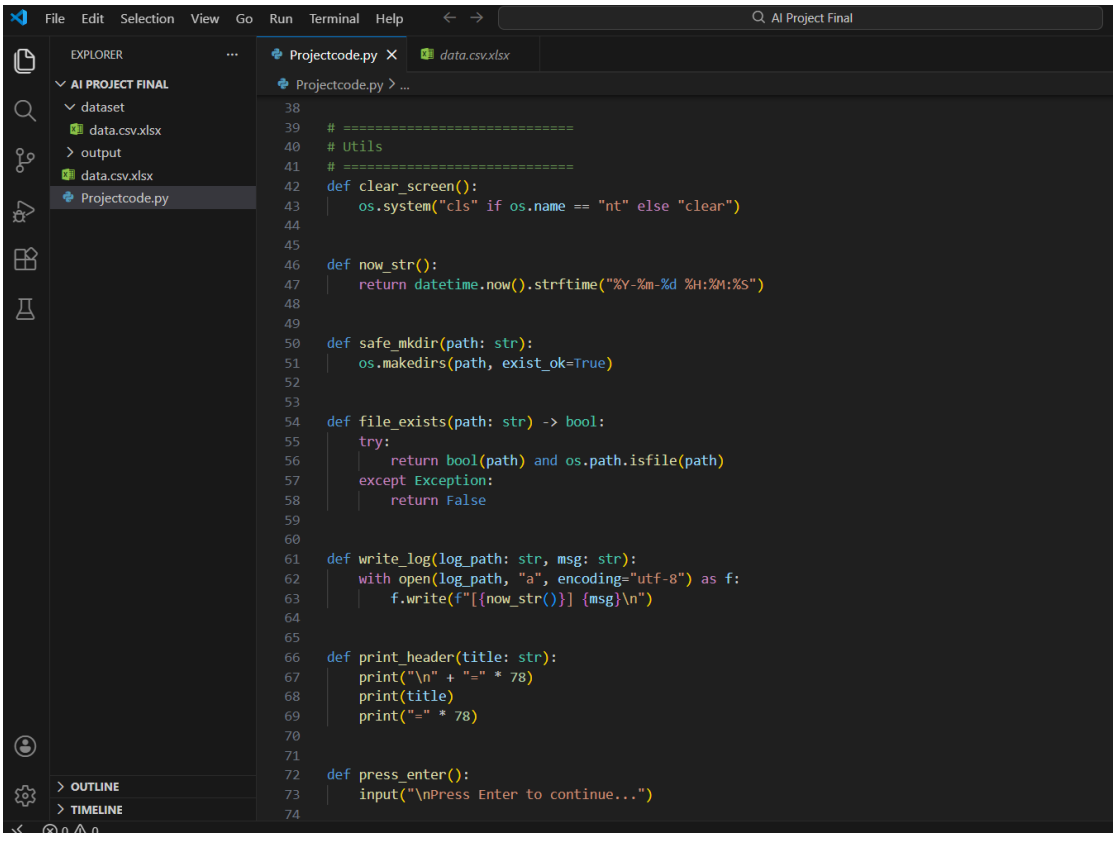

Figure 2.3: Implemenation Code

without requiring labeled attack data.

The trained model then evaluates new or unseen data to compute anomaly scores. These scores represent the degree of deviation from normal behavior. Based on predefined thresholds, detected anomalies are classified into different risk levels such as LOW, MEDIUM, HIGH, and CRITICAL.

To improve interpretability, the system identifies the most influential features responsible for each anomaly. This explainability component provides a concise explanation of why a particular event was flagged, making the alerts analyst-friendly.

Finally, the system generates multiple outputs including anomaly result files, prioritized alert lists, visual plots of anomaly scores, detailed reports, and log files. A real-time SOC-style simulation mode is also provided, where alerts are streamed sequentially to mimic live security monitoring environments.

Tools and libraries

The implementation of the AI-Driven Zero-Day Threat Indication Anomaly Detection system utilizes a combination of programming tools and machine learning libraries to ensure efficiency, scalability, and reliability.

Python is used as the core programming language due to its simplicity, extensive library support, and strong adoption in the field of artificial intelligence and cybersecurity.

NumPy is used for numerical computations and efficient handling of multidimensional data structures.

Pandas is utilized for data loading, preprocessing, feature selection, and structured data manipulation.

Scikit-learn is used to implement the Isolation Forest algorithm for unsupervised anomaly detection, as well as for data preprocessing techniques such as scaling and encoding.

Matplotlib is employed to generate visual representations of anomaly scores, enabling graphical analysis of detected abnormal behavior.

OpenPyXL / CSV handling utilities are used to support both Excel and CSV dataset formats, increasing dataset compatibility.

Logging utilities are used to maintain alert logs and execution traces, supporting auditability and analysis.

Command-line interface (CLI) tools are used to create an interactive, menu-driven user experience that allows users to configure settings, run detection modules, and simulate real-time monitoring.

2.3 Algorithms

The project employs unsupervised anomaly detection algorithms, primarily Isolation Forest. The algorithm isolates anomalous data points by randomly partitioning the fea-

ture space, allowing abnormal behavior to be detected efficiently without labeled data. Risk classification and explainability mechanisms are applied on top of anomaly scores to provide meaningful insights for analysts.

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

Simulation Environment : The simulation environment defines the software and execution conditions under which the proposed AI-Driven Zero-Day Threat Indication Anomaly Detection system was developed, tested, and evaluated.

The system was implemented and executed using the Python programming language, which provides strong support for data analysis and machine learning applications. All experiments were performed on a standard desktop environment to ensure reproducibility and ease of deployment.

Operating System: Windows / Linux based system (Python-supported platform)

Programming Language: Python 3.x

Development Environment: Visual Studio Code (VS Code)

Machine Learning Framework: Scikit-learn

Supporting Libraries: NumPy, Pandas, Matplotlib

Execution Interface: Command Line Interface (CLI) with menu-driven interaction

Dataset Format: CSV and Excel (XLSX) files containing system and network telemetry data

Model Used: Isolation Forest (Unsupervised Machine Learning)

Execution Modes Supported:

Real Dataset Mode

Demo Simulation Mode (synthetic data generation)

The environment was selected to reflect a realistic yet lightweight setup suitable for academic demonstration, experimentation, and real-world adaptability.

Simulation Procedure:

The simulation procedure outlines the step-by-step process followed to evaluate the performance and behavior of the proposed system under controlled conditions.

First, the project script was executed through the command line interface. Upon

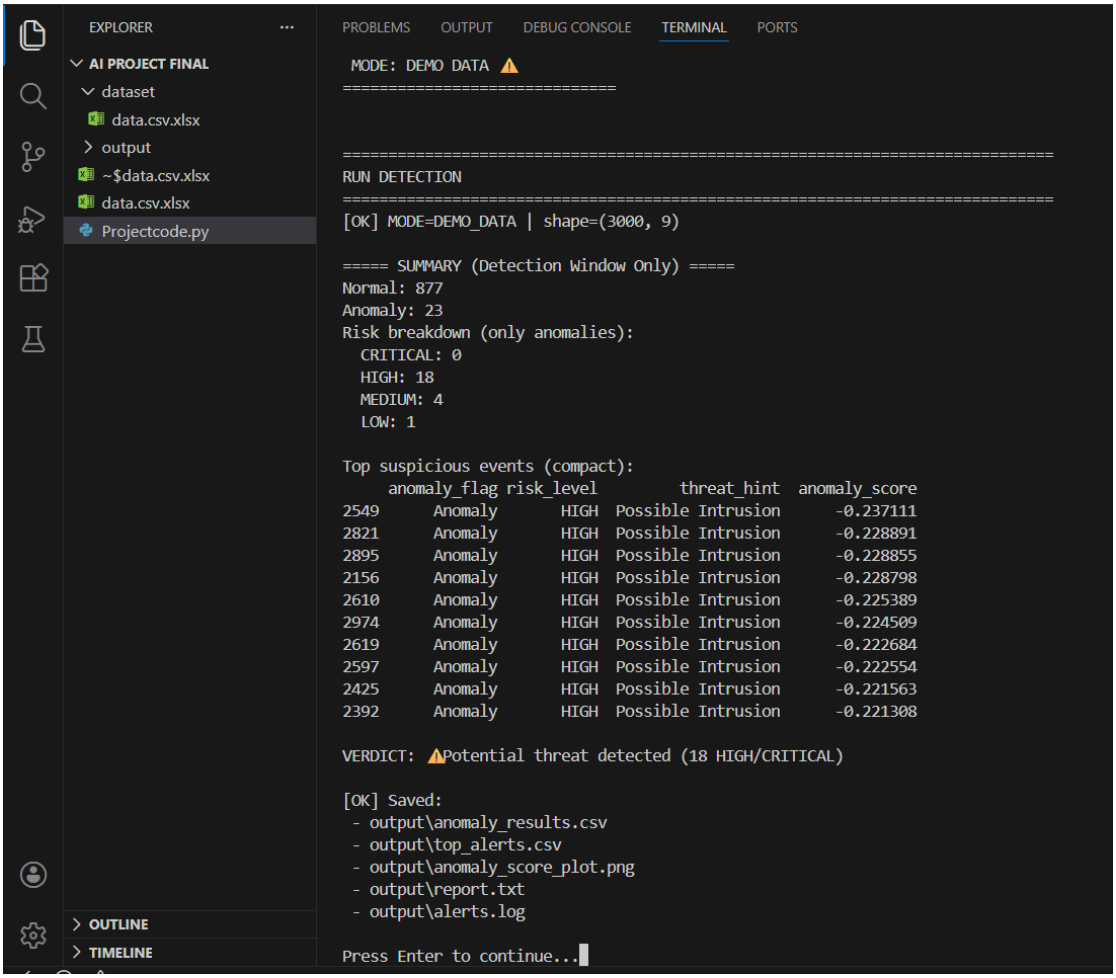

Figure 3.1: Output Code

Figure 3.2: initial 26

execution, the system displayed a menu-driven interface allowing the user to select detection, real-time simulation, dataset configuration, or system settings.

If a real dataset was available, the dataset path was configured using the Dataset Wizard. In cases where no dataset was found, the system automatically switched to demo simulation mode, generating synthetic data to demonstrate functionality.

Next, the input data was preprocessed. This included removing irrelevant columns, encoding categorical features, normalizing numerical values, and handling missing data to ensure consistency and model stability.

After preprocessing, the dataset was divided into two segments. The first segment was used to train the Isolation Forest model to learn normal system behavior. The remaining segment was used to detect anomalous events based on deviations from the learned baseline.

Once detection was completed, anomaly scores were generated for each data instance. These scores were evaluated against predefined thresholds to classify anomalies into risk levels such as LOW, MEDIUM, HIGH, and CRITICAL.

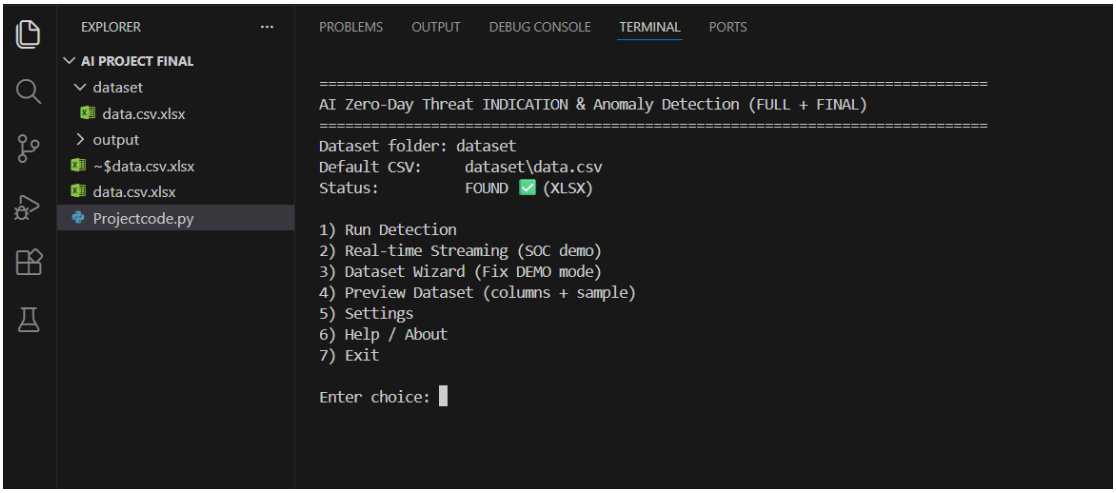

Figure 3.3: Output Code

Figure 3.4: Implemenation Code

To simulate a Security Operations Center (SOC) environment, the real-time simulation mode was executed. In this mode, detected anomalies were streamed sequentially, displaying compact alerts along with short explanations highlighting the key features responsible for each alert.

Finally, the system generated output artifacts including anomaly result files, top alert summaries, graphical plots of anomaly scores, detailed textual reports, and alert logs. These outputs were reviewed to evaluate detection effectiveness, explainability, and system behavior.

3.2 Results Analysis/Testing

The experimental results show that the system successfully detects abnormal behavior patterns within the dataset. Anomalies were correctly identified and categorized into appropriate risk levels. The explainability module highlighted the most influential features contributing to each alert, improving interpretability. The real-time simulation mode effectively demonstrated SOC-style alert streaming.

3.3 Results Overall Discussion

Overall, the experimental results demonstrate that the proposed AI-driven anomaly detection system performs reliably in identifying anomalous behavior that may indicate potential zero-day cyber threats. The system consistently detected deviations from learned normal behavior patterns across both demo and real dataset scenarios. These results validate the effectiveness of unsupervised learning techniques for cybersecurity applications, particularly in environments where labeled attack data or prior signatures are unavailable.

The generated anomaly scores, risk classifications, and alert summaries provide meaningful insights into system behavior and potential security risks. The risk-level categorization further enhances interpretability by prioritizing alerts based on severity, which is essential for real-world security monitoring and incident response workflows. Additionally, the explainable output mechanism highlights contributing features for each anomaly, allowing analysts to understand why an event was flagged. Overall, the results confirm that anomaly-based detection is a practical and scalable approach for early threat indication in modern cyber defense systems.

3.3.1 Complex Engineering Problem Discussion

This project addresses a complex engineering problem that lies at the intersection of cybersecurity, machine learning, and real-time system monitoring. The challenge involves processing high-dimensional system and network telemetry data, learning a stable representation of normal behavior using unsupervised learning, and accurately detecting deviations that may signal unknown or emerging threats. Designing such a system requires careful consideration of data preprocessing, feature normalization, model sensitivity, and detection thresholds.

Another major engineering challenge is managing false positives while maintaining high detection sensitivity. Excessive false alerts can overwhelm analysts, whereas overly strict thresholds may miss critical threats. The system also emphasizes explainability, ensuring that detected anomalies are interpretable rather than opaque model outputs. Furthermore, adaptability to evolving data patterns is crucial, as system behavior and network conditions change over time. By integrating modular design, risk-based alerting, and SOC-style real-time simulation, the project demonstrates a structured solution to a multifaceted and real-world engineering problem.

Chapter 4

Conclusion

4.1 Discussion

This project demonstrates that AI-driven anomaly detection is a highly practical and effective approach for indicating potential zero-day cyber threats in modern computing environments. By learning a baseline of normal system and network behavior, the proposed system is able to identify abnormal deviations that may represent previously unseen or unknown attacks. Unlike traditional signature-based security mechanisms, this approach does not rely on predefined attack patterns, making it especially suitable for detecting zero-day threats.

The use of an unsupervised learning model allows the system to operate even in scenarios where labeled attack data is unavailable, which closely reflects real-world cybersecurity conditions. Additionally, the modular system design enables clear separation between data preprocessing, model training, anomaly detection, risk classification, and alert generation. The inclusion of explainable outputs, which highlight the contributing features behind each alert, enhances trust and usability for security analysts. Overall, the project successfully bridges theoretical AI concepts with practical cybersecurity applications, making it suitable for both academic study and real-world SOC-style monitoring environments.

4.2 Limitations

Despite its effectiveness, the proposed system has certain limitations that must be acknowledged. The system does not predict or label specific attack types, malware families, or threat names; instead, it focuses solely on indicating suspicious behavior through anomaly detection. As a result, human analyst intervention is still required to investigate and confirm the nature of detected threats.

The performance of the system is also dependent on the quality and representativeness of the baseline training data. If the training data contains noise, outliers, or abnormal behavior, the model may learn an inaccurate baseline, which can lead to increased false-positive or false-negative rates. Furthermore, while the real-time simulation provides SOC-style alert streaming, the system does not directly integrate with live network

traffic or automated response mechanisms. Therefore, the project is intended primarily for threat indication and decision support rather than fully autonomous intrusion prevention.

4.3 Scope of Future Work

There are several promising directions for future enhancement of this project. One major improvement would be the integration of multiple anomaly detection models, such as Autoencoders or One-Class SVMs, to form an ensemble-based detection framework that could improve robustness and detection accuracy. Developing a web-based dashboard using tools like Streamlit or Flask would enable interactive visualization of alerts, anomaly scores, and system behavior over time.

Future versions of the system could also incorporate live network traffic capture and system telemetry collection, allowing the model to operate in real-time production environments. Automated PDF report generation and scheduled alert summaries could further support security operations and management reporting. Additionally, integrating the system with SIEM or SOC platforms and extending explainability features would enhance its applicability in enterprise cybersecurity environments. These enhancements would significantly increase the system's scalability, usability, and real-world impact.

References