ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Automatic Classification of Security Alerts in SOC

Tomás Ladeiro Domingues

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Cybersecurity And Systems Administration

Advisor: Jorge Pinto Leite
Supervisor: Gonçalo Amaro

Porto, April 8, 2025

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, April 8, 2025

Dedicatory

To my family and friends.

Abstract

Threats to Cybersecurity have grown ever more sophisticated over the years, making Security Operations Centres (SOC) more important than ever. The topic of this dissertation is the application of Machine Learning (ML) to automate the triage of security alerts, addressing issues such as alert fatigue and false positives. The research proposes leveraging ML models to improve existing Security Information and Event Management (SIEM) systems by classifying alerts, prioritizing actionable threats, and thus enabling fast detection and response times.

Keywords: Cybersecurity, SIEM, QRadar, Machine Learning, Automation, Ticket Triage

Resumo

A crescente complexidade e frequência das ameaças cibernéticas têm destacado a importância dos Centros de Operações de Segurança (SOC) na defesa de organizações contra incidentes de segurança. Este trabalho investiga a aplicação de técnicas de Aprendizagem Automática (ML) para automatizar a triagem de alertas de segurança, reduzindo os desafios enfrentados pelos analistas, como a fatiga causada pelo alto volume de alertas e a alta taxa de falsos positivos. A integração de modelos de ML com sistemas de Gestão de Informações e Eventos de Segurança (SIEM) procura melhorar a eficiência da classificação de alertas, priorizar ameaças críticas e reduzir os tempos médios de detecção e resposta.

Este trabalho inclui uma avaliação dos principais sistemas SIEM, ferramentas de gestão de tickets e frameworks de ML, destacando as suas vantagens e limitações em ambientes SOC.

Palavras-Chave: Cibersegurança, SIEM, QRadar, Aprendizagem Automática, Automação, Triagem de Alertas

Acknowledgement

I would like to express my deepest gratitude to my advisor, Professor Jorge Pinto Leite at ISEP, for his invaluable guidance, support, and dedication throughout this project. His insights and expertise were crucial in shaping this work and guiding me through the challenges of research and development.

I also extend my heartfelt thanks to my supervisor, Gonçalo Amaro, at ArtResilia, for accepting my idea and providing the opportunity to implement it within the organization. His continuous support, advice, and willingness to share his knowledge were essential in bringing this project to fruition.

Their combined mentorship and encouragement were instrumental in the completion of this dissertation, and I am profoundly grateful for their time and efforts.

Contents

List of Figures

List of Tables

List of Acronyms

24/7	twenty-four seven.
AI	Artificial Intelligence.
APT	Advanced Persistent Threats.
CND	Computer Network Defense.
CSIRT	Computer Security Incident Response Team.
DNN	Deep Neural Network.
EDR	Endpoint Detection and Response.
IDS	Intrusion and Detection System.
IPS	Intrusion and Prevension System.
ML	Machine Learning.
RF	Random Forest.
RL	Reinforcement Learning.
SIEM	Security Information and Event Management.
SOC	Security Operations Center.

Chapter 1

Introduction

This chapter provides context for automating alert triage and its significance in enhancing security operations in a SOC, particularly for managing high volumes of security alerts. It begins with defining the problem and detailing the challenges associated with manual alert analysis and the critical requirements for automation. First, the objectives of this thesis are presented; second, this project's expected outcomes, including efficiency and accuracy improvements, are outlined, along with the selected approach for integrating an existing ML model to automate alert classification. Finally, an overview of the report's structure is provided.

1.1 Context

In today's world, cybersecurity has become essential for organizations of all sizes. (Vielberth et al. 2020) The rapid advancement of technology, particularly artificial intelligence, has fostered innovation and growth and equipped cybercriminals with increasingly sophisticated tools and techniques. Cyber threats have surged over recent years, and this trend is expected to continue, with global cybersecurity spending projected to rise from $6 trillion to $30 trillion by 2030. (Sovilj et al. 2020)

As these threats become more complex and frequent, Security Operations Center (SOC) have emerged as the primary line of defense. SOCs are crucial in monitoring, detecting, and responding to cyber threats, ensuring the protection of organizations' core digital assets—data, applications, and infrastructure. By combining human expertise with advanced technology, SOCs proactively defend systems, respond to incidents, and work to minimize the impact of potential breaches, safeguarding essential operations. (Jalalvand et al. 2024)

However, managing the overwhelming number of alerts generated daily within SOCs presents a significant challenge. Analysts must review and respond to thousands of tickets, a process that can quickly lead to burnout, as stated by Tines (2023), especially when the majority of alerts are false positives. A study found that many organizations receive over 10,000 alerts daily, with more than 50% being false positives. (CriticalStart 2019)

One of the stages in the SOC workflow with a higher alert volume is the initial triage process. At this stage, Tier 1 analysts are responsible for gathering raw data, assessing alarms, and determining the urgency and nature of each alert. For every ticket, they must evaluate whether the alert is valid or a false positive, assign priority based on the alert's severity, and identify any potential high-risk incidents. (Vielberth et al. 2020) This process demands significant time and effort, placing a heavy mental load on analysts as they deal with large volumes of information.

Integrating automation in the triage process, particularly for determining alert criticality and classification, could substantially reduce analyst fatigue. Machine Learning (ML) techniques are up-and-coming for processing large data volumes. They facilitate the classification and prioritization of alerts, enabling analysts to focus on high-priority threats. (Jalalvand et al. 2024)

This work explores how SOCs can use automation and ML to streamline alert triage, improve SOC efficiency, and more effectively protect organizations from an ever-evolving cyber threat landscape.

1.2 Problem

The main objective of this work is to develop an independent program that automates the triage of security alerts within a Security Operations Center (SOC) environment. This automation aims to enhance the overall efficiency and accuracy of the triage process, reduce the number of false positives, and alleviate the workload during the initial stages of the SOC workflow.

Improving the speed of analyzing initial alerts is crucial in a Security Operations Center (SOC) workflow. Faster analysis enables SOC analysts to respond to incidents quickly, which helps minimize the time that threats remain undetected. Without automation, this initial alert analysis depends heavily on manual inspection. Analysts face the challenge of sifting through a large volume of alerts to identify potential threats, which is time-consuming and susceptible to human error.

This reactive, manual method increases response times and raises the risk of missing critical threats. Analysts must manually classify alerts, assess their severity, and determine appropriate response actions while managing a continuous influx of new alerts. This approach can strain resources and often leads to alert fatigue, making it difficult for analysts to prioritize and respond effectively.

Key metrics such as Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) can significantly improve by reducing the time required for initial analysis. These metrics are essential for maintaining strong cybersecurity defenses. Automation in alert triage allows for faster and more accurate threat identification, enabling analysts to focus on high-priority threats. This not only increases overall productivity but also reduces fatigue among analysts, while boosting the effectiveness of cybersecurity.

This efficiency results in a more manageable workload, decreased burnout, and greater job satisfaction for employees. For the organization, streamlined response times enhance the security posture, lower the risk of breaches, and ultimately protect critical assets more effectively.

1.3 Objectives

The objectives for this project are as follows:

1. Literature search and review
    - A research into the latest advancements in security analysis and a thorough analysis of the tools, methodologies, and approaches researchers use in SOC automation and machine learning applications.

2. Study of existing Machine Learning (ML) models
    - An analysis of different models for alert triage in SOC environments and related fields, aiming to identify and improve the adopted approach.
3. Collection of security alert datasets
    - A compilation of accessible datasets on security alerts organized to supply training data for the selected ML model.
4. Integration with SIEM and other data sources
    - Connect the SIEM and relevant data sources for real-time data ingestion and analysis.
5. Developing a custom dashboard for analysts to submit feedback on the machine learning model.
    - While the ML model will be trained with existing datasets before deployment, it is still beneficial to provide a method for analysts to continue training the model over time.
6. Testing and evaluation of the selected ML model
    - The selected ML model will be tested with live data, followed by an analysis to assess its effectiveness in categorizing alerts and minimizing false positives.

1.4 Research Contextualization

This research will be conducted at ArtResilia, a cybersecurity firm dedicated to improving organizational cyber resilience through proactive threat anticipation, strong security measures, and efficient recovery. As they emphasize:

> "Everyone will be attacked someday, what matters is how fast they recover!"

This reflects their focus on minimizing the impact of cyber threats and enabling organizations to maintain operational continuity (ArtResilia n.d.).

ArtResilia offers a broad range of cybersecurity services divided into three main areas:

- **Defensive Security:** Design, implementation, and management of security solutions. Their state-of-the-art Security Operations Center (SOC), known as Helix, provides end-to-end security operations, including threat anticipation, detection, protection, and incident recovery.
- **Offensive Security:** Comprehensive testing and validation of organizational assets by simulating real-world threat actor techniques, including penetration testing, threat intelligence, and deception-based strategies.
- **Advisory Services:** Consulting services focused on governance, risk management, compliance (GRC), and security architecture to ensure organizations meet regulatory and operational security requirements.

ArtResilia was approached for a research project exploring strategies to enhance cybersecurity resilience through automation and machine learning. The project aligns with ArtResilia's mission to strengthen incident detection and response capabilities against emerging cyber threats.

After reviewing the proposal, ArtResilia accepted the project and committed to providing support and resources for its implementation. This collaboration underscores ArtResilia's dedication to fostering cybersecurity innovation and ensuring real-world applications in professional security operations.

1.5 Research Questions

For this research, three key research questions (RQs) were devised to address the central issue of improving the efficiency and accuracy of security alert triage within Security Operations Centers (SOCs). Each question offers a targeted perspective for examining specific facets of the problem, contributing to the overarching objective of minimizing both false positives and false negatives in security alerts. Below are the RQs, along with a brief discussion of their significance.

1. **How can ML techniques be applied to the triage of security alerts to reduce false positives and negatives?**
   This question explores how machine learning (ML) can tackle the challenge of excessive alerts in SOCs, characterized by high false positive and negative rates. The goal is to identify effective ML techniques and assess their impact on improving alert accuracy.
2. **What are the main challenges in implementing an AI bot integrated with SIEM systems, and how can they be mitigated?**
   Implementing AI in security systems presents a range of challenges, from broader issues like data integration, system compatibility, and performance maintenance under varying conditions, to narrower challenges such as trusting the AI to serve as an advisor to the manual work of the SOC analyst. It's also crucial to ensure that the percentage of incorrect classifications does not jeopardize trust in the solution. This research question aims to identify these challenges and propose strategies for successful AI integration.
3. **What ML frameworks and methodologies exist, and which are most suitable for developing and training AI models for ticket triage in SOC environments?**
   Understanding ML frameworks and methodologies is essential for choosing the right tools for AI model development in SOC ticket triage. This question examines the suitability of different frameworks based on their capabilities and alignment with SOC needs.

To answer these research questions systematically and comprehensively, a structured approach was adopted. This approach involved defining keywords, utilizing relevant digital libraries, and employing robust frameworks for the literature review.

The following keywords and their combinations were used to identify relevant papers:

- **Keywords:** Machine learning, AI, security alerts, SOCs, SIEM systems
- **Keywords:** False positives, false negatives, alert triage, alert fatigue
- **Keywords:** AI integration, ticket prioritization, frameworks, methodologies

The primary digital libraries used were:

- **ACM Digital Library**: For accessing peer-reviewed articles on machine learning and security applications.

- **ResearchGate**: For exploring academic discussions, preprints, and supplementary materials.

Search strings using these keywords were used to search the digital libraries effectively.

- **Search String 1:** ("Machine learning" AND "security alerts") AND ("false positives" OR "false negatives")
- **Search String 2:** ("AI integration" AND "SIEM systems") OR ("ticket prioritization")
- **Search String 3:** ("Frameworks" AND "SOC environments") AND ("alert triage")

In addition to the search strings, a new constraint was implemented to limit the findings to the most recent five years, ensuring the collection of up-to-date research and information.

To structure my literature review, two complementary frameworks were used: **PICOCS** and **snowballing**.

- **PICOCS Framework**: This framework helped define the Population, Intervention, Comparison, Outcome, Context, and Study type for each RQ as can be seen on Figure 1.1. Using PICOCS ensured a focused and systematic approach to identifying papers aligned with this research objectives.

Table 1.1: PICOCS Framework for Research Questions

Element	Definition	RQ1	RQ2	RQ3
P	Population	SOC analysts dealing with alerts	SOC analysts and SIEM administrators	SOC analysts developing AI models
I	Intervention	Machine learning techniques	AI bots integrated with SIEM systems	ML frameworks and methodologies
C	Comparison	Manual triage methods	Existing rule-based systems	Alternative ML frameworks
O	Outcome	Reduction in false positives/negatives	Mitigation of integration challenges	Identification of suitable frameworks
C	Context	SOC environments	SOC environments using SIEM systems	SOC environments implementing AI models
S	Study Type	Empirical studies, experiments	Case studies, technical reports	Framework evaluations, experiments

- **Snowballing Method**: Starting with an initial set of highly relevant papers, I used snowballing to identify additional studies by exploring their references (backward snowballing) and citations (forward snowballing). This iterative process expanded the breadth of my review.

Combining PICOCS and Snowballing The combination of PICOCS and snowballing provided a balance between structure and adaptability. PICOCS ensured a methodical and targeted search, while snowballing allowed for iterative exploration beyond initial search results, capturing emerging trends and overlooked studies.

By following this approach, it was possible to systematically answered the RQs and developed insights that directly contribute to improving the triage of security alerts in SOC environments.

1.6 Dissertation Structure

[Structure]

Chapter 2

State of the Art

This chapter is divided into three parts. Part I starts on the theoretical side by introducing Machine Learning, Security Operation Centers, Security Information and Event Managers, and response mechanisms such as Intrusion and Prevension System (IPS), Intrusion and Detection System (IDS), and Endpoint Detection and Response (EDR). It also explains how popular ML classification models work and how to evaluate them. Understanding these topics is the groundwork for material considered later in the chapter.

The second part focuses on the automatic classification of security alert tickets. It begins with an overview of proposed solutions and then reviews several publicly available studies. The chapter concludes with a detailed evaluation of existing systems that are similar to the one presented in this thesis.

The third and final part discusses existing technologies that SOC may utilize. These technologies are designed to support the workflow outlined in the first section about SOC. At least three different technologies will be presented and discussed for each aspect of the SOC workflow. The section will conclude with a summary of the key differences among the technologies and, if relevant, evaluate which one may be superior.

2.1 Theoretical Introduction

Before diving into the technologies, tools, and methodologies used for automating the classification of security alert tickets, it is essential to understand some fundamental concepts. Concepts like, SOCs, Security Information and Event Management (SIEM) systems, Artificial Intelligence (AI), particularly ML, and advanced threat detection and response mechanisms. Such a foundation is indispensable for comprehending these systems' functions, construction work, advantages and limitations, and the actual performance comparison among different methods. The chapter introduces SOC and how it functions, then moves on to the foundations of ML and provides an overview of models frequently used for classification problems. It provides an overview of methods for assessing the performance of classification models, placing this in context for their use within the project.

2.1.1 Security Operations Center

The complexity and frequency of cyber threats are increasing (Arianna 2024), which has led to the emergence of SOCs as a critical component of modern IT enterprises. SOCs are the primary defenders in incident response planning, vulnerability management, and regulatory compliance. In today's interconnected world, integrating security operations to reduce

defensive barriers allows organizations to optimize resources, enhance security posture, and safeguard critical assets.

A SOC is a unit (Rutledge 2024) that provides tailored and centralized Computer Network Defense (CND) (Zimmerman 2014). It defends computer networks against the growing world of cyber threats. The main objective of a SOC is to ensure continuous monitoring and incident response for enterprise systems (Zimmerman 2014). This primarily focuses on preventing unauthorized access, cyber-attacks, and data breaches.

This twenty-four seven (24/7) facility leverages advanced technologies and skilled information security professionals to monitor the network continuously (Zimmerman 2014). With sophisticated tools to detect anomalies, a SOC can address threats before they escalate.

Definition and Characteristics of a SOC

Zimmerman (2014) defined a SOC as:

> "A team primarily composed of security analysts organized to detect, analyze, respond to, report on, and prevent cybersecurity incidents."

This definition integrates elements from various sources, including the historical definition of Computer Security Incident Response Team (CSIRT) as detailed in references (Shirey 2007) and (Brownlee and Guttman 1998).

For an organization to qualify as a SOC, according to Zimmerman (2014), it must:

1. Establish a system for constituents to report cybersecurity incidents.
2. Provide comprehensive support for managing and resolving incidents effectively.
3. Convey incident-related information to internal and external stakeholders.

Key Responsibilities of a SOC

A SOC has several critical missions, as outlined by various sources (Muniz, McIntyre, and AlFardan 2015; Zimmerman 2014):

- Preventing cybersecurity incidents by implementing proactive measures such as vulnerability scanning and threat analysis.
- Monitor, detect, and analyze potential security intrusions.
- Handle confirmed incidents and coordinate resources for effective countermeasures.
- Providing stakeholders with situational awareness regarding cybersecurity incidents, trends, and threats.

Tiers of Operation

Analysts within a SOC operate in tiers (Vielberth et al. 2020). Tier 1 analysts monitor and conduct initial investigations, escalating complex cases to Tier 2 analysts, who perform in-depth analyses and take further actions like blocking activities or deactivating accounts. Generally, higher-tier analysts handle more complex incidents, which require more time to resolve.

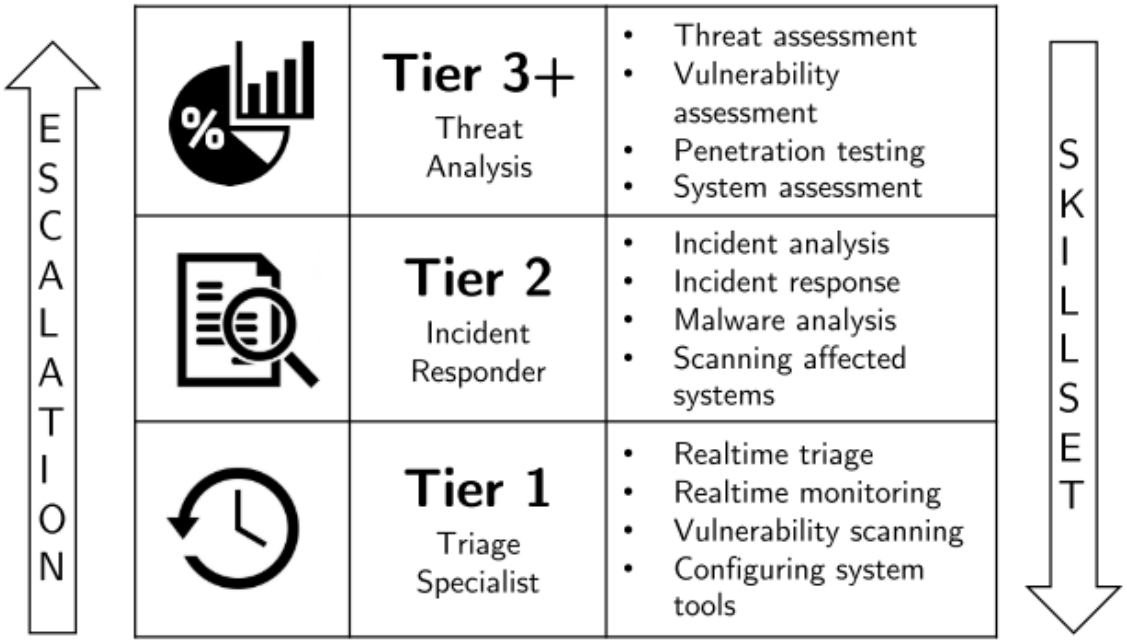

Figure 2.1: SOC Analyst Tier Responsibilities, from (Kokulu et al. 2019).

Extra levels may exist that handle responsibilities like threat hunting, vulnerability assessments, and penetration testing. These levels are collectively termed Tier 3+. Figure 2.1 was taken from a paper on a qualitative study on a SOC (Kokulu et al. 2019) and illustrates a visual representation of these tiers and their associated tasks. Not all SOCs follow a hierarchical model. In some collaborative frameworks, team members may possess comparable skill sets, enabling them to manage incidents independently (Kokulu et al. 2019).

Triage Specialist

Since this project focuses on automating the triage process, it is crucial to gain a detailed understanding of the role of a triage specialist, their functions, and the advantages of automated triage in comparison.

Tier 1 analysts, also known as triage specialists, play a critical role in the initial stages of a SOC workflow (Vielberth et al. 2020).

As stated by Vielberth et al. (2020), a Tier 1 analyst's primary responsibilities include:

1. Collecting and analyzing raw data.
2. Reviewing alarms and alerts generated by monitoring systems.
3. Determining the validity and criticality of each alert.

They must improve alerts with additional contextual information and decide whether an alert represents a real threat or a false positive (Hámornik and Krasznay 2018; Sundaramurthy et al. 2014). This process demands meticulous attention to detail, as the triage specialist must assess individual alerts, notify potential high-risk events, and prioritize them according to their severity (Tao 2018).

The repetitive nature of triage work, coupled with the need to escalate unresolved issues to Tier 2 analysts, can result in mental fatigue and burnout (Iamnitchi et al. 2017; Tines 2023).

This exhaustion affects individual performance and can compromise the overall efficiency of the SOC, as delayed or missed alerts may result in critical threats going undetected (CriticalStart 2019).

In a 2018 study (Crowley and Pescatore 2018), 53% of respondents in a security survey identified inadequate automation as the most common shortcoming.

This study demonstrates that effectively structured and implemented automation can help mitigate some or many of a SOC's weaknesses, particularly in the repetitive aspects of the SOC's workflow, such as the triage process.

2.1.2 Security Information and Event Management

The increasing complexity of cybersecurity threats has compelled organizations to implement advanced technologies to protect their digital assets. SIEM systems have emerged as vital tools in this context (Shaw 2022).

SIEM systems gather and centralize security-related data to detect threats and respond to incidents effectively. These systems connect logs from various sources to support security analytics, enabling real-time monitoring and retrospective analysis of past events (Shaw 2022). They integrate with cyber threat intelligence platforms, providing human analysts with advanced visual tools for seamless information sharing between organizations. Additionally, they retain event data over extended periods, ensuring robust log management capabilities.

The key features of a SIEM system, as gathered from various published sources (Ali, Shah, and ElAffendi 2024; Harper et al. 2010; Sheeraz et al. 2023), are:

- **Log Collection:** SIEM gathers log data from various network devices such as servers, firewalls, and switches. Data can be collected using two methods:
    1. **Agent-based collection:** An intermediary agent collects and forwards logs.
    2. **Agent-less collection:** Servers retrieve logs directly from the source devices.
- **Log Aggregation:** Collected logs are analyzed and structured for meaningful insights. Aggregation methods include:
    1. **Push method:** Devices actively send logs to the SIEM.
    2. **Pull method:** SIEM retrieves logs as needed.
- **Parsing and Normalization:** Parsing converts raw logs into structured data, while normalization standardizes logs from diverse sources to eliminate redundancy.
- **Threat Analysis and Detection:** By correlating log data with known threat indicators, SIEM systems identify malicious activities. Statistical methods and predefined rules enhance their ability to detect sophisticated threats.
- **Response Automation:** SIEM systems issue real-time alerts and notifications, enabling rapid responses to potential incidents.
- **Reporting and Visualization:** Advanced reporting tools provide security analysts with actionable insights, enabling detailed investigations and trend analysis.

Architectural Components

The architectural components of a security information and Event Management (SIEM) system consist of several essential elements that enable effective security monitoring, incident detection, and response (Sheeraz et al. 2023).

Data sources provide the raw material for analysis and threat detection (Ali, Shah, and ElAffendi 2024). These include a wide variety of log-generating devices and applications:

- **Network devices**: Firewalls, routers, and switches.
- **Endpoint devices**: Workstations, servers, and mobile devices.
- **Applications**: Web servers, databases, and cloud platforms.

A variety of data sources is crucial for effective monitoring and threat detection.

Data collection is a vital step with two main approaches to consider:

- **Agent-based collection**: Agent-based collection uses proxy agents on endpoint devices for better control and flexibility in log collection, but it is costly and complex to manage.
- **Agent-less collection**: Agent-less collection allows devices to send data directly to the SIEM, simplifying deployment but may reduce efficiency in high-volume data environments.

The **SIEM processing engine** is one of the critical components (Sheeraz et al. 2023) responsible for:

- **Parsing**: The conversion of raw log data into a structured format used for analysis.
- **Normalization**: Ensure the log formats are standardized to facilitate easier comparisons.
- **Correlation**: Finding relationships between events to strengthen security posture or incident response.

Storage and rationalization are vital for ensuring scalability and compliance. SIEM systems must store logs for future analysis. This means there has to be enough storage and a logical way of organizing this data for scalability or compliance requirements like GDPR and HIPAA (Sheeraz et al. 2023). Effective storage solutions should scale dynamically to handle large datasets without compromising performance or compliance standards.

Visualization and reporting tools play a key role in making data accessible and actionable. Critical functions that significantly benefit from good data visualization and reporting tools are incident investigation, trend analysis, and compliance audits (Sheeraz et al. 2023). These features not only improve user experience but also aid in making data actionable and accessible. Organizations can use these tools to conduct incident investigations, identify patterns occurring over time, and monitor compliance (Sheeraz et al. 2023). This comprehensive approach enables the development of scalable systems that can effectively handle increasing data demands.

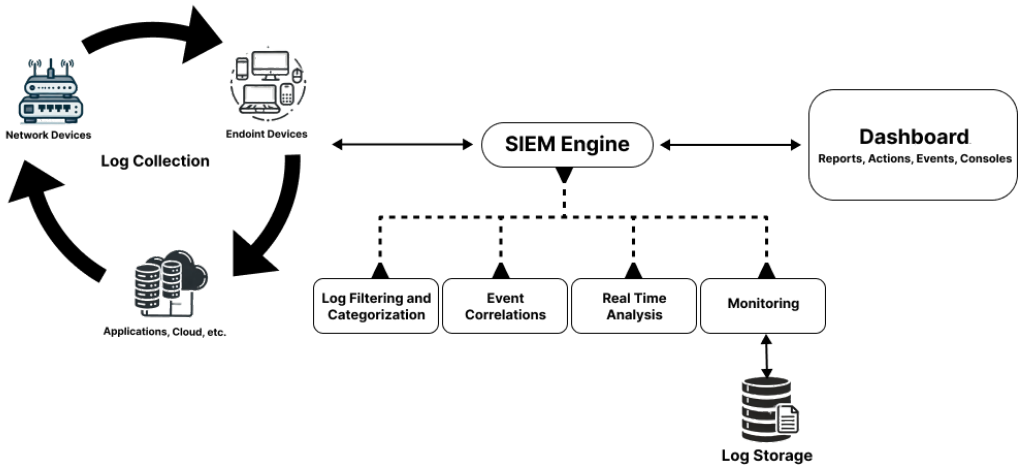

Figure 2.2: SIEM Architecture

Figure 2.2 illustrates the architecture of a SIEM system, highlighting its critical components and their interactions. Data flows from network and endpoint devices, as well as applications, to the log collection module. The SIEM processing engine then parses, normalizes, and correlates this data for real-time security alert analysis, event monitoring, and threat detection.

The diagram also emphasizes the importance of visualization and reporting tools for turning data into actionable insights.

The graphic illustrates how these elements integrate into a SIEM.

2.1.3 Machine Learning

ML is a core part of AI but also overlaps with data mining, statistics, probability, and mathematics (Mohri, Rostamizadeh, and Talwalkar 2012). Unlike traditional rule-based systems that rely on predefined logic, ML uses induction—it learns patterns from past data and forms assumptions that can be generalized to new cases (Ali, Shah, and ElAffendi 2024). This method relies on datasets, which are groups of examples the ML algorithm analyzes to find patterns (Mohri, Rostamizadeh, and Talwalkar 2012; Suthaharan 2016). The goal is to use these learned patterns to predict or describe new data.

There are three primary types of techniques employed in machine learning, (Mohri, Rostamizadeh, and Talwalkar 2012):

1. **Reinforcement Learning**: This is a subset of machine learning in which an agent learns by interacting with the environment (Moradi, Acker, and Denil 2023). It observes the environment, selects an action, and receives a reward if the action is beneficial or a penalty if it is detrimental. Over time, it refines its approach to achieve maximum rewards.

2. **Supervised Learning**: In this type of machine learning, models are trained on labeled data, meaning that each example includes input features and the corresponding expected output (or label). The model learns to map the inputs to the outputs, enabling it to predict the output for new, unseen data.

3. **Unsupervised Learning**: Unsupervised learning analyzes unlabeled data to reveal patterns without predefined labels. Unlike supervised learning, it allows algorithms to explore data independently. It is often used for clustering similar data points or modeling probability distributions. This approach is valuable for understanding the inherent organization in data without prior knowledge.

Supervised learning can be further categorized into several types, (Mohri, Rostamizadeh, and Talwalkar 2012):

- **Regression**: Regression algorithms predict numerical values within a continuous range by analyzing input data to identify patterns. They can forecast future values, such as calculating the next number in a sequence based on previous numbers and trends. Techniques like linear regression, polynomial regression, and others enable these algorithms to draw conclusions and make predictions effectively.

- **Similarity**: Similarity algorithms analyze and compare two distinct instances to measure their resemblance. They are vital in recommender systems for suggesting products based on user preferences and in visual identity tracking and verification by comparing images or features. Their versatility makes them essential in data analysis, security, and personalized user experiences.

- **Classification**: Classification algorithms classify the input data into predefined groups. Classification tasks can be binary when there are only two possible categories (such as a yes-no decision) or multiclass when the number of categories exceeds two, such as recognizing handwritten letters in the alphabet.

However, ML has its limitations. Since datasets are finite, no algorithm can predict every scenario, which highlights an essential aspect of inductive reasoning: it can suggest likely outcomes but cannot ensure certainty (Mohri, Rostamizadeh, and Talwalkar 2012; Suthaharan 2016).

In ML, achieving an optimal model involves balancing between two critical concepts: **bias** and **variance**. Bias refers to the error introduced when a model is excessively simplistic, which can lead to underfitting. Underfitting occurs when the model fails to capture the underlying patterns of the data, resulting in poor performance on both training and unseen datasets (ElSahly and Abdelfatah 2023). On the other hand, variance arises when a model becomes overly complex and sensitive to the fluctuations in the training data, culminating in overfitting. Overfitting means the model performs exceptionally well on the training dataset but poorly on new, unseen data because it has memorized the noise instead of learning the actual signal (ElSahly and Abdelfatah 2023).

The primary objective in constructing a machine learning model is to identify the appropriate level of complexity with the right balance, ensuring the model generalizes well and performs effectively on new data (Suthaharan 2016).

Despite these challenges, ML continues to evolve, with increasingly sophisticated algorithms enabling breakthroughs in many fields (Mohri, Rostamizadeh, and Talwalkar 2012).

2.1.4 Machine Learning Models

The processes and computations of training a machine learning model are structured within a framework. This chapter will briefly overview the most frequently used algorithms relevant to this project's goals.

Decision Trees

Decision Trees (DTs) are a fundamental supervised machine learning algorithm for classification and regression tasks (Huang 2024). They work by splitting data into subsets based on the value of input features, forming a tree-like structure composed of nodes and branches.

The process begins at the root node, representing the entire dataset, and iteratively divides the data into homogeneous subsets using decision nodes until a terminal leaf node is reached, representing the output prediction or class (Chauhan 2022), as seen in Figure 2.3.

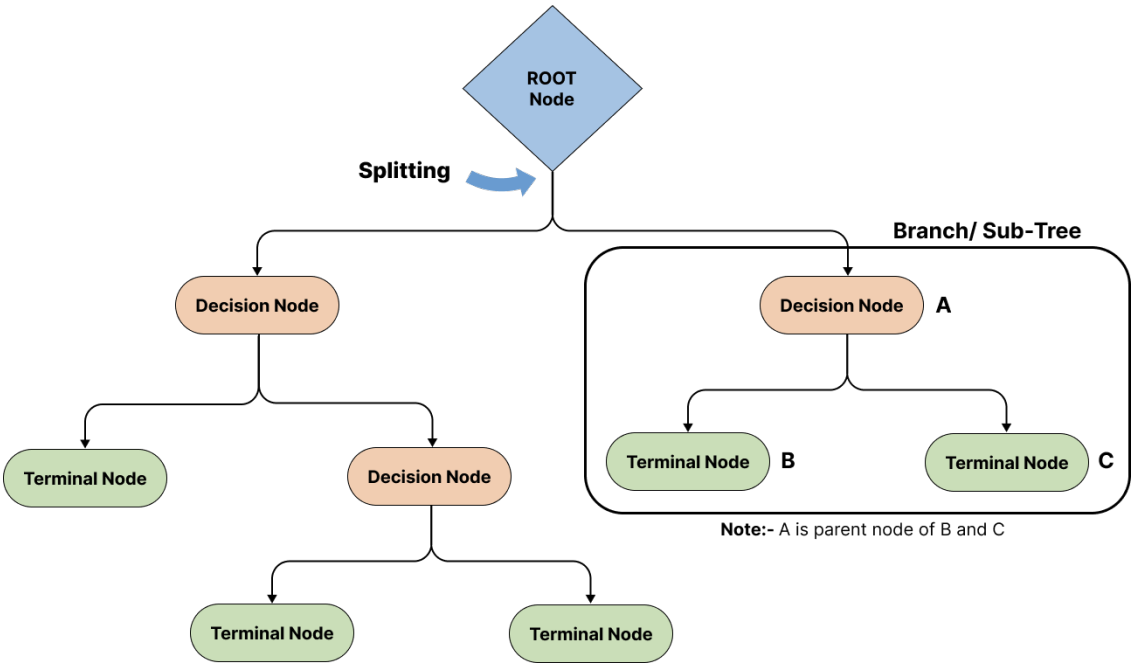

Figure 2.3: A basic structure of a Decision Tree, based on (Chauhan 2022)

The algorithm operates by evaluating all potential splits and selecting the one that optimizes a specific criterion, such as information gain or Gini impurity. Information gain measures the reduction in entropy after a split, while Gini impurity quantifies the likelihood of misclassification at a node.

The entropy of S is defined as:

$$H(S) = \sum_{i=1}^{n} -P(s_i) \times \log_b P(s_i) \tag{2.1}$$

where S is a set of values $s_1, s_2, \ldots, s_n$, $P(s_i)$ is the probability of observing a certain value, and b is the logarithm base, most commonly 2, e, or 10.

Using entropy, the **information gain (IG)** for a node t and a candidate split x is calculated as:

$$IG(t, x) = H(t) - H(x, t) \tag{2.2}$$

In contrast, the **Gini index**, another commonly used metric for evaluating splits, is defined as:

$$\text{Gini} = 1 - \sum_{i=1}^{n} P(s_i). \tag{2.3}$$

At each decision node, the algorithm tests a single feature and branches according to its value, guiding data instances down the tree until they reach a leaf node (Chauhan 2022). During training, the algorithm continues splitting until a stopping criterion is met, such as achieving a maximum tree depth, minimum node size, or no further improvement in the splitting metric (Chauhan 2022).

Pruning techniques prevent overfitting, which occurs when a tree becomes too complex and overly specific to the training data. These involve removing unnecessary branches or nodes to simplify the tree while maintaining predictive accuracy. Pruning can be preemptive (stopping tree growth early) or post hoc (removing branches after the tree is fully grown). This ensures that the decision tree generalizes well to unseen data (Huang 2024).

Decision trees are non-parametric, meaning they do not assume any specific distribution for the data, and they can capture both linear and non-linear relationships (Huang 2024). However, they are sensitive to slight variations in the dataset (Huang 2024), which may cause significant changes in the tree structure. Despite this, they remain popular for their simplicity, interpretability, and ability to handle numerical and categorical data (Chauhan 2022).

Ensemble classifiers

Ensemble learning is a powerful machine learning technique that combines the predictions of multiple base models to achieve superior performance compared to individual learners (Joseph 2022).

The fundamental idea is to address the limitations of single models, such as high variance, high bias, or low accuracy, by leveraging the diversity and complementary strengths of multiple models (Dasarathy and Sheela 1979; Hansen and Salamon 1990; Mienye and Sun 2022).

The concept of ensemble learning has evolved significantly since its inception. Early work by Dasarathy and Sheela (1979) introduced the partitioning of feature spaces using multiple classifiers. Later, Hansen and Salamon (1990) demonstrated that ensembles of artificial neural networks achieved superior predictive performance compared to single networks. Schapire (1990) groundbreaking work laid the foundation for boosting, one of the primary techniques in ensemble learning.

Ensemble learning methods are evaluated on two main principles: **accuracy** and **diversity** of the base learners (Hansen and Salamon 1990; Mienye and Sun 2022).

- **Accuracy**: Each base learner should perform better than random guessing on the given task (Li, Yu, and Zhou 2012).

- **Diversity**: The errors made by individual learners should be uncorrelated, which can be achieved through techniques like subsampling, feature randomization, or algorithmic diversity (Breiman 1996; Schapire 1990).

A model is accurate if it generalizes well on unseen instances, while diversity ensures that the errors of individual base models are not correlated. Achieving this balance is crucial for effective ensembles.

Ensemble Classifiers offer several advantages over single models, making them a popular choice in diverse applications of machine learning:

- **Improved Generalization**: Ensembles reduce overfitting and improve generalization by aggregating the predictions of multiple learners (Hansen and Salamon 1990).
- **Bias-Variance Trade-off**: Techniques like bagging reduce variance while boosting minimizes bias, addressing the critical limitations of individual learners (Li, Yu, and Zhou 2012; Mienye and Sun 2022).
- **Adaptability**: Ensembles can be designed to work with homogeneous (same algorithm) or heterogeneous (different algorithms) base learners, making them versatile across domains (Mienye and Sun 2022).

Ensemble learning methods are broadly categorized into:

- **Parallel Ensembles**: Base learners are trained independently on different subsets of data or features. Techniques like bagging and Random Forests fall into this category (Breiman 1996). The illustration in Figure 2.4 demonstrates how parallel ensembles operate.
- **Sequential Ensembles**: Models are trained iteratively, with each learner focusing on correcting the errors of its predecessor. Boosting is the most prominent example of this approach (Schapire 1990). The illustration in Figure 2.5 demonstrates how sequential ensembles operate.
- **Stacked Ensembles**: Predictions from multiple base models are combined using a meta-model trained on the outputs of the base learners (Mienye and Sun 2022).

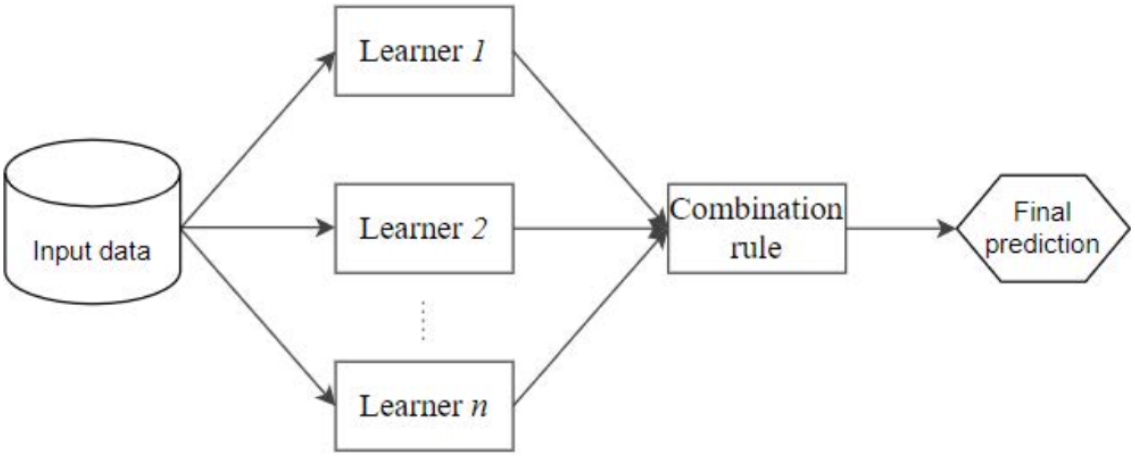

Figure 2.4: Diagram of Parallel Ensemble Learning from (Mienye and Sun 2022)

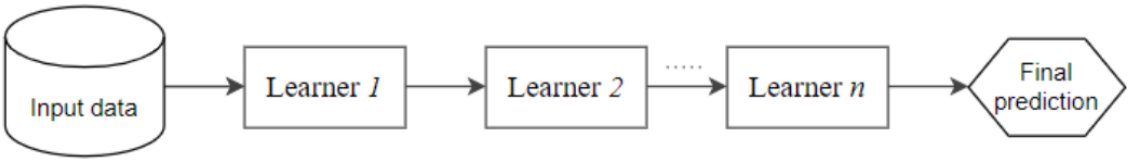

Figure 2.5: Diagram of Sequential Ensemble Learning. from (Mienye and Sun 2022)

Ensemble methods can then be categorized into three main approaches based on how they train and combine base models: **Bagging**, **Boosting**, and **Stacking**. Each method has a distinct mechanism for generating diversity and aggregating predictions, addressing the specific limitations of single models.

Bagging Bagging, which stands for bootstrap aggregating, is a parallel ensemble technique where multiple base models are independently trained on random subsets of the training data, referred to as bootstrapped samples. The final predictions are made by aggregating the outputs of these models, often using majority voting or averaging. Random Forests are a well-known implementation of bagging, recognized for their ability to reduce variance without increasing bias (Mienye and Sun 2022).

Boosting Boosting is a sequential ensemble method that aims to reduce bias by iteratively training base models. Each subsequent model focuses on correcting the errors made by the previous one, assigning higher weights to misclassified samples. Algorithms such as AdaBoost and Gradient Boosting exemplify this approach, often achieving superior performance on complex datasets, but this can lead to increased susceptibility to overfitting (Mienye and Sun 2022).

Stacking Stacking integrates predictions from multiple base models using a meta-model that learns the optimal way to combine their outputs. Unlike bagging and boosting, stacking allows for heterogeneous base learners, providing greater flexibility and diversity. The meta-model is typically trained on the predictions made by the base models, enabling it to make more accurate decisions (Mienye and Sun 2022).

Random Forests

Random Forests represent a significant advancement in machine learning, particularly in the domain of ensemble classifiers (Ali, Shah, and ElAffendi 2024).

As a bagging-based ensemble method, Random Forests combine multiple decision trees to improve classification accuracy and robustness. This approach leverages the strengths of individual trees while mitigating their limitations, such as overfitting, by aggregating their predictions (Ali, Shah, and ElAffendi 2024).

The Random Forest algorithm begins by generating multiple decision trees, as can be seen on Figure 2.6, each trained on a random subset of the training data through bootstrapping[1].

[1]Bootstrapping is a resampling technique where random samples are drawn with replacement from the dataset.

Additionally, features are randomly sampled at each split to ensure diversity among the trees, a process that reduces the correlation between individual tree predictions (Farooq and Otaibi 2018). Once trained, the predictions of all trees are aggregated, typically through majority voting for classification tasks or averaging for regression tasks, to produce the final output (Nila, Apostol, and Patriciu 2020).

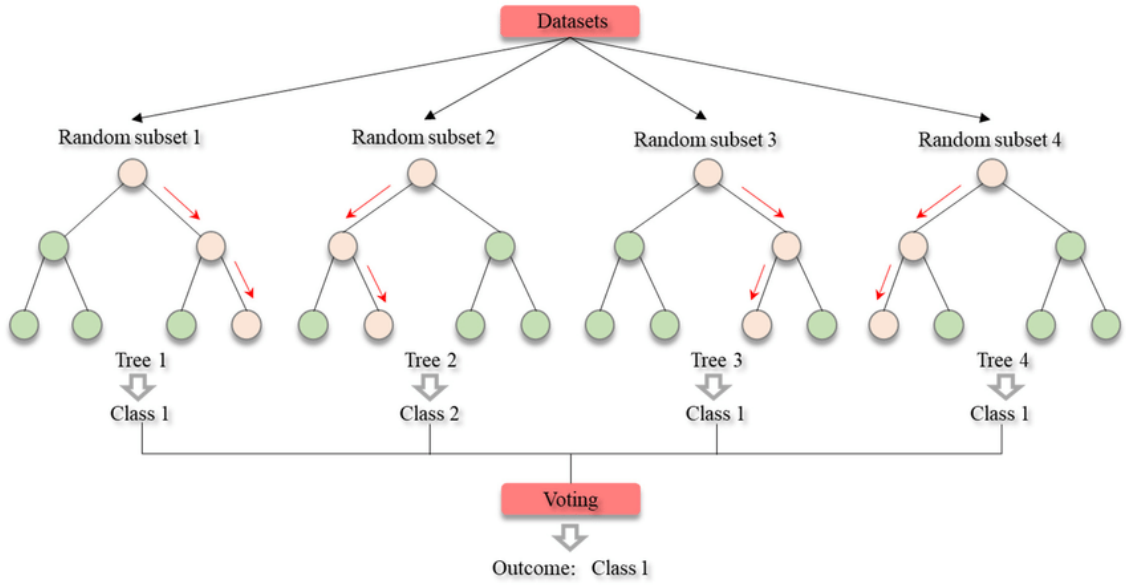

Figure 2.6: Workflow of Random Forests: Bootstrapping, Decision Trees, and Aggregated Predictions from (Yang et al. 2019)

Random Forests are highly valued for their versatility and robustness. They perform exceptionally well with large and high-dimensional datasets, effectively handling numerical and categorical data. Furthermore, their inherent ability to measure feature importance makes them interpretable, a critical requirement in security domains where trust and explanation are essential (Ali, Shah, and ElAffendi 2024).

Random Forests also excel at reducing overfitting, a common issue with individual decision trees. By averaging the predictions of multiple trees, the model achieves better generalization, even when faced with noisy data (Sopan et al. 2019).

Random Forests have been widely adopted in cybersecurity applications, particularly for detecting anomalies, classifying alerts, and predicting the likelihood of malicious behavior. For example, they are effectively used in SIEM systems to process large volumes of log data, identify patterns, and reduce false positives (Ali, Shah, and ElAffendi 2024; Farooq and Otaibi 2018).

Studies have shown that Random Forest-based models can achieve high accuracy in detecting advanced persistent threats (APTs) and other cyberattacks by leveraging their ability to handle complex feature interactions and high-dimensional data (Ali, Shah, and ElAffendi 2024). For instance, combining Random Forests with additional modules for feature selection and preprocessing has improved detection rates and reduced manual intervention in incident response workflows (Nila, Apostol, and Patriciu 2020).

Naive Bayes

The Naive Bayes algorithm is a probabilistic classifier based on Bayes' theorem, which assumes conditional independence among features given the class (Chandra, Challa, and Pasupuleti 2016). This simplicity makes it computationally efficient and easy to implement.

Naive Bayes is particularly effective in scenarios where the assumption of feature independence is approximately attained, such as spam detection and document classification. The model's reliance on this assumption allows it to scale effectively to high-dimensional datasets while remaining computationally efficient (Chandra, Challa, and Pasupuleti 2016).

Naive Bayes calculates the posterior probability of each class given the observed data using the formula:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \tag{2.4}$$

where:

- $P(C|X)$ is the posterior probability of class C given the feature vector X,
- $P(X|C)$ is the likelihood of observing X given class C,
- $P(C)$ is the prior probability of class C,
- $P(X)$ is the probability of the feature vector X.

In practice, the Naive Bayes model, Figure 2.7, estimates the prior probability $P(C)$ for each class and the conditional probability $P(X_i|C)$ for each feature X_i given the class. These probabilities are typically derived from the frequency of occurrences in the training data. For continuous features, it is common to assume a Gaussian distribution and compute the likelihood accordingly. The model assigns a new instance to the class with the highest posterior probability $P(C|X)$ (Chandra, Challa, and Pasupuleti 2016; Nila, Apostol, and Patriciu 2020).

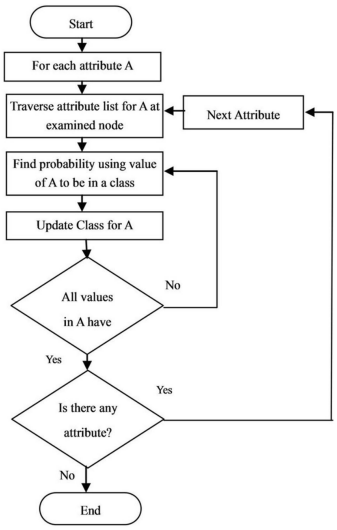

Figure 2.7: Schematic Diagram of the Naive Bayes Classification Process from (Sneha and Gangil 2019)

This decision rule is computationally efficient, making Naive Bayes suitable for real-time prediction tasks (Chandra, Challa, and Pasupuleti 2016; Nila, Apostol, and Patriciu 2020).

However, the strong independence assumption may only hold in some scenarios, potentially leading to suboptimal performance when features are highly correlated. Despite this limitation, Naive Bayes remains popular due to its simplicity and effectiveness in many practical applications (Chandra, Challa, and Pasupuleti 2016).

2.2 Automatic Classification of Security Alerts in SOC

The growing complexity of cyber threats, especially Advanced Persistent Threats (APT), has led to significant advancements in automated detection and mitigation mechanisms. Many of these mechanisms utilize ML techniques. This section examines key works, focusing on their methodologies, limitations, and relevance to the problem of alert triage automation in SOC.

Recent research has explored hybrid approaches to anomaly detection. Saini et al. (2023) proposed a hybrid ensemble model combining Random Forest and XGBoost classifiers, achieving a 99.91% accuracy on the CIC-IDS2017[2] dataset with a False Positive Rate (FPR) as low as 0.12%. Despite its high performance, the model relies on static datasets, which limits its adaptability to evolving attack patterns. Similarly, Ghafir et al. (2018) developed Machine Learning for Advanced Persistent Threats (MLAPT), a multi-phase system incorporating correlation frameworks for APT detection, achieving an accuracy of 84.8%. While this approach improved early-stage prediction, its accuracy was limited compared to ensemble methods and lacked scalability for dynamic environments.

Ali, Shah, and ElAffendi (2024) extended APT datasets to include a diverse range of alerts. They integrated Random Forest and XGBoost models into a SIEM environment, demonstrating the ability to reduce FPR and achieve a near-perfect 99.6% accuracy. However, the computational demands of such systems may hinder practical deployment in resource-constrained environments. On the contrary, Brogi and Tong (2016) utilized Information Flow Tracking (IFT) for APT detection, concentrating on the correlation between the various stages of an attack. While this method excels at tracing complex multi-stage attacks, it requires further automation to support large-scale SOC operations effectively.

Incorporating reinforcement learning, Sethi et al. (2020) introduced a Deep Q-Network (DQN)-based context-adaptive intrusion detection system, which improved FPR and demonstrated robustness against several attacks. Nevertheless, the distributed nature of their approach introduces practical deployment challenges, especially in SOC environments where centralized oversight is critical. Similarly, Nila, Apostol, and Patriciu (2020) explored ML-based triage systems to automate alert classification and reduce false positives. Their work effectively alleviated analyst fatigue by prioritizing actionable alerts, yet the scope of implementation remains limited to standalone systems.

Chandra, Challa, and Pasupuleti (2016) conducted a comprehensive study on email-based APT entry points, explicitly exploring the effectiveness of Bayesian spam filters[3]. These filters demonstrated a strong capability to identify and differentiate between spear-phishing attempts and general spam emails, which is crucial for enhancing cybersecurity measures.

[2] Labeled dataset of network traffic, including normal behavior and various attacks, used to evaluate intrusion detection models.

[3] Bayesian spam filters classify emails as spam or legitimate using probabilities based on Bayes theorem.

However, the authors noted that while their solution provided significant benefits in detecting these specific types of email threats, it needed to address the broader range of APT life cycles. This limitation restricts its effectiveness in SOC environments with diverse attack vectors, requiring a more holistic approach to threat detection and response strategies to encompass the full spectrum of APT strategies and tactics.

Table 2.1 presents a detailed summary of the relevant studies explored, highlighting their respective strengths, weaknesses, and potential areas for enhancement.

Table 2.1: Summary of the Existing Related Works based on Ali, Shah, and ElAffendi (2024).

References	APT Life Cycle Coverage	AI Models Used	Lowest FPR	Highest Accuracy	Limitations
Ghafir et al. (2018)	Complete	Decision Tree, SVM, KNN, Ensemble Classifier	4.5%	84.8%	The blacklist-based detection modules require continuous update. Moreover, the accuracy is lower.
Brogi and Tong (2016)	Complete	No ML model used	High	Not calculated	Information Flow Tracking is used to detect APTs; however, the FPR is high.
Giura and Wang (2012)	Complete	No ML model used	27.88%	Not calculated	Sufficient knowledge is required to set up the mechanism.
Sopan et al. (2019)	No	Random Forest	Not calculated	98.5%	The post-alert decision is not automated; it involves the intervention of security experts.
Farooq and Otaibi (2018)	Partial	SVM, Random Forest	Not calculated	Not calculated	The process anomaly detection is presented formally using One-Class SVM. However, the paper lacks ML-based experimental results.
Sethi et al. (2020)	No	Deep Q-network	0.35%	96.12%	The proposed model enables the detection of APTs.
Nila, Apostol, and Patriciu (2020)	No	ZeroR, OneR, NaiveBayes, SVM, J48, RandomForest	Not calculated	95.5%	The authors did not discuss the FPR of the proposed model.
Chandra, Challa, and Pasupuleti (2016)	Partial	NaiveBayes	Not calculated	87%	The authors did not discuss the FPR of the proposed model.
Saini et al. (2023)	Partial	Random Forest, XGBoost	0.12%	99.91%	This study used existing datasets, posing a challenge because these datasets might not contain the latest attack scenarios.
Ali, Shah, and ElAffendi (2024)	Complete	Random Forest, XGBoost	Not calculated%	99.6%	High resource demand for training; dataset generated in controlled environments, limiting real-world adaptability.

This study introduces a new approach to automating alert triage in SOCs by combining ML models with existing SIEM systems. In contrast with the previous studies, this research will use live data as its training data and will be deployed in a real-world environment. Therefore, analysts can focus on high-risk issues by analyzing the alerts based on the severity of the cyber threat shown by the ML model.

While the related works reviewed in this study showcase significant advancements in SOC automation and alert triage, none closely reflects this research's objectives. However, beyond academic research, commercially available solutions like Check Point's SOC automation tools demonstrate functionalities that align with aspects of this study. These solutions offer valuable insights into the application of AI and automation in SOC environments, providing a practical point of comparison for the approach proposed in this research.

Check Point's approach to SOC automation stands out for its focus on streamlining SOC operations using artificial intelligence (AI) and automation tools (CheckPoint n.d.).

SOC automation, as described by Check Point, employs AI to take over repetitive and manual tasks in the SOC, such as alert triage, incident response, and threat detection. Their solution includes tools like:

- **Generative AI:** Used to improve usability and streamline workflows, enabling analysts to query and interact with data using natural language.
- **Playbooks:** Predefined and customizable workflows for incident response that enable rapid and consistent remediation.
- **Threat Detection and Analytics:** Leveraging advanced analytics, behavioral insights, and proprietary threat intelligence from Check Point Research and ThreatCloud AI to identify threats and reduce false positives.
- **Malware Analysis and Phishing Detection:** Using sandboxes and natural language processing (NLP) to analyze suspicious files and emails.
- **Infinity Extended Prevention and Response (XDR/XPR):** A platform that integrates automated playbooks with real-time threat intelligence and analytics to correlate events across the security estate, detect sophisticated attacks, and prevent lateral movement.

The research in this study aligns closely with Check Point's SOC automation approach. Both aim to automate alert triage and streamline SOC operations using AI to enhance efficiency. This study also shares key goals, such as reducing false positives, prioritizing alerts, and providing analysts with tools to focus on high-risk threats. Like Check Point, this research emphasizes the use of playbooks for consistent responses and aims to alleviate the burden of repetitive tasks on analysts.

While the Check Point solution is comprehensive and includes a full suite of tools, this research focuses on a customized solution tailored to the specific needs of the organization conducting this study differing in:

1. **Integration with Existing Infrastructure:** This study develops a solution that seamlessly integrates with the organization's current SIEM system, avoiding the need for a complete overhaul of existing systems, as would be required to adopt Check Point's platform.

2. **Live Data and Real-World Deployment:** Unlike Check Point's reliance on proprietary threat intelligence and prebuilt systems, this research involves using live data collected within the organization's environment, ensuring the model is trained and deployed in a real-world context.

2.3 Technologies

The following section presents some of the important technologies available in the three categories discussed: SIEM Tools, Ticketing Tools, and Machine Learning Frameworks. The company utilizes QRadar as its SIEM solution and IBM QRadar SOAR and JIRA for ticketing and incident management. The subsequent subsections provide an overview of the tools in each category, followed by a comparative analysis highlighting their strengths and trade-offs.

The following section presents some of the important technologies available in the three categories discussed: SIEM Tools, Ticketing Tools, and Machine Learning Frameworks. Each category serves a unique purpose in the automatic classification of security alerts in a SOC. The tools used for this project were the same as those the company utilizes, including QRadar as its SIEM solution, IBM QRadar SOAR, and JIRA for ticketing and incident management. The subsequent subsections provide an overview of existing tools in each category, followed by a comparative analysis highlighting their strengths and trade-offs.

2.3.1 SIEM Tools

In this subsection, the three leading SIEM tools, QRadar, Splunk, and LogRhythm (exabeam 2024), were chosen for their advanced threat detection, log aggregation, and real-time security analytics. Based on industry adoption, integration capabilities, and performance in SOC environments, each platform offers unique features for efficient incident detection and response.

QRadar

QRadar is a SIEM solution initially developed by IBM and later acquired by Palo Alto Networks (Alto 2024). It excels in advanced threat detection, log aggregation, and automated incident response by collecting and correlating data from various sources like firewalls and network devices. Its features, including anomaly detection and event prioritization, help analysts focus on critical threats, reducing false positives.

A key advantage of QRadar is its integration with other IBM and Palo Alto tools, facilitating seamless workflows. Its machine learning support enables organizations to detect emerging threats, while its modular architecture ensures scalability for businesses of all sizes. Recent updates enhance QRadar's capabilities for modern hybrid and cloud-native infrastructures.

Splunk

Splunk is a powerful and versatile SIEM platform known for its real-time monitoring and analytics. It enables organizations to detect and respond to cybersecurity threats quickly. It excels at processing large volumes of machine-generated data, making it essential for security teams in complex IT environments.

With a broad ecosystem of applications, Splunk supports various use cases such as threat hunting, compliance management, and anomaly detection. Splunk Enterprise Security (ES)'s add-on offers tailored solutions for SOC environments, including prebuilt dashboards and customizable alerts.

Splunk's user-friendly interface and robust reporting tools cater to analysts of all skill levels, while its integration with machine learning enhances proactive threat management. Splunk's flexibility and capabilities make it a preferred choice for organizations seeking a reliable SIEM solution.

LogRhythm

LogRhythm is a robust SIEM platform aimed at enhancing threat detection and incident response. Its user-friendly design and integration capabilities help streamline SOC workflows. The platform utilizes advanced analytics, behavioral anomaly detection, and machine learning for effective threat identification.

LogRhythm also excels in compliance management with preconfigured templates for regulations like GDPR, HIPAA, and PCI-DSS. Its Threat Lifecycle Management (TLM) framework facilitates efficient threat detection and remediation with structured workflows.

A key highlight is its seamless integration with various third-party tools, along with a centralized dashboard that provides actionable insights for SOC teams. Additionally, LogRhythm offers flexible deployment options, making it suitable for diverse infrastructure needs.

2.3.2 Ticketing Tools

This subsection focuses on three ticketing tools: IBM QRadar SOAR, ServiceNow, and JIRA. These tools were selected for their capabilities in automating workflows, improving incident management, and integrating with SIEM systems. Practical ticketing tools are vital for efficiently managing security incidents, reducing response times, and promoting collaboration within SOC teams.

IBM QRadar SOAR

IBM QRadar SOAR enhances the QRadar SIEM platform by streamlining incident management and security operations. It automates workflows, minimizing manual tasks, and features playbook automation that allows analysts to execute consistent responses tailored to organizational policies. The comprehensive case management system centralizes tracking of incidents, evidence, and communications, fostering real-time collaboration. Additionally, it integrates with various third-party tools like EDR systems and cloud services, enabling SOC teams to focus on high-priority threats and improving overall efficiency in incident resolution (IBM n.d.).

ServiceNow

ServiceNow is a versatile platform originally designed for IT service management (ITSM) that has expanded to include strong incident management for cybersecurity. It is widely adopted across industries and integrates effectively with SIEM tools and other security systems, making it valuable for SOC teams.

The Security Incident Response (SIR) module offers a centralized approach for managing security incidents, featuring workflow automation for incident triage, escalation, and remediation. Its dashboard capabilities enable real-time monitoring and reporting, providing security managers with visibility into ongoing incidents.

A key strength of ServiceNow is its seamless integration with vulnerability scanners, endpoint protection platforms, and threat intelligence feeds, enhancing its ability to prioritize alerts and correlate events. Its scalable architecture makes it suitable for organizations of all sizes, from small businesses to global enterprises.

JIRA

JIRA is widely recognized as a project management tool and has been adapted to manage security incidents in SOC environments. Its user-friendly interface allows organizations to enhance incident management with minimal training requirements. JIRA's ticketing system helps analysts create, assign, and update tickets for security alerts, ensuring systematic documentation and resolution.

JIRA integrates effectively with SIEM tools and offers extensive customization options for workflows and notifications, further enhancing its functionality. Although it lacks advanced features like playbook-driven automation found in IBM QRadar SOAR, JIRA remains a practical and cost-effective solution for incident management, especially for organizations already familiar with its ecosystem.

2.3.3 Machine Learning Frameworks

This section discusses three popular machine learning frameworks: Scikit-learn, TensorFlow, and PyTorch (GeeksforGeeks 2024). These frameworks effectively implement scalable machine learning models in research and production.

Scikit-learn

Scikit-learn is a Python library known for its simplicity in implementing classical machine learning algorithms like Random Forest, Naive Bayes, and Support Vector Machines (SVM). It excels in classification, regression, and clustering tasks and offers comprehensive preprocessing tools for effective data cleaning and transformation. The library features strong cross-validation capabilities for evaluating model performance and integrates well with other Python libraries such as NumPy, pandas, and matplotlib. Scikit-learn is ideal for quick prototyping and development, making it a popular choice for both beginners and professionals, though it may not be as effective for deep learning or very large datasets.

TensorFlow

TensorFlow is an open-source framework developed by Google, designed for scalable machine learning and deep learning applications. It efficiently handles large-scale computations and supports various techniques, including traditional algorithms and advanced architectures like CNNs and RNNs. The Keras API within TensorFlow simplifies model building for users with limited experience.

One of its key strengths is deploying models across platforms like mobile, edge computing, and cloud environments. TensorBoard, a visualization tool, helps users monitor training

and performance, enabling effective algorithm fine-tuning. Despite a steeper learning curve compared to some alternatives, TensorFlow's extensive capabilities make it ideal for complex applications, including cybersecurity tasks like anomaly detection and threat prediction.

PyTorch

PyTorch is a flexible machine learning library that has gained popularity among researchers and developers due to its dynamic computation graph, allowing real-time changes during model execution. It excels in deep learning tasks like natural language processing, computer vision, and reinforcement learning, with the torch.nn module simplifying neural network creation and the autograd module providing efficient gradient computation. PyTorch's intuitive design makes it easy for new users and integrates well with libraries like NumPy and ONNX.

2.3.4 Comparative Analysis

A comparative overview of all the discussed technologies will be presented here. This analysis highlights the strengths and limitations of the SIEM tools, ticketing tools, and machine learning frameworks identified in previous subsections.

SIEM Tools

Table 2.2: Comparative Analysis of SIEM Tools.

Feature	**QRadar**	**Splunk**	**LogRhythm**
Integration	Deep IBM ecosystem integration	Extensive third-party support	Moderate third-party support
Threat Detection	Advanced anomaly detection	Real-time search and analytics	Strong ML-based detection
Compliance	Strong compliance capabilities	Flexible for custom compliance	Focused on compliance workflows
Machine Learning	Limited native ML	Third-party ML integrations	Native ML capabilities
Ease of Use	Moderate	User-friendly UI	Moderate

As shown in Table 2.2, QRadar stands out due to its deep integration with IBM solutions, making it an excellent choice for organizations already utilizing IBM's ecosystem. In contrast, Splunk offers powerful real-time analytics and extensive integration with third-party tools, which allows it to be adaptable for various use cases. LogRhythm focuses on compliance and features native machine learning capabilities, although it may not provide the same level of seamless integration within IBM environments as QRadar does.

Ticketing Tools

Table 2.3: Comparative Analysis of Ticketing Tools.

Feature	QRadar SOAR	ServiceNow	JIRA
Integration	Seamless with QRadar SIEM	Broad integrations across IT	Extensive integrations with development tools
Automation	Playbook-driven workflows	Customizable workflow automation	Moderate workflow automation
Incident Management	Case management and playbooks	Advanced IT service management	Simplified incident tracking
Ease of Use	Moderate	Customizable but complex	Highly user-friendly

Table 2.3 outlines the strengths and weaknesses of various ticketing tools. IBM QRadar SOAR closely integrates with the QRadar SIEM platform, providing strong automation through playbooks and advanced incident management features. ServiceNow is notable for its flexibility and capability to cater to multiple IT departments. In contrast, JIRA offers a more user-friendly and cost-effective solution, with moderate automation functionalities and extensive integration capabilities.

Machine Learning Frameworks

Table 2.4: Comparative Analysis of Machine Learning Frameworks.

Feature	Scikit-learn	TensorFlow	PyTorch
Algorithms Supported	Classical ML (Random Forest)	Deep Learning and Classical ML	Deep Learning and Research ML
Scalability	Moderate	High	High
Ease of Use	Beginner-friendly	Moderate complexity	High flexibility for research
Deployment	Less optimized for deployment	Production-ready	Suitable for research & testing
Flexibility	Limited to classical models	High for both models and tuning	High for experimentation

As outlined in Table 2.4, Scikit-learn is a user-friendly and effective library for implementing classical machine learning algorithms such as Random Forest and Naive Bayes. TensorFlow excels in scalability and is well-suited for production environments, making it ideal for deep learning applications. On the other hand, PyTorch is known for its flexibility and dynamic computation graphs, making it the preferred choice for research-focused machine learning and experimentation.

2.3.5 Conclusion

Based on the comparative analysis, QRadar, IBM QRadar SOAR, and Scikit-learn emerge as the most suitable technologies for this project due to their strong integration, automation capabilities, and alignment with the project goals. Importantly, these tools also align with the technologies available within the company where this project will be implemented. Therefore, had it not been the case and regardless of any prior conclusions, QRadar, IBM QRadar SOAR, and Scikit-learn will be used, ensuring both feasibility and alignment with organizational resources.

Chapter 3

Method and Implementation

This chapter details the methodology employed to tackle the problem, providing an overview of the technologies utilized and the implementation process.

It consists of two sections: the first describes the method's design and proposed solutions, while the second covers the proof of concept, including implementation and optimization in a live environment.

3.1 Method

This section describes the methodological foundation of the project. It begins by presenting the technologies used and continues with an outline of three distinct solution strategies to the problem.

Each approach is described through its pipeline and architectural design, offering a comparative view of possible solutions.

3.1.1 Technological Overview

The solution will be developed and implemented using the Python programming language, version 3.10.12. The following libraries are expected to be used throughout the development of the project:

- **scikit-learn** (version 1.6.1): This library will be used for building machine learning models, including the Random Forest model. It provides necessary functionalities for model training, testing, and performance evaluation.
- **pandas** (version 2.2.3): Pandas will be used for data manipulation and analysis, particularly for handling, processing, and cleaning the alert datasets. It provides efficient data structures for handling large amounts of structured data.
- **fastapi** (version 0.115.11): FastAPI will be used to create the necessary APIs for data flow between the components. It will expose endpoints for the system, allowing real-time predictions and feedback from analysts to be communicated between the machine learning models and the IBM SOAR interface.
- **matplotlib** (version 3.10.1): Matplotlib will be used to generate plots and graphs, particularly for data exploration, visualizing model performance, and evaluating results. It will assist in identifying patterns in the data and understanding how well the model is classifying the security alerts.

- **SentenceTransformer** (likely needed): SentenceTransformer will be used for text vectorization, particularly for converting textual data such as alert descriptions and analyst comments into numerical embeddings. These embeddings will be used as inputs to machine learning models for better understanding and classification of textual data.

3.1.2 Problem-Solving Approaches

This subsection analyzes three distinct approaches to tackling the identified problem. It outlines the specific methodologies employed for each approach, detailing the pipeline process involved and the architectural framework that supports its implementation.

Pipeline Figure 3.1 represents the general data processing pipeline for all solutions, illustrating the flow of data through various stages before it reaches the analyst.

The pipeline is designed to handle security alerts from multiple sources, process the data, and present the predictions to the analyst.

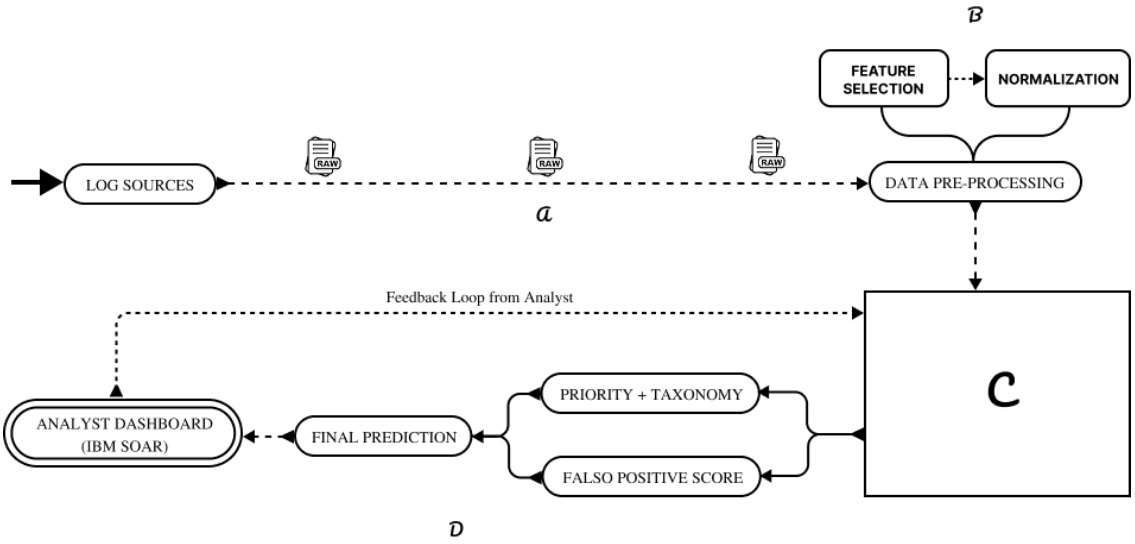

Figure 3.1: General Pipeline for all Three Solutions

This pipeline remains largely the same across all solutions, with the exception of section `C`, where the machine learning model varies depending on the solution. While the overall structure of the pipeline is consistent, the type of model used in section `C` determines the specific prediction process and outputs.

The described pipeline includes several key stages:

- **Raw security alert data ingestion:** In section `A`, alert data is collected from various log sources, such as SIEM systems, IDS, and firewalls. This data typically contains a wide range of information, including alert descriptions, timestamps, and metadata. In this section, the data is prepared for processing by standardizing the format across different sources.
- **Data pre-processing:** In section `B`, the collected data undergoes a series of cleaning and normalization procedures. This stage includes feature extraction, where relevant information from the raw data is transformed into structured, usable features.

- **Machine Learning Model (specific to each solution):** Section C involves feeding the pre-processed data into the machine learning model. The core of this section differs depending on the solution.
- **Final predictions for the analyst dashboard:** After the predictions are generated in section C, the results are passed on to the IBM SOAR dashboard, section D. This dashboard presents the predictions to the analyst, who can review the outputs, including taxonomy, priority, false positive status, and confidence score. The analyst can provide feedback on these predictions, which is sent back into the system to refine future predictions, contributing to the model's long-term accuracy and adaptability.

This pipeline was designed to be modular and flexible, allowing for easy integration with existing systems and the ability to adapt to evolving security threats.

Random Forest with Reinforcement Learning Feedback Loop

The first solution proposed in this study is a two-layered machine learning system that integrates a Random Forest model with a Reinforcement Learning feedback loop.

This solution addresses the problem using a pre-trained RF model on historical data combined with an RL model that refines predictions based on analyst feedback. This integration of RF with RL contributes to the model's adaptability, making it highly responsive to new threats.

Design The RF model serves as the decision-making core, trained on a historical dataset of security alerts to classify incoming alerts by taxonomy and priority—Objective 3.

While the RF model provides strong initial predictions, it struggles to adapt to new attack vectors not represented in its training data. To address this, the RL model enhances the predictions made by the RF model and evaluates alerts as false positives or true positives, using feedback from security analysts.

The implementation complexity of this solution is considered moderate, as it does not involve highly intricate algorithms or architectures. However, the necessity to implement and integrate two distinct models—a RF for initial predictions and a RL model for feedback-driven refinement—adds an additional layer of complexity.

Key features of this solution include:

- The RL model adjusts its parameters through a continuous feedback loop, improving classification and generating confidence scores.
- This dynamic learning process ensures the system is able to handle both `cold start` issues effectively (thanks to the pre-trained RF model) and false positives in real-time.
- The system's adaptability to new threats is high via the RL model and helps manage false positives by learning patterns of false positives or true positives, refining its algorithms, and potentially reducing analysts' workloads.
- Designed to efficiently manage and analyze real-time alerts generated from a diverse range of log sources—Objective 4.

The interpretability of this solution is considered moderate due to the significant role played by AI in its decision-making processes. The RF model, being a supervised learning algorithm,

provides a level of transparency as its outputs can be traced back to the quality and features of the historical dataset used for training.

However, integrating RL complicates interpretability. RL's iterative feedback and rewards lead to less transparent reasoning, making analyzing and communicating insights challenging.

Integrating this solution with IBM SOAR is relatively easy and straightforward. By relying on the API endpoints provided by this solution, IBM SOAR can seamlessly communicate with the solution platform. This is vital for enhancing the overall effectiveness of the solution—Objective 5.

IBM SOAR provides customization of dashboards presented to analysts, enabling:

- Creation of custom fields where analysts can view the solution's outputs and mark them as correct or incorrect.
- Feedback from analysts, which is crucial for the RL model to learn from mistakes and improve over time.

This feedback loop supports continuous refinement and enhances accuracy for future predictions. Overall, this strategy leverages the strengths of supervised and reinforcement learning to create a robust solution for security alert triage. It enhances operational efficiency and threat response capabilities by:

- Minimizing false positives through continuous improvement.
- Allowing integration with IBM SOAR to evaluate and test the solution's effectiveness in categorizing alerts and minimizing false positives—Objective 6.

Pipeline The illustration in Figure 3.2 corresponds to section `C` of the general pipeline in Figure 3.1.

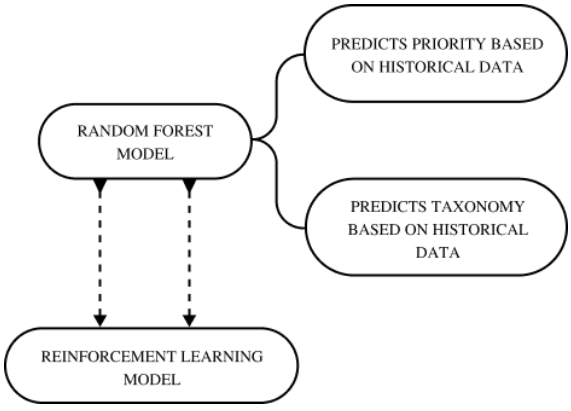

Figure 3.2: Part C of the General Pipeline for the Random Forest with Reinforcement Learning Feedback Loop Solution

This figure provides a detailed representation of the machine learning model's specific implementation for this solution.

It elaborates on the processes and components that occur within section `C`, as outlined in the general pipeline, showcasing the unique aspects of this solution's approach to data processing and prediction generation.

End-to-End Deep Learning Classifier with Feature Fusion

The second solution in this study employs a deep neural network (DNN) model trained end-to-end on labeled security alert data, based on historical alerts dataset—Objective 3.

Unlike solution one, which uses a pre-trained RF model, this approach processes raw SIEM logs, analyst comments, and metadata through the DNN's attention layers to automatically extract features and classify data, eliminating the need for manual feature engineering.

While solution one adapts to new threats with a reinforcement learning feedback loop, solution two relies solely on the DNN model, necessitating training on a comprehensive dataset before deployment. The DNN excels in complex feature extraction environments but requires periodic retraining based on analyst feedback.

Design In this solution, the DNN model processes raw security alert data directly, leveraging attention mechanisms to identify and prioritize critical features like alert descriptions, origins, and metadata.

The attention layers enhance the model's ability to focus on relevant patterns, improving classification accuracy and adaptability.

By automating feature extraction, this approach streamlines the pipeline, reducing dependency on domain-specific preprocessing techniques.

Key aspects of this solution are:

- The DNN model takes in raw security alert data, which is vectorized, and outputs predictions on alert taxonomy, priority, false positive status, and confidence score.
- The DNN model's adaptability is moderate and depends on the training data's quality and diversity. Once trained, it performs well on known data but is less responsive to new attack patterns unless retrained with fresh data.
- Solution two faces challenges with cold start problems, as the DNN model requires a substantial amount of labeled training data before it can begin making accurate predictions. Unlike solution one, which benefits from the pre-trained RF model, this solution requires extensive training before it can function effectively.
- Engineered to effectively process and evaluate real-time alerts originating from a wide variety of log sources—Objective 4.

The implementation complexity of solution two is considered high due to the extensive computational resources required to train the DNN model. Training a deep neural network, especially one that handles raw alert data with attention mechanisms, demands significant time and resources, making it more complex to implement.

The interpretability of the solution is low, primarily because the DNN model operates as a "black box". While it offers powerful predictions, the model's inner workings are difficult to explain, particularly when attention layers are involved, making it harder to interpret why certain predictions were made.

This solution integrates with IBM SOAR for real-time feedback and performance evaluation—Objective 5, allowing analysts to review predictions and provide feedback, which is subsequently used for periodic retraining to assess and validate the solution's capability in classifying alerts and reducing the occurrence of false positives—Objective 6.

Rule-Augmented Decision Tree with Feedback Aggregation

The third solution proposed in this study integrates static expert rules with a lightweight Decision Tree Classifier (DTC). This hybrid system uses a rule engine to pre-filter known benign or critical alert patterns before passing any unknown or ambiguous cases to the Decision Tree Model (DTM).

The rules provide initial filtering for quick decisions on clear alerts. For complex cases, the DTC predicts based on available data. Feedback is reviewed in batches, enabling periodic updates to the rule base and retraining of the decision tree.

Design This solution combines the benefits of domain-specific rules with machine learning. It starts with a rule engine that quickly processes alerts based on known patterns, tagging them as benign or critical and minimizing the need for further analysis. This approach is efficient in environments with well-defined, static patterns. This solution will be designed to ingest data from a variety of log sources, including SIEM systems and other security tools—Objective 4.

The rule engine will preprocess and filter alerts from these sources, leveraging predefined rules to handle known patterns efficiently, while less adaptive, predefined rules to filter known benign patterns, ensuring a baseline reduction in false positives.

For alerts that don't fit predefined rules, the DTC classifies them using features from raw data, allowing the system to manage both known and unknown patterns while balancing interpretability and accuracy.

The DTC works with a clean, segmented dataset to differentiate between alerts that match known patterns and those requiring further analysis—Objective 3.

Key features of this solution include:

- The rule engine filters known benign or critical patterns before passing uncertain alerts to the DTC. This approach reduces the overall processing time and helps prioritize more ambiguous cases.
- Feedback is logged and applied asynchronously, meaning that the system doesn't update in real-time. Instead, the rule base is updated, and the decision tree is retrained on a periodic basis (e.g., weekly). This ensures that the system evolves over time but does not immediately adjust to new threats in real-time.
- The system's ability to handle new and evolving threats is low, as it depends on manually updating the rule base to address new threats.
- *Cold start* handling is excellent because the rules-based filtering system can operate immediately without requiring training, and the DTC is lightweight enough to be quickly deployed after the initial setup.

The implementation complexity of this solution is low, as it uses well-understood decision tree models and a simple rule engine. This makes it relatively easy to implement when compared to the other solutions. However, the trade-off is that the DTC may not be as powerful as deep learning-based models or even RL models when dealing with complex, high-dimensional data.

The interpretability of this solution is considered high. Both the rule engine and the decision tree are inherently interpretable, allowing analysts to understand the reasoning behind

the classifications. This makes it particularly useful in environments where auditability and transparency are important. However, the model's inability to adapt in real-time can make it less suitable for rapidly evolving attack patterns.

Integrating this solution with IBM SOAR is straightforward, allowing for seamless communication between the alert system and the dashboard used by analysts. Alerts classified by the rule engine and decision tree are sent to the IBM SOAR dashboard, where analysts can review predictions and provide feedback—Objective 5. The feedback is stored and reviewed periodically to refine the rules and retrain the decision tree.

Architecture

The diagram in Figure 3.3 illustrates the architecture for all solutions.

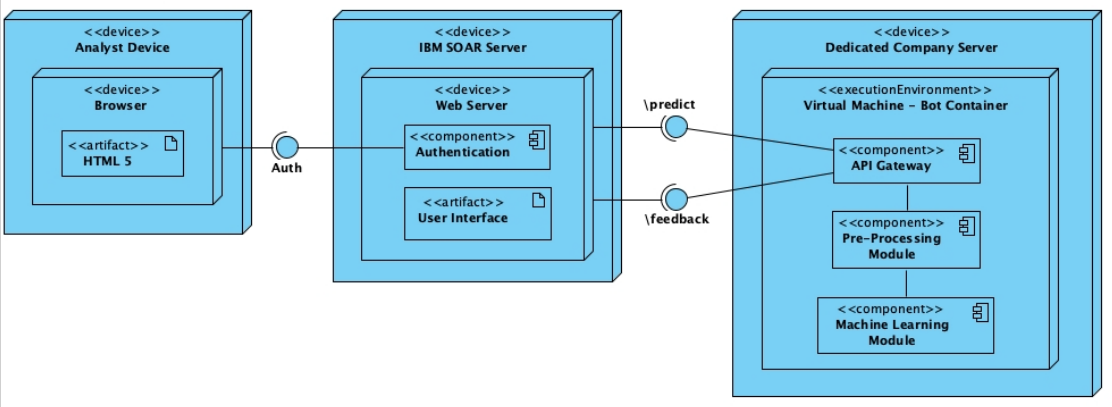

Figure 3.3: Architecture of the Random Forest with Reinforcement Learning Feedback Loop Solution

It's organized into three major sections:

1. **Analyst Device**: This device acts as the point of interaction where security analysts review and manage alerts. Represents the user interface. The analyst logs into the system through the **Authentication** component, enabling access to the dashboard.
2. **IBM SOAR Server**: The IBM SOAR Server is the central system for managing security alerts. The server sends requests to the **Bot Container** through the `\predict` endpoint, triggering the alert classification process. Once the alert is processed, the `\feedback` endpoint is used to receive the analyst's input for further training of the RL model.
3. **Dedicated Company Server**: The **Dedicated Company Server** hosts the **Bot Container**, which is deployed on a **Virtual Machine (VM)**. This container is responsible for the core functionality of the solution, consisting of three main components:
    - **API Gateway**: This component serves as the entry point for receiving requests from the IBM SOAR Server and handling communication between the components inside the bot container. It processes the incoming alert data and forwards it to the appropriate modules.

- **Pre-Processing Module**: This module cleans and prepares the incoming alert data, extracting features and transforming them into a suitable format for the model.
- **Machine Learning Module**: The **Machine Learning Module** applies the Random Forest (RF) model to classify the alert. These predictions are then further refined through the Reinforcement Learning (RL) model.

This architecture is designed to be modular and scalable, allowing for easy integration with existing systems and the ability to adapt to evolving security threats.

3.1.3 Comparative Analysis

Table 3.1 presents a comparison of the three proposed approaches based on key evaluation criteria. The goal is to assess their suitability in the context of a real-world SOC environment, taking into consideration implementation complexity, adaptability, performance, interpretability, and integration potential.

Table 3.1: Comparison of Proposed Solutions

Criteria	RF + RL	Deep Neural Network (DNN)	Rules + Decision Tree
Architecture Type	RF + RL (Two-layer)	DNN	Rule-based + Decision Tree
Implementation Complexity	Moderate	High	Low
Adaptability to New Threats	High (via RL)	Moderate (via retraining)	Low (manual updates)
Learning from Feedback	Online (via RL)	Periodic retraining	Batch/manual integration
Cold Start Handling	Excellent (RF pre-trained)	Poor	Excellent (rules pre-set)
Interpretability	Moderate	Low	High
Scalability	High	High	Moderate
Integration with SIEM (IBM SOAR)	Easy	Easy	Easy
False Positive Reduction	Adaptive (confidence scoring)	Model confidence only	Rigid (rule-defined)
Performance in Evolving Scenarios	High	Moderate	Low

Solution 1 is optimal because it offers adaptability, scalability, and real-time learning. Integrating RF with RL continuously adapts to new threats through real-time feedback, making it highly effective in dynamic cybersecurity environments.

Solution 2 excels in feature extraction using DNN but lacks immediate adaptability. It requires periodic retraining, making it less responsive to evolving threats than Solution 1.

Solution 3, while simple and interpretable, is not adaptable. It relies on static rules and manual updates, making it less suitable for fast-changing threat landscapes.

In conclusion, Solution 1 is the best option for real-time alert classification and long-term adaptability in a SOC environment.

3.2 Proof of Concept

This section presents the practical implementation of the proposed solution, including the setup of the test environment, preparation of the dataset, and execution of the machine learning models. It aims to demonstrate the feasibility and performance of the selected approach under realistic conditions.

3.2.1 Dataset Division

This subsection details how the dataset was split for training, validation, and testing.

3.2.2 Data Processing

Explanation of how the raw data was prepared for use in the machine learning pipeline.

3.2.3 Data Normalization

Discussion of normalization techniques applied to the data to improve model performance.

3.2.4 Machine Learning Model Development

In this section, the implementation of selected machine learning algorithms is discussed.

3.2.5 Hyperparameter Tuning

Details of the strategy used to tune hyperparameters and optimize the models' performance.

Chapter 4

Model Evaluation and Results Analysis

Chapter 5

Conclusion and Future Work

5.1 Future Work

Bibliography

Ali, Gauhar, Sajid Shah, and Mohammed ElAffendi (Sept. 2024). *Enhancing Cybersecurity Incident Response: AI-Driven Optimization for Strengthened Advance Persistence Threat Detection*. doi: 10.20944/preprints202409.1725.v1. url: https://www.preprints.org/manuscript/202409.1725/v1.

Alto, Palo (2024). *Palo Alto Networks® Closes Acquisition of IBM's QRadar SaaS Assets - Palo Alto Networks*. url: https://www.paloaltonetworks.com/company/press/2024/palo-alto-networks--closes-acquisition-of-ibm-s-qradar-saas-assets.

Arianna, By (2024). *Cybersecurity Management Services: Why ITSM Services Essential?* url: https://forlicoupon.it/2024/07/17/cybersecurity-management-services-why-itsm-services-essential/.

ArtResilia (n.d.). *ArtResilia*. url: https://www.artresilia.com/.

Breiman, Leo (1996). "Bagging predictors". In: *Machine Learning* 24 (2), pp. 123–140. issn: 08856125. doi: 10.1007/BF00058655/METRICS. url: https://link.springer.com/article/10.1007/BF00058655.

Brogi, Guillaume and Valérie Viet Triem Tong (Dec. 2016). "TerminAPTor: Highlighting advanced persistent threats through information flow tracking". In: *2016 8th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2016*. doi: 10.1109/NTMS.2016.7792480.

Brownlee, N. and E. Guttman (June 1998). "Request for Comments 2350 Expectations for Computer Security Incident Response," in: url: http://www.ietf.org/rfc/rfc2350.txt.

Chandra, J. Vijaya, Narasimham Challa, and Sai Kiran Pasupuleti (Aug. 2016). "A practical approach to E-mail spam filters to protect data from advanced persistent threat". In: doi: 10.1109/ICCPCT.2016.7530239.

Chauhan, Nagesh (Feb. 2022). *Decision Tree Algorithm, Explained - KDnuggets*. url: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

CheckPoint (n.d.). *What is SOC Automation? - Check Point Software*. url: https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-soc/what-is-soc-automation/.

CriticalStart (2019). *THE IMPACT OF SECURITY ALERT OVERLOAD*. url: https://www.criticalstart.com/wp-content/uploads/2021/02/CS_Report-The-Impact-of-Security-Alert-Overload.pdf.

Crowley, Christopher and John Pescatore (2018). *The Definition of SOC-cess? A SANS Survey*.

Dasarathy, Belur V. and Belur V. Sheela (1979). "A composite classifier system design: Concepts and methodology". In: *Proceedings of the IEEE* 67 (5), pp. 708–713. issn: 0018-9219. doi: 10.1109/PROC.1979.11321. url: https://www.academia.edu/30910041/A_composite_classifier_system_design_concepts_and_methodology.

ElSahly, Osama and Akmal Abdelfatah (Aug. 2023). "An Incident Detection Model Using Random Forest Classifier". In: *Smart Cities* 6 (4), pp. 1786–1813. issn: 26246511. doi: 10.3390/SMARTCITIES6040083.

exabeam (2024). *Best SIEM Solutions: Top 10 SIEM systems and How to Choose | Exabeam*. url: https://www.exabeam.com/explainers/siem-tools/siem-solutions/.

Farooq, Hafiz M. and Naif M. Otaibi (Dec. 2018). "Optimal machine learning algorithms for cyber threat detection". In: *Proceedings - 2018 UKSim-AMSS 20th International Conference on Modelling and Simulation, UKSim 2018*, pp. 32–37. doi: 10.1109/UKSIM.2018.00018.

GeeksforGeeks (2024). *Top 10 Machine Learning Frameworks in 2025 - GeeksforGeeks*. url: https://www.geeksforgeeks.org/machine-learning-frameworks/.

Ghafir, Ibrahim et al. (July 2018). "Detection of advanced persistent threat using machine-learning correlation analysis". In: *Future Generation Computer Systems* 89, pp. 349–359. issn: 0167739X. doi: 10.1016/J.FUTURE.2018.06.055. url: https://bradscholars.brad.ac.uk/handle/10454/17614.

Giura, Paul and Wei Wang (2012). "A context-based detection framework for advanced persistent threats". In: *Proceedings of the 2012 ASE International Conference on Cyber Security, CyberSecurity 2012*, pp. 69–74. doi: 10.1109/CYBERSECURITY.2012.16.

Hámornik, Balázs Péter and Csaba Krasznay (2018). "A team-level perspective of human factors in cyber security: Security operations centers". In: *Advances in Intelligent Systems and Computing* 593, pp. 224–236. issn: 21945357. doi: 10.1007/978-3-319-60585-2_21. url: https://www.researchgate.net/publication/318177610_A_Team-Level_Perspective_of_Human_Factors_in_Cyber_Security_Security_Operations_Centers.

Hansen, Lars Kai and Peter Salamon (1990). "Neural Network Ensembles". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (10), pp. 993–1001. issn: 01628828. doi: 10.1109/34.58871. url: https://www.researchgate.net/publication/3191841_Neural_Network_Ensembles.

Harper, Allen et al. (Dec. 2010). *Security Information And Event Management (SIEM) Implementation*. Ed. by MCGRAW-HILL EDUCATION - EUROPE, p. 464. isbn: 9780071701099.

Huang, Xiang (July 2024). "Predictive Models: Regression, Decision Trees, and Clustering". In: *Applied and Computational Engineering* 79 (1), pp. 124–133. issn: 2755-273X. doi: 10.54254/2755-2721/79/20241551. url: https://www.ewadirect.com/proceedings/ace/article/view/13989.

Iamnitchi, Adriana et al. (2017). "An Anthropological Study of Security Operations Centers to Improve Operational Efficiency". In: url: https://digitalcommons.usf.edu/etd.

IBM (n.d.). *Integrations - IBM QRadar SOAR*. url: https://www.ibm.com/products/qradar-soar/integrations?utm_source=chatgpt.com.

Jalalvand, Fatemeh et al. (Nov. 2024). "Alert Prioritisation in Security Operations Centres: A Systematic Survey on Criteria and Methods". In: *ACM Computing Surveys* 57 (2), pp. 1–36. issn: 0360-0300. doi: 10.1145/3695462. url: https://dl.acm.org/doi/10.1145/3695462.

Joseph (Aug. 2022). *TensorFlow Ensemble Learning: What You Need to Know - reason.town*. url: https://reason.town/tensorflow-ensemble-learning/.

Kokulu, Faris Bugra et al. (Nov. 2019). "Matched and mismatched SOCs: A qualitative study on security operations center issues". In: *Proceedings of the ACM Conference on Computer and Communications Security*. Association for Computing Machinery, pp. 1955–1970. isbn: 9781450367479. doi: 10.1145/3319535.3354239.

Li, Nan, Yang Yu, and Zhi Hua Zhou (2012). "Diversity regularized ensemble pruning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7523 LNAI (PART 1), pp. 330–345. issn:

03029743. doi: 10.1007/978-3-642-33460-3_27. url: https://www.researchgate.net/publication/267404624_Diversity_Regularized_Ensemble_Pruning.

Mienye, Ibomoiye Domor and Yanxia Sun (2022). "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects". In: *IEEE Access* 10, pp. 99129–99149. issn: 21693536. doi: 10.1109/ACCESS.2022.3207287.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2012). *Foundations of Machine Learning*. isbn: 978-0-262-01825-8. url: https://www.hlevkin.com/hlevkin/45MachineDeepLearning/ML/Foundations_of_Machine_Learning.pdf.

Moradi, Mehrdad, Bert Van Acker, and Joachim Denil (Feb. 2023). "Failure Identification Using Model-Implemented Fault Injection with Domain Knowledge-Guided Reinforcement Learning". In: *Sensors 2023, Vol. 23, Page 2166* 23 (4), p. 2166. issn: 1424-8220. doi: 10.3390/S23042166. url: https://www.mdpi.com/1424-8220/23/4/2166/htm%20https://www.mdpi.com/1424-8220/23/4/2166.

Muniz, Joseph, Gary McIntyre, and Nadhem AlFardan (2015). *Security Operations Center: Building, Operating, and Maintaining your SOC*. url: https://books.google.pt/books?hl=pt-PT&lr=&id=riraCgAAQBAJ&oi=fnd&pg=PT28&ots=SBNh-G3lCL&sig=6w_ipaEbsLDSFja8OQABKEBW7GE&redir_esc=y#v=onepage&q&f=false.

Nila, Constantin, Ioana Apostol, and Victor Patriciu (June 2020). "Machine learning approach to quick incident response". In: *2020 13th International Conference on Communications, COMM 2020 - Proceedings*, pp. 291–296. doi: 10.1109/COMM48946.2020.9141989.

Rutledge, Samantha (2024). *How SOC as a Service can help Sarah in Operations*. url: https://fractionalciso.com/how-soc-as-a-service-can-help-sarah-in-operations/.

Saini, Neeraj et al. (Dec. 2023). "A hybrid ensemble machine learning model for detecting APT attacks based on network behavior anomaly detection". In: *Concurrency and Computation: Practice and Experience* 35 (28), e7865. issn: 1532-0626. doi: 10.1002/CPE.7865. url: https://researcher.manipal.edu/en/publications/a-hybrid-ensemble-machine-learning-model-for-detecting-apt-attack.

Schapire, Robert E (1990). *The Strength of Weak Learnability*.

Sethi, Kamalakanta et al. (Dec. 2020). "A context-aware robust intrusion detection system: a reinforcement learning-based approach". In: *International Journal of Information Security* 19 (6), pp. 657–678. issn: 16155270. doi: 10.1007/S10207-019-00482-7.

Shaw, Megan (Aug. 2022). *Importance of SIEM in Supporting Digital Transformation Initiatives | DNIF*. url: https://www.dnif.it/en/blog/importance-of-siem-in-supporting-digital-transformation-initiatives.

Sheeraz, Muhammad et al. (Apr. 2023). *Effective Security Monitoring Using Efficient SIEM Architecture*. doi: 10.22967/HCIS.2023.13.023.

Shirey, R (2007). *Network Working Group*. url: http://tools.ietf.org/html/rfc4949.

Sneha, N. and Tarun Gangil (Dec. 2019). "Analysis of diabetes mellitus for early prediction using optimal features selection". In: *Journal of Big Data* 6 (1). issn: 21961115. doi: 10.1186/S40537-019-0175-6.

Sopan, Awalin et al. (May 2019). "Building a Machine Learning Model for the SOC, by the Input from the SOC, and Analyzing it for the SOC". In: *2018 IEEE Symposium on Visualization for Cyber Security, VizSec 2018*. doi: 10.1109/VIZSEC.2018.8709231.

Sovilj, Dušan et al. (2020). "A comparative evaluation of unsupervised deep architectures for intrusion detection in sequential data streams". In: *Expert Systems with Applications* 159, p. 113577. issn: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2020.113577. url: https://www.sciencedirect.com/science/article/pii/S0957417420304012.

Sundaramurthy, Sathya Chandran et al. (Nov. 2014). "A tale of three security operation centers". In: *Proceedings of the ACM Conference on Computer and Communications Security* 2014-November (November), pp. 43–50. issn: 15437221. doi: 10.1145/2663887.2663904. url: https://www.researchgate.net/publication/265786529_A_Tale_of_Three_Security_Operation_Centers.

Suthaharan, Shan (2016). *Machine Learning Models and Algorithms for Big Data Classification*. Vol. 36. Springer US. isbn: 978-1-4899-7640-6. doi: 10.1007/978-1-4899-7641-3. url: https://link.springer.com/10.1007/978-1-4899-7641-3.

Tao, Lin (May 2018). "A DATA TRIAGE RETRIEVAL SYSTEM FOR CYBER SECURITY OPERATIONS CENTER". In.

Tines (2023). *Voice of the SOC 2023 Report*. url: https://www.tines.com/reports/voice-of-the-soc-2023/.

Vielberth, Manfred et al. (2020). "Security Operations Center: A Systematic Study and Open Challenges". In: *IEEE Access* 8, pp. 227756–227779. issn: 21693536. doi: 10.1109/ACCESS.2020.3045514.

Yang, Jianxin et al. (Nov. 2019). "Delineation of urban growth boundaries using a patch-based cellular automata model under multiple spatial and socio-economic scenarios". In: *Sustainability (Switzerland)* 11 (21). issn: 20711050. doi: 10.3390/SU11216159.

Zimmerman, Carson (2014). *Ten Strategies of a World-Class Cybersecurity Operations Center*. isbn: 9780692243107. url: www.mitre.org.