Machine Learning and Optimization Framework for Efficient Alert Management in a Cybersecurity Operations Center

JALAL GHADERMAZI and ANKIT SHAH, University of South Florida, Tampa, USA
SUSHIL JAJODIA, George Mason University, Fairfax, USA

Cybersecurity operations centers (CSOCs) protect organizations by monitoring network traffic and detecting suspicious activities in the form of alerts. The security response team within CSOCs is responsible for investigating and mitigating alerts. However, an imbalance between alert volume and available analysts creates a backlog, putting the network at risk of exploitation. Recent research has focused on improving the alert-management process by triaging alerts, optimizing analyst scheduling, and reducing analyst workload through systematic discarding of alerts. However, these works overlook the delays caused in alert investigations by several factors, including: (i) false or benign alerts contributing to the backlog; (ii) analysts experiencing cognitive burden from repeatedly reviewing unrelated alerts; and (iii) analysts being assigned to alerts that do not match well with their expertise. We propose a novel framework that considers these factors and utilizes machine learning and mathematical optimization methods to dynamically improve throughput during work shifts. The framework achieves efficiency by automating the identification and removal of a portion of benign alerts, forming clusters of similar alerts, and assigning analysts to alerts with matching attributes. Experiments conducted using real-world CSOC data demonstrate a 60.16% reduction in the alert backlog for an 8-h work shift compared to currently employed approach.

CCS Concepts: • **Security and privacy** → **Network security**; **Intrusion/anomaly detection and malware mitigation**; • **Mathematics of computing** → **Mathematical optimization**; • **Computing methodologies** → **Machine learning algorithms;**

Additional Key Words and Phrases: Cyber alert management, ML and optimization framework, unsupervised learning, mathematical programming, alert clusters, analysts to alerts assignment

ACM Reference Format:
Jalal Ghadermazi, Ankit Shah, and Sushil Jajodia. 2024. Machine Learning and Optimization Framework for Efficient Alert Management in a Cybersecurity Operations Center. *Digit. Threat. Res. Pract.* 5, 2, Article 19 (June 2024), 23 pages. https://doi.org/10.1145/3644393

1 INTRODUCTION

Cybersecurity operations centers (CSOCs) are primarily responsible for protecting organizations from cyber threats. Equipped with state-of-the-art **intrusion detection systems (IDS)** or **security information and event management (SIEM)** systems, often powered by **machine learning (ML)** and **deep learning (DL)**

The work of Sushil Jajodia was partially supported by the U.S. Office of Naval Research Grants No. N00014-23-1-2132 and No. N00014-18-1-2670, and by the U.S. National Science Foundation Grant No. CNS-1822094.
Authors' addresses: J. Ghadermazi and A. Shah, University of South Florida, 4202 E. Fowler Avenue, Tampa, Florida 33620, USA; e-mails: jghadermazi@usf.edu, ankitshah@usf.edu; S. Jajodia, Center for Secure Information Systems, George Mason University, 10401 York River Road, Fairfax, Virginia 22030-4422, USA; e-mail: jajodia@gmu.edu.

ACM 2576-5337/2024/06-ART19
https://doi.org/10.1145/3644393

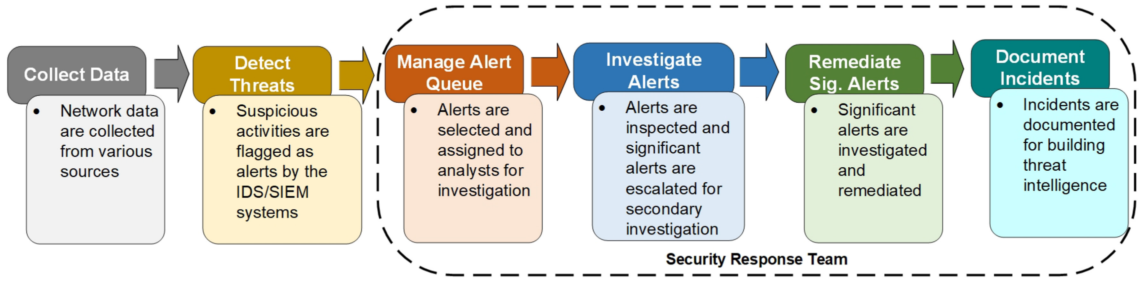

Collect Data
Network data are collected from various sources
Detect Threats
Suspicious activities are flagged as alerts by the IDS/SIEM systems
Manage Alert Queue
Alerts are selected and assigned to analysts for investigation
Investigate Alerts
Alerts are inspected and significant alerts are escalated for secondary investigation
Remediate Sig. Alerts
Significant alerts are investigated and remediated
Document Incidents
Incidents are documented for building threat intelligence
Security Response Team

Fig. 1. Typical alert-management process at cybersecurity operations centers (CSOCs).

algorithms, CSOCs continuously monitor network traffic and generate alerts upon detecting suspicious activities [5–7]. Concurrently, adversaries possess advanced toolchains, enhancing their ability to generate cyber-attacks in scale as well as sophistication. As a result, CSOCs face the challenge of investigating a large number of suspicious activities flagged by their detection mechanisms in a timely manner to identify and remediate attacks.

Security response team (SRT) within a CSOC is responsible for thoroughly investigating and timely mitigating these alerts to minimize the damage from cyber-attacks [21]. IDS/SIEM alerts awaiting investigation are assigned to SRT personnel using an organization-specific triage and assignment process. Cybersecurity analysts perform two levels of alert inspections: primary and secondary. The primary inspection aims to identify significant alerts among the flagged suspicious activities, typically requiring a few minutes to examine each alert. Significant alerts are escalated, and then sent for secondary inspection for further investigation and remediation of the attack incident, a process that may take several hours to days to complete. The findings from these investigations are documented to develop threat intelligence and response procedures for future incidents. Figure 1 shows the typical alert-management process followed by a CSOC.

The effectiveness of a CSOC in protecting the organization from cyber threats is directly tied to the performance of its SRT. However, a notable issue arises due to a severe imbalance in the number of alerts that are generated for investigation and the number and types of security analysts available in a work shift to efficiently perform these investigations. As a result, a significant backlog of uninvestigated alerts is created, posing a substantial risk of network exploitation for the organization. Recent research studies have examined this problem from both analyst-centric and alert-centric perspectives, aiming to address this critical challenge faced by organizations of various sizes.

Ganesan et al. [14, 15] and Shah et al. [30] presented mathematical models to optimally schedule regular and on-call analysts to meet the demands of expected alert workloads with minor deviations across various shifts in 14-day work cycles. Altner et al. [3] and Chen et al. [9] studied scheduling of analysts under different probabilistic scenarios. Ganesan et al. [14] and Shah et al. [29, 32] developed optimization models to allocate analysts to strategically placed sensors in computer networks, ensuring an even distribution of alert workload and minimizing the risk of unanalyzed alerts at the end of work shifts. These studies considered analyst characteristics such as experience (seniority), tooling proficiency, and credentials in their scheduling and sensor allocation models. Other notable studies presented innovative approaches to reduce the alert workload for analysts, utilizing data-driven and learning-based techniques [4, 9, 13, 25, 26, 33, 34, 37]. Shah et al. [34] developed a **reinforcement learning (RL)** model to intelligently discard alerts from the investigation queue, while Chen et al. [9] employed supervised ML to identify and remove potential false alerts. In another study, Shah et al. [33] presented an alert prioritization strategy based on organization-specific factors and developed an optimization model to maximize the selection of alerts for investigation in each work shift. We present a detailed review of the literature in the next section. Notably, these studies did not consider identification and grouping of similar alerts and matching them with analyst skills. In particular, none of these studies took into account the specific skill sets and familiarity (attributes) of the analysts, such as their expertise in investigating particular types of alerts, **operating systems (OS)**, machines, or assets, before assigning them to alerts.

The existing literature overlooks two crucial factors that hinder the SRT's ability to conduct efficient alert investigations. (i) Repeatedly reviewing unrelated alert logs can strain the cognitive abilities of security analysts, resulting in inefficient alert investigations [42]. (ii) The analyst-to-alert assignment decisions are made without matching their respective attributes. This means that an analyst who excels in handling Windows OS-related alerts may spend more time to thoroughly investigate an alert related to Linux OS, directly impacting the alert queue length in sub-optimal pairings. Our study aims to fill these gaps in the literature by proposing a novel framework that groups alerts into clusters based on similarities in their features and then matches the attributes of the formed clusters of alerts to the skill sets of analysts for alert investigation assignment decisions.

The primary contribution is the development of a novel ML- and optimization-enabled framework that forms clusters of alerts and assigns them to analysts with matching skills for efficient alert management. The clustering of alerts is achieved with the development of an unsupervised ML model, and the alert investigation assignment decisions are made with the development of a mathematical model. The sequential execution of these components in the framework enables a CSOC to dynamically optimize alert investigations at the start of a work shift based on the number and types of analysts reported to work and the number and types of alerts waiting in the investigation queue. An additional distinctive feature of this framework is its automated feedback component, which utilizes a rule-based human-centric approach to identify and filter out a subset of benign alerts through historical investigations conducted at the CSOC. The insights obtained from the execution of the framework on a real-world data set validates the need for matching the skills of analysts to those required by alerts for efficient alert investigation. In addition, CSOCs can identify new skills required in their SRTs obtained from the key features of the formed clusters. They can then use this information to better staff/hire their workforce and to provide training to enhance the skill sets of their analysts.

The remainder of this article is presented in the following order. Section 2 provides a review of the relevant literature in alert management in CSOCs. In Section 3, the novel ML and optimization framework is presented, along with the description of its components. Section 4 describes the experimental setup and presents the results obtained from each component of the framework. Section 5 provides a comparison of our methodology with another alert-management approach from recent literature to demonstrate the effectiveness of our framework. Finally, Section 6 discusses the conclusions and future directions.

2 LITERATURE REVIEW

An efficient alert management at a CSOC involves making critical decisions that include scheduling the appropriate types and amounts of resources (analysts) who have varying levels of experience and technical skills, as well as optimizing the alert investigation process. The goal of the SRT in alert-management process is to minimize the organization's risk from cyber threats while considering resource availability, work schedules, and other organization-specific constraints. The alert-management process is examined from various perspectives in the literature, including analyst scheduling and task assignment, team formation, and alert prioritization. Next, we present a review of the recent literature from these aspects, followed by the observed gaps.

One of the key focus areas in alert management has been the optimization of cybersecurity analyst scheduling and their allocation to sensors that generate the alerts. This optimization aims to achieve different objectives, such as workload balancing and risk minimization [3, 14–17, 29, 32]. Mathematical models have been developed to address these optimization challenges. For instance, Ganesan et al. [14] presented a model for the optimal scheduling of regular analysts and their allocation to sensors. Additionally, they proposed dynamic scheduling techniques for the cyber workforce that includes on-call analysts, aiming to minimize the risk of unanalyzed significant alerts remaining at the end of a shift [15]. To further optimize the alert-management process, Ganesan et al. [18] proposed an optimization modeling framework that considered historical and predicted demand patterns for alert analysis over a 14-day work cycle. This framework selected additional on-call analysts required for a shift and provided the optimal allocation of all required analysts on a day-to-day basis. Alert-management strategies are also explored in the literature, in which sensors are first organized into clusters, and then analysts are assigned to investigate alerts generated by these sensors. Analysts are matched with sensor clusters based on factors such as experience, tooling, and credential levels [14, 29, 32], without taking alert characteristics into account.

Another crucial aspect is team formation for SRT, which involves selecting groups of analysts who work together to investigate and mitigate alerts. Killcrese et al. [23] identified the first responders, who perform triage analysis [11], and the alert handlers as the core members of these teams. Effective team formation is recognized as a critical factor in determining the success of a CSOC by Steinke et al. [36]. Recent studies have focused on having the right mix of experience and tooling proficiency levels among analysts when forming teams [32, 35]. Shah et al. [35] presented a novel team formation framework that integrates optimization, simulation, and scoring methods to form effective teams utilizing these characteristics of the analysts.

Alert characteristics are primarily considered for prioritizing the alert investigation queue in literature studies [8, 19, 24, 40, 41, 43]. Apruzzese et al. [8] presented a pivoting detection algorithm based on network flow analyses, without relying on a priori assumptions about protocols and hosts. They introduced a prioritization algorithm that ranked detected paths according to a threat score, enabling security analysts to focus on the most suspicious pivoting tunnels. Experimental assessments demonstrated the proposal's higher accuracy and performance compared to related algorithms. Gupta et al. [19] proposed a novel event classification approach, identifying features through graphical analysis and utilizing a deep neural network model for classification. Their experimental evaluation with real CSOC event log data yielded promising results in terms of classification accuracy. Vidović et al. [41] introduced a method employing a learning-to-rank algorithm to rank device reports, aiding end-users in detecting higher-priority alarms more easily. Their incremental training approach showed an upward trend in model accuracy across various testing scenarios. An important challenge in alert prioritization is that adversaries may take advantage of the alert prioritization rules to evade detection, specifically by mounting attacks that trigger alerts that are less likely to be investigated be the security personnel. Tong et al. [40] presented an innovative approach to computing a policy for alert prioritization using adversarial reinforcement learning, addressing the challenge of adversaries exploiting prioritization rules. Laszka et al. [24] modeled alert prioritization with adaptive adversaries using a Stackelberg game, offering an approach to compute the optimal prioritization of alert types, considering the potential for adversaries to exploit prioritization rules for evading detection. Yen et al. [43] developed the Beehive framework that automatically extracted features from security log data in large enterprises. They identified 15 features to characterize outbound communications and employed clustering techniques to detect infected hosts.

In a recent study by Shah et al. [33], they introduced a **quantitative value function hierarchy (QVFH)** method to calculate the risk score of alerts. Their method incorporates alert characteristics and organization-specific factors to compute this score. Thereafter, a mathematical model is employed to select alerts for investigation, taking into account the total amount of resources that are available. To quantify and monitor the **level of operational effectiveness (LOE)** of a CSOC, Shah et al. [31] presented an average total time for alert investigation metric. To dynamically optimize the LOE of a CSOC under uncertainty during the 14-day work cycles, the authors in Reference [30] proposed a dynamic decision-making model using RL. The results demonstrated that a ML and optimization model can assist CSOC managers in making better decisions compared to current practices in determining when and how many resources to allocate for alert investigation under adverse conditions. It is worth noting that the alert selection process in these literature studies does not take into consideration the specific skill sets or attributes of the analysts.

Researchers have also directed their attention towards improving the efficiency of cybersecurity analysts [12, 38, 39], recognizing its impact on the alert-management process. Recent literature has explored the setup of the security personnel workforce, taking into account their unique characteristics such as experience levels and salary costs, within the context of IDS alert management [3, 14, 33]. Effective cybersecurity requires precise intrusion analysis, but the pervasive issue of false alerts poses a significant challenge. Various studies [4, 13, 25, 26, 37] aim to address distinct aspects of false alerts generated by IDS. Ndichu et al. [26] introduced a bidirectional methodology to combat severe class imbalance in security alert data analysis. Leveraging an ensemble of three oversampling techniques, this approach generates high-quality synthetic positive samples while efficiently removing noisy negative samples through a data subsampling algorithm. Experimental results showcase notable improvements in recall and false positive rates, signaling potential for enhanced AI-assisted security operations. Aminanto et al. [4] proposed an unsupervised alert screening scheme, utilizing the isolation forest method to manage overwhelming alert logs. This scheme leverages temporal information to identify distinct characteristics for each period, reducing the number of vast threat alerts and laying the groundwork for combating alert fatigue, thereby elevating the overall quality of service in cybersecurity operations.

McElwee et al. [25] conducted a case study introducing the federated analysis security triage tool prototype, applying ML to streamline the initial triage of security alerts within the Department of Defense. This tool integrates TensorFlow, Elasticsearch, and Kibana, providing an alternative approach for cyber defense analysts by efficiently categorizing and summarizing tens of thousands of daily alerts. Su et al. [37] proposed a scheme leveraging kernel density estimation to automate false positive alert filtering, demonstrating a significant performance improvement of 34% to 62% compared to other algorithms. Feng et al. [13] developed a user-centric machine learning framework for real enterprise cybersecurity operation centers, addressing typical data sources, workflow, and strategies for leveraging and processing data sets to construct an effective machine learning system. Together, these studies contribute diverse methodologies to reduce manual burdens, enhance accuracy, and improve overall efficiency in cybersecurity operations.

In summary, existing literature has focused on establishing the cybersecurity workforce, forming optimal teams of analysts, and scheduling them efficiently to meet the requirements of covering certain characteristics, such as tools and experience levels across work shifts. Analysts are assigned to specific sensors for alert investigations, considering their experience, tooling proficiency, and credential levels. The specific skill sets or attributes of analysts, such as their expertise in alert types, OS, machines, or assets are not considered when assigning them to the alerts. Also, these studies do not cluster alerts with similar attributes and assign them directly to matching analyst attributes for efficient investigations. Similarly, alert prioritization and selection schemes have been explored, but without considering resource attributes. In this study, our research focus is on improving the alert-management process, which involves better managing the alert queue of investigation with an efficient utilization of analysts available during the work shifts. We propose a novel ML and optimization framework that addresses the aforementioned gaps in the CSOC alert-management process. Our approach involves grouping similar alerts to clusters using unsupervised learning techniques and optimally assigning them to analysts with matching attributes using mathematical programming, reducing cognitive overload of the analysts and enhancing the overall efficiency of the alert-management process.

3 ALERT-MANAGEMENT FRAMEWORK

Our proposed **machine learning and optimization (ML+OPT)** framework for alert management comprises two main components: (i) the ML component that forms clusters of alerts based on similar features by developing unsupervised learning models and (ii) the optimization component for analysts to alert clusters assignment by developing a mathematical program that matches their respective attributes for investigation. Figure 2 shows the ML+OPT framework for an efficient alert management in CSOCs. Next, we describe each component in detail.

3.1 Machine Learning Component for Alert Cluster Formation

The ML component receives input in the form of alert data, typically presented in structured and machine-readable formats, such as the common event format (in the case of a SIEM system) used for logging

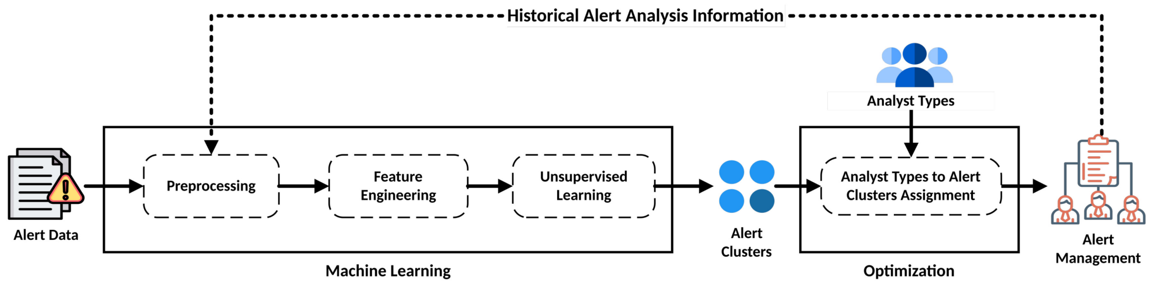


Fig. 2. Machine learning and optimization (ML+OPT) framework for alert management.

security-related issues. Alternatively, it may arrive in the form of plain text e-mails originating from an external information-sharing center responsible for monitoring the organization's network traffic. To facilitate analysis, the alert data undergoes preprocessing and feature engineering steps to generate clusters of related alerts.

3.1.1 Preprocessing. The preprocessing step helps transform the alert data into a standardized form, especially in the case of receiving alert data in a text-based (e.g., an e-mail) format. In this step, the relevant information is extracted from the alert data using a parser. This parser interprets the incoming alert data, typically in the form of text data that contains information about the alert, and then converts this data into structured alert data. By converting the raw data into a structured format, the parser ensures that the alert data is standardized and ready for further analysis and investigation by the analysts. Various techniques are applied during the preprocessing stage, including data cleaning, the removal of irrelevant or duplicate information, and information extraction. Information extraction involves identifying and extracting relevant pieces of information that can be utilized in the next step for generating meaningful features.

This step also takes into account the information obtained from recent alert analysis. Notably, we prioritize human-centric knowledge using a rule-based approach and do not use another ML/DL-based classifier that might override decisions made by an ML/DL-based IDS. The feedback process within the alert-management framework incorporates analyst-centric information about the alerts to automate the identification and discarding of redundant alert data. As part of this process, relevant information, such as the affected host IP and machine type, is stored in a database each time an alert is investigated. This information is used in the preprocessing step, which employs techniques like regex to extract host IP information from incoming alerts. These IP addresses are then cross-referenced with those from completed analyst investigations. If there is a match, indicating that the current alert pertains to a host IP that has already been addressed, then our automated process discards the redundant (benign) alert. Furthermore, our automated process considers indicators like the machine patching status. For example, if the historical investigation data shows that a particular host IP has already been patched or is currently undergoing remediation by analysts, then it serves as an indicator that alerts related to that host IP can be safely discarded as redundant. It is essential to emphasize that the feedback information is specifically applied in the work shifts immediately following the one that has just concluded. For instance, an alert might be generated while a machine is undergoing the patching process and the incident is in the midst of remediation. In this context, the feedback from each work shift is used only in the subsequent work shifts and not for extended time periods. This approach with a an upper bound for feedback time also helps address the potential issues that may arise if the patching fails. This data-driven approach streamlines the alert-management workflow and reduces the analysts' workload by eliminating benign alerts associated with previously addressed issues, enabling them to focus on new alerts.

3.1.2 Feature Engineering. Feature engineering involves creating new features from the information extracted in the previous step that can be passed to the unsupervised ML model in the next step. The creation of relevant and informative features is a crucial step that requires domain knowledge and feedback from the **subject matter experts (SMEs)** at the CSOC. Based on our discussions with a large collaborating organization that maintains an internal CSOC and a recent study from literature [33], we identified three main types of features that can be constructed from the extracted features, which are described as follows:

- **Alert-based** feature engineering involves the creation of features from alerts that can provide valuable information about the threats or the machines. Examples of such features include types of attacks (malware, DDoS, phishing, and others) and OS of the affected machines (Windows, Mac, or Linux).
- **Time-based** feature engineering involves determining if there exists a recent history of alerts issued for the affected machine in the alert data. In some cases, it may be more important to investigate newly affected machines to prevent the spread of an attack to other parts of the organization's network. In other cases, it may be more important to investigate machines that have frequently showed up in alerts to identify potential vulnerabilities in them.
- **Location-based** feature engineering involves creating features that identify the location of the affected machines in the network. Using network location as a feature may assist the SRT in identifying alerts from certain zones, in which their critical assets are placed.

3.1.3 Unsupervised Learning. Unsupervised learning is a ML approach that involves analyzing data without any predefined labels or target variables to uncover patterns and insights from the data points. Clustering is a widely used unsupervised learning technique that involves grouping similar data points into clusters based on certain similarity metrics. Some of the most commonly used clustering methods are listed below:

- Hierarchical clustering: It is a method that involves creating a hierarchy of clusters by iterative merging or splitting of clusters based on their similarity. This method can be agglomerative, where each data point starts as its own cluster and clusters are then iteratively merged, or divisive, where all data points start in one cluster and are then iteratively split.
- Centroid-based clustering: It organizes the data into non-hierarchical clusters, in contrast to hierarchical clustering defined above. The most widely-used centroid-based clustering algorithm is k-means that involves partitioning the data points into k number of clusters based on their proximity to each other.
- Density-based clustering: It is a method that involves identifying areas of high density in a data set and partitioning the data based on these identified areas. The most popular density-based clustering algorithm is DBSCAN [28], which groups data points into clusters based on their density and connectivity.
- Model-based (Distribution-based) clustering: It involves fitting a statistical model to the data and using this model to partition the data into clusters. The most popular model-based clustering algorithm is Gaussian mixture models, which assumes that the data points are drawn from a mixture of Gaussian distributions.

The selection of clustering method to group alerts depends upon the properties of the data set. Experimenting with different clustering methods and using appropriate metrics, such as elbow method, silhouette score, or coherence, is important to determine the best approach for an organization-specific use case. We will develop and compare algorithms from each of the above-mentioned categories. These algorithms are Fuzzy C-Means (Centroid-based clustering), K-Modes (Model-based clustering), Hierarchical clustering, and DBSCAN (Density-based clustering). A cost metric, known as the **within-cluster sum of squares (WCSS)** will be used to compare their performances. WCSS represents the cumulative sum of squared distances between data points and their respective cluster centers. Consider $C = [C_1, C_2, \ldots, C_k]$ be the clusters of the data set $X = [x_1, x_2, \ldots, x_n]$, with $C(x_i)$ denoting the cluster assignment for x_i. The WCSS value will be computed using Equation (1):

$$\text{WCSS} = \sum_{\forall x_i} \left\| x_i - mean(C(x_i)) \right\|^2 . \tag{1}$$

The cost values are used as a measure of the quality of the clustering algorithm, with lower costs indicating more compact and well-defined clusters. Based on their performances using the WCSS metric, the respective clustering algorithm will be chosen. To identify the important alert attributes that distinguish the properties of the alert data within each cluster, cluster profiling techniques, such as visualization and rule mining, are used.

3.2 Optimization Component for Assignment of Analysts to Alert Clusters

For efficient alert handling in a CSOC, it is important to ensure that the alerts are investigated by security analysts with appropriate skills. However, the skill sets of analysts in a CSOC can vary significantly across different analyst experience levels (types) that have been scheduled for the work shifts. Hence, the objective of the optimization component is to match the skill requirements of alert clusters, obtained from the ML component, to the skill sets of the available security analyst types in the shift.

A CSOC hires a mix of analyst types, including junior, intermediate, and senior experience levels, to maintain the quality of analysis [3, 14, 15]. In this work, we assume that the mix of analysts is maintained during each shift. The time required to thoroughly investigate an alert depends on the attributes of the alert cluster and the skill set of the assigned analyst. This means that if there is a complete match between the attributes of the alert cluster and the assigned analyst, the expected investigation time is optimal. However, in the case of a partial or no match, the investigation time proportionally increases. The objective is to pair analyst types and alert clusters such that the skills required for alert investigation in the cluster and the skill types of the analysts are maximally matched. Next, we present the mathematical formulation of this problem, including input parameters, decision variables, objective function, constraints, and the outputs of the optimization model.

- **Sets:**
  C: Set of alert clusters
  A: Set of analyst types
  S: Set of skills
- **Indices:**
  c: Cluster identity ($c \in C$)
  a: Analyst type identity ($a \in A$)
  s: Skill identity ($s \in S$)
- **Input parameters:**
  $Y_{a,s}$: Indicates if the analyst type a has skill s,

$$Y_{a,s} = \begin{cases} 1 & \text{if } a \in A \text{ has skill } s \in S; \\ 0 & \text{otherwise.} \end{cases}$$

  $Z_{c,s}$: Indicates if the cluster c requires skill s,

$$Z_{c,s} = \begin{cases} 1 & \text{if } c \in C \text{ requires skill } s \in S; \\ 0 & \text{otherwise.} \end{cases}$$

  E_c: Total expected time required for investigating alerts in cluster $c \in C$.
  T_a: Total time available by analyst type $a \in A$. For example, if there are two analysts of type I ("Senior") and the shifts are scheduled for 8 h, then the total time available for analyst type I is $2 \times 8 \times 60 = 960$ min.
  P: Discount factor for computing the lower bound for total time required for each cluster. The factor value is organization-specific and is determined by the CSOC.
- **Decision variables:**
  $x_{a,c}$: Indicates if analyst type a is assigned to cluster c,

$$x_{a,c} = \begin{cases} 1 & \text{if } a \text{ is assigned to cluster } c, \\ 0 & \text{otherwise.} \end{cases}$$

  d_c: Number of unmatched skills (deviations) for cluster c.
  $r_{a,c}$: Time allocated by analyst type a to cluster c.

— **Objective function:**
The objective is to maximize meeting the skill requirements of each cluster, thereby minimizing the number of deviations in these requirements across all the clusters:

$$w = Min \sum_{c=1}^{|C|} d_c. \tag{2}$$

— **Constraints:**
The constraint for calculating the number of deviations for each cluster is given by

$$\sum_{a=1}^{|A|} \sum_{s=1}^{|S|} |Y_{a,s} - Z_{c,s}| * x_{a,c} \leq d_c, \qquad \forall c \in C. \tag{3}$$

The constraints for ensuring that the analyst assignment satisfies the time requirement for alert investigation in each cluster are given by

$$P * E_c \leq \sum_{a=1}^{|A|} r_{a,c} \leq E_c, \qquad \forall c \in C, \tag{4}$$

$$r_{a,c} \leq x_{a,c} * T_a, \qquad \forall a \in A,\ c \in C. \tag{5}$$

The constraint for ensuring that the total time assigned by each analyst type across all cluster assignments does not exceed its available time is given by

$$\sum_{c=1}^{|C|} r_{a,c} \leq T_a, \qquad \forall a \in A. \tag{6}$$

— **Output:**
The output of the optimization model is the assignment of the security analysts to clusters of alerts for mitigation.

The research problem of matching analyst types to alert clusters for investigation is formulated as a mixed integer programming problem. Such problems fall under the NP complexity class [22, 27]. The complexity of this problem is $2^{|A|*|C|}$, and the mathematical model can be solved using specialized solvers like Gurobi [20] or CPLEX [10]. Algorithm 1 outlines the implementable steps of the optimization model.

4 EXPERIMENTS AND RESULTS

This section presents the numerical setup employed for conducting the experiments and the results obtained for each framework component. First, we describe the alert data that we used for the experiments, followed by the experiment details and the results of the ML and optimization models.

4.1 Alert Data

We investigated the alert data of a large collaborating organization that maintains an internal CSOC. The SRT of the CSOC receives security alert data generated by a single IDS security appliance using the Albert network monitoring and management system in the form of an e-mail report. Albert is a renowned IDS specifically tailored for U.S. State, Local, Tribal, and Territorial government organizations. This is a common format of receiving alert data using an information sharing and analysis center such as MS-ISAC [1]. The received e-mails also include threat intelligence information about emerging cybersecurity threats, vulnerabilities, and indicators of compromise that could impact the organization's network in the future. Advisory security updates are also part of the receiving e-mails at the CSOC. These e-mails are updates on cybersecurity news, trends, and events that may

ALGORITHM 1: Optimization algorithm for assignment of analysts to alert clusters.

```
Input  : Alert clusters, $C$; analyst types, $A$; skills, $S$; analyst type skill set, $Y_{a,s} \forall a \in A,\ s \in S$; alert cluster required skill
         set, $Z_{c,s} \forall c \in C,\ s \in S$; total expected time required for investigating alerts in cluster $c$, $E_c$; total time available
         by each analyst type $a$, $T_a$; discount factor, $P$.
Output : Assignment of the security analysts to clusters of alerts for mitigation, $x_{a,c}$ and $r_{a,c} \quad \forall a, c$.
1  /*Initiate a solution search using an integer programming solver*/ repeat
2      for a set of $x_{a,c} = 1$ and $r_{a,c}$, /*Potential solution in a search*/ check for feasibility: do
3          $\sum_{a=1}^{|A|} \sum_{s=1}^{|S|} |Y_{a,s} - Z_{c,s}| * x_{a,c} \leq d_c \quad \forall c \in C$
4          $P * E_c \leq \sum_{a=1}^{|A|} r_{a,c} \leq E_c \quad \forall c \in C$
5          $r_{a,c} \leq x_{a,c} * T_a \quad \forall a \in A,\ c \in C$
6          $\sum_{c=1}^{|C|} r_{a,c} \leq T_a \quad \forall a \in A$
7      end
8      if Feasible then
9          $w = Min \sum_{c=1}^{|C|} d_c$
10     end
11 until Stopping criteria /*Optimal value for w is found*/;
12 return $x_{a,c}$ and $r_{a,c} \quad \forall a, c$.
```

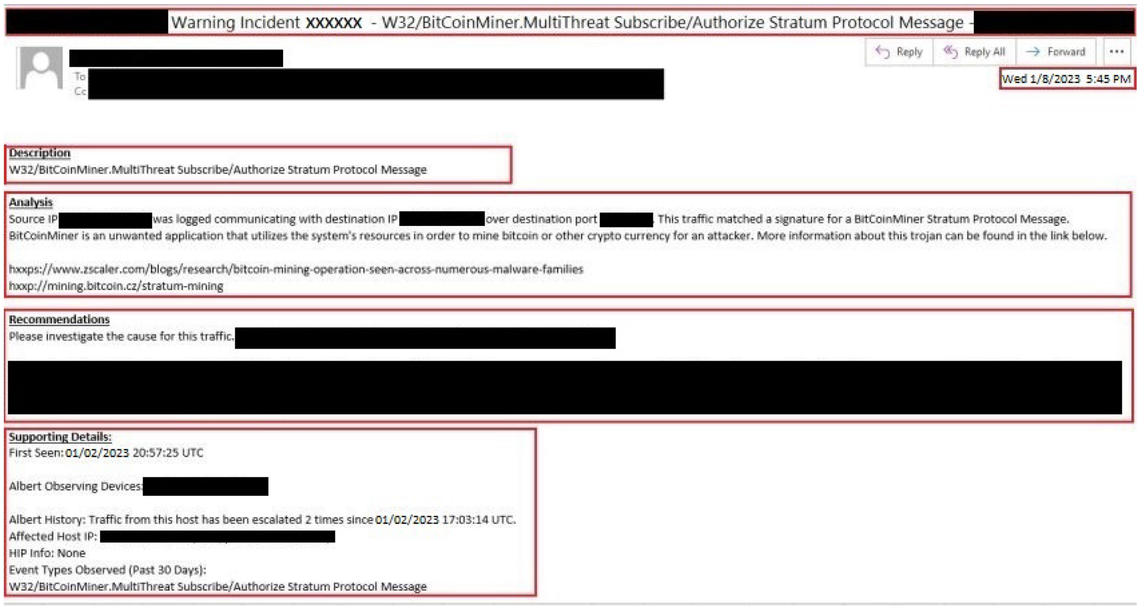
impact the organization. Our preprocessing steps, including data cleaning and information extraction, identifies these e-mails and removes them from the queue of alerts that need investigation.

The security analysts conduct alert investigations on a shift basis, with reported alerts assigned for investigation at the start of an 8-h shift. We were provided with personal storage table (.pst) files containing alerts for each 8-h period. For each 8-h shift, we received the alert data in a single .pst file. In the first experiment (indicating an 8-h work shift), the .pst file contained 1,561 e-mails. Additionally, for the scalability experiment in Section 5.1, we received 21 more .pst files containing 29,789 e-mails in total. The total number of alert data we obtained amounted to 931 (for the first experiment) and 16,002 (for the scalability experiment), summing up to 16,933 security alerts. This number represents the actual alert e-mails following the preprocessing of the .pst files, which involved the removal of redundant and irrelevant e-mails. We first present the numerical setup of one .pst file (indicating an 8-h work shift). The other 21 .pst files were setup similarly for further experiments in Section 5.1 Figure 3 illustrates a sample of the alert data.

4.2 Experiments and Results for the ML Component

Next, we describe the experimental details and the results obtained for the ML component of the framework, which is divided into three steps.

4.2.1 Preprocessing. We parse the .pst file to extract e-mail data. Various techniques are applied in the preprocessing stage, including data cleaning and information extraction. Data cleaning involves the removal of irrelevant or duplicate information from the investigation queue, which is used to form clusters of alerts. Irrelevant information, from the cluster formation perspective, includes threat intelligence and advisory e-mails, and the duplicate information are e-mails regarding the same affected IPs that trigger the same incident repeatedly (e.g., a denial-of-service incident). We extract key fields of each e-mail, highlighted in red boxes in Figure 3, including the e-mail title, e-mail date and time, description, analysis, recommendations, and supporting details. The textual data in the e-mail fields contain important information about the alerts. We apply information extraction techniques to extract relevant information for each alert. In this study, we employ a rule-based method, given the template-like structure of the e-mails. This approach relies on predefined patterns or rules, such as regular expressions [2], to identify and extract specific information. We used the regex module, which has an API


[redacted] Warning Incident XXXXXX - W32/BitCoinMiner.MultiThreat Subscribe/Authorize Stratum Protocol Message - [redacted]

[redacted]

To
Cc [redacted]

Wed 1/8/2023 5:45 PM

Description
W32/BitCoinMiner.MultiThreat Subscribe/Authorize Stratum Protocol Message

Analysis
Source IP [redacted] was logged communicating with destination IP [redacted] over destination port [redacted]. This traffic matched a signature for a BitCoinMiner Stratum Protocol Message.
BitCoinMiner is an unwanted application that utilizes the system's resources in order to mine bitcoin or other crypto currency for an attacker. More information about this trojan can be found in the link below.

hxxps://www.zscaler.com/blogs/research/bitcoin-mining-operation-seen-across-numerous-malware-families
hxxp://mining.bitcoin.cz/stratum-mining

Recommendations
Please investigate the cause for this traffic. [redacted]

[redacted]

Supporting Details:
First Seen: 01/02/2023 20:57:25 UTC

Albert Observing Devices: [redacted]

Albert History: Traffic from this host has been escalated 2 times since 01/02/2023 17:03:14 UTC.
Affected Host IP: [redacted]
HIP Info: None
Event Types Observed (Past 30 Days):
W32/BitCoinMiner.MultiThreat Subscribe/Authorize Stratum Protocol Message

Fig. 3. Sample of the reported alert data.

Table 1. Complete List of Extracted Features from the Alert E-mails

Feature	Extraction Field	Description
Incident ID	Description	The unique identifier for the incident.
Destination IP	Analysis	The IP address identified as the destination of the alert, indicating the affected device (machine).
Source IP	Analysis	The IP address identified as the source of the alert, indicating the origin of the suspicious network activity.
Source Port	Analysis	Source port number associated with the incident.
Destination Port	Analysis	Destination port number associated with the incident.
Protocol Type	Analysis	Type of protocol used (TCP or UDP).
Alert Name	Description	Name or description of the incident.
Observing Devices	Supporting Details	The security appliance or system involved in detection of the incident.
Website Visited (if applicable)	Description	Website visited as part of the incident.
User/Account Involved	Analysis	User or account associated with the incident.
Host/Device Involved	Analysis	Host or device related to the incident.
Related References	Analysis	References or additional context related to the incident.
Device History	Supporting Details	A history or timeline of the incident within the security appliance system.
Host Intrusion Prevention (HIP) Info	Supporting Details	Information related to the host's HIP status.
First Seen	Supporting Details	Additional information about the incident, including the first observed date and time.

compatible with the standard library re package in Python. Table 1 shows the complete list of the extracted features and their source in the alert e-mail data. During this process, we also identify and remove alerts, including duplicates, completed investigations, and ones with mitigated machine vulnerabilities. Note that we considered an upper bound of two days for this feedback, a parameter that could be customized by the CSOC based on their investigation and patching schedules. This step is performed using historical alert analysis information, as shown in the feedback loop in Figure 2. At the end of this preprocessing step, there were 931 alerts in the investigation queue.

Table 2. Description of the Engineered Features

Feature Type	**Feature**	**Value**	**Description**
Alert-based	OS	Windows, Mac, Linux	Indicates the OS associated with the machine in the alert.
	Alert Type	DDoS, Malware, Botnet, ...	Represents the specific threat type associated with the alert.
Location-based	Network Location	Zone-1, Zone-2, Zone-3	Indicates organization-specific network zone, in which the threat is found based on the destination IP address.
Time-based	Machine History	0, 1	Indicates whether there is a history of alerts generated for the affected machine (1) or not (0).

Table 3. Clustering Results of Different Algorithms Using the Cost (WCSS) Metric

Number of Clusters	Fuzzy C-Means	K-Modes	Hierarchical Clustering	DBSCAN
1	2,073	1,569	2,354	2,621
2	1,836	1,157	2,125	2,304
3	1,654	875	1,976	2,232
4	1,298	725	1,647	2,121
5	1,287	624	1,567	2,003
6	1,160	568	1,530	2,010
7	980	552	1,310	1,768
8	957	426	1,150	1,532
9	856	412	1,067	1,479

4.2.2 Feature Engineering. This step involves extracting features from the structured data obtained from the previous step for the alerts. We obtained other computer- and network-related information from the SMEs at the CSOC for feature engineering. Table 2 provides a summary of the key engineered features. Alert-based features included the OS and alert type, location-based feature included the location of the affected machine in the network as indicated by zones 1, 2, and 3, and time-based feature included a binary value of 1 or 0, indicating whether the affected machine had any alert history or not, respectively.

4.2.3 Unsupervised Learning. The objective of this step is to create clusters of alert data in an unsupervised manner using the feature set values from the previous step. We developed various clustering techniques, including Fuzzy C-Means, K-Modes, Hierarchical Clustering, and DBSCAN. The outcomes of these diverse clustering algorithms, employed to categorize alert data, are summarized in Table 3. The numbers within the table represent the associated costs for different cluster counts using the respective clustering techniques. Notably, the K-Modes algorithm exhibited superior cluster separation performance across all cluster counts, from cluster 1 to cluster 9, consistently demonstrating the lowest cost. The cluster profiling is explained after describing the approach to determining the optimal number of clusters to form.

To select the optimal number of clusters, we applied the elbow method. In this method, the optimal number of clusters is the number of clusters before the addition of another cluster that does not significantly reduce the cost. Figure 4 displays the elbow method results for selecting the optimal number of clusters. We found eight clusters to be an ideal choice (as shown in a red circle in Figure 4 from the K-Modes algorithm).

Next, we performed a visual profiling of the clusters to identify the characteristics of each cluster and their corresponding skills required from the analysts. We use a stacked bar chart, which displays the percentage of alerts of each feature value within each cluster. Figure 5 presents the profiling outcomes for each cluster based on different features. The clusters showed better segregation compared to other approaches based on the different

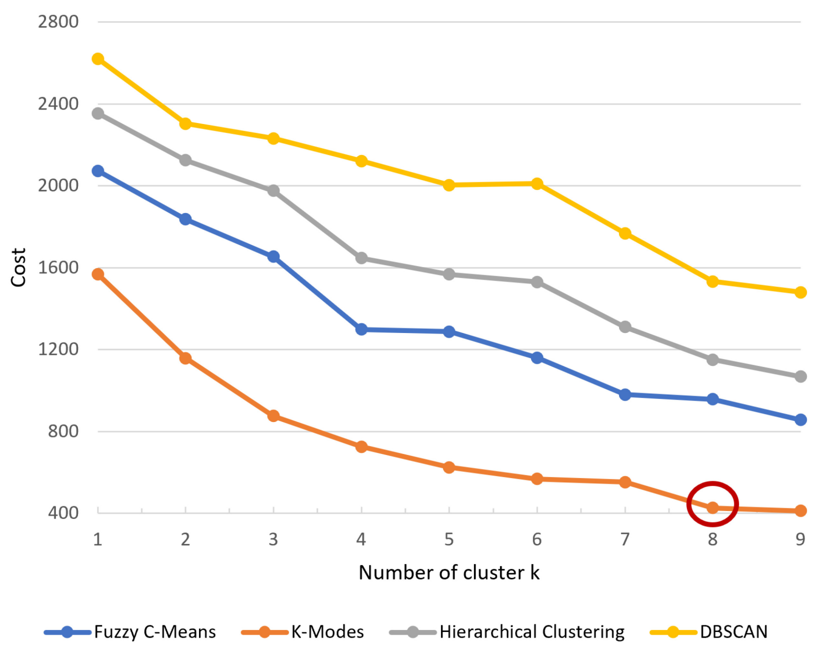


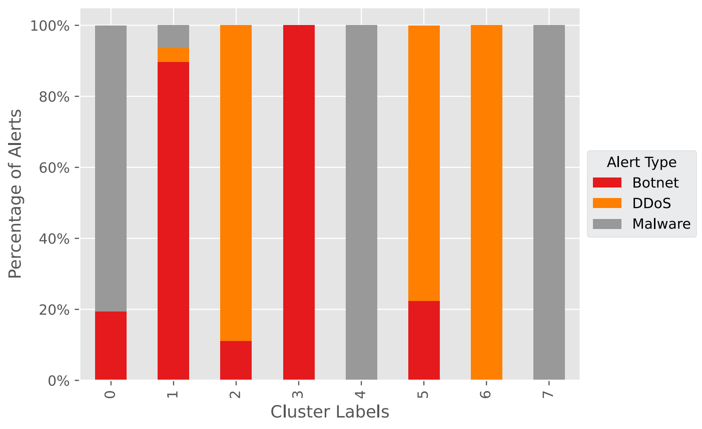
Fig. 4. Determining the optimal number of clusters using the elbow method.

categorical values of the features. For instance, cluster numbers 4 and 7 only contained the Malware attack type, while cluster numbers 3 and 6 contained the Botnet and DDoS attack types, respectively. Similar segregation among the clusters can be observed from the perspective of the other two features, OS and network location, in Figure 5. The time-based feature did not play a prominent role in segregating the clusters. Moreover, the time-based feature does not have a corresponding skill requirement from the analysts. Next, we present the experimental details and the results obtained for the optimization component of the framework.

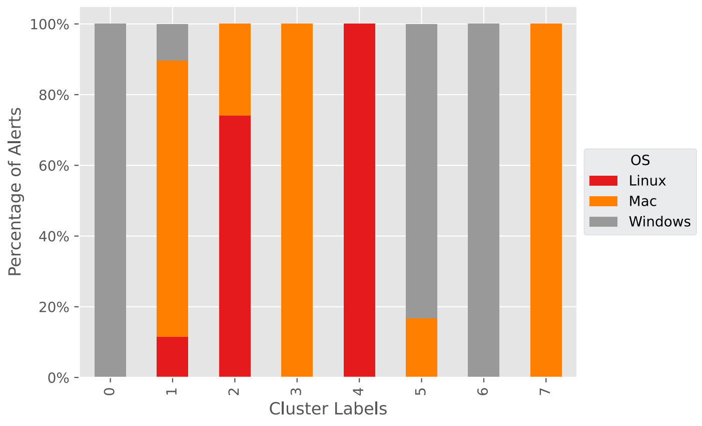
4.3 Experiments and Results for the Optimization Component

The cluster information from the ML component and the analyst information from the CSOC are presented to the optimization component as inputs. We transform the input information to a format that can be used for mathematical programming. Table 4 shows the skill requirements for the eight clusters ($|C| = 8$) based on the profiling results (see Figure 5) in a one-hot encoding format. The goal of the ML-based clustering process was to group alerts such that they are optimally segregated. However, achieving a perfect separation of feature values in clusters is not always possible. In cases where the perfect distinction between clusters for certain feature values does not exist, we use the feature with the largest proportion of alerts to indicate the respective skill required by that cluster.

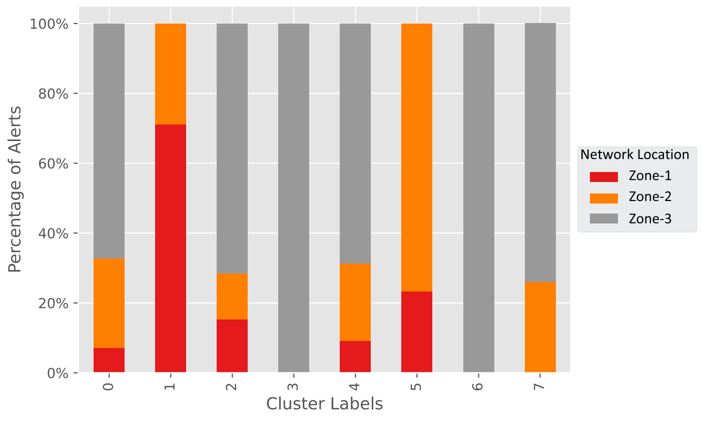
From our discussions with the CSOC and based on the literature studies [14, 15, 29], we consider three types of analysts: senior, intermediate, and junior. Within the context of this study, each of these groups possesses varying skill sets. It is essential to note that the categorization of analysts into these groups is primarily based on their expertise level and not based on their years of service. The attributes (skill sets) of the analyst types are given in Table 5, which can be obtained by the CSOC. As illustrated in Table 5, senior analysts demonstrate a broad and advanced skill set, enabling them to effectively handle a diverse range of alerts optimally. However, junior analysts exhibit more limited expertise.



(a) Alert Type



(b) OS



(c) Network Location

Fig. 5. Cluster profiling for clusters based on various features.

Table 4. Skills Required in Each Cluster ($Z_{c,s}$)

Cluster (c)	Skill(s) Required								
	Alert Type			OS			Network Location		
	Botnet	DDoS	Malware	Linux	Mac	Windows	Zone-1	Zone-2	Zone-3
0	0	0	1	0	0	1	0	1	0
1	1	0	0	0	1	0	1	0	0
2	0	1	0	1	0	0	0	1	0
3	1	0	0	0	1	0	0	1	0
4	0	0	1	1	0	0	0	1	0
5	0	1	0	0	0	0	0	0	1
6	0	1	0	0	0	1	0	1	0
7	0	0	1	0	1	0	0	1	0

Table 5. Attributes for Each Analyst Type ($Y_{a,s}$)

Analyst Type (a)	Skill(s)								
	Alert Type			OS			Network Location		
	Botnet	DDoS	Malware	Linux	Mac	Windows	Zone-1	Zone-2	Zone-3
Senior	0	1	1	1	1	1	1	0	0
Intermediate	0	1	0	1	0	1	0	1	1
Junior	0	0	1	1	0	0	0	0	1

The representation of analysts attributes in Table 5 might seem rigid with binary skillset values (1 and 0). However, in this study, an attribute value of 1 in Table 5 indicates that the analyst is highly skilled in a specific skill s and can investigate alerts requiring skill s optimally. Conversely, an attribute value of 0 does not imply an inability of an analyst to investigate the alerts requiring skill s but rather reflects the need for more time due to the analyst's lower level of expertise. In Section 5, we address the nuances of analyst skill by using what we refer to as "matching score" metric defined in Equation (7) during the investigation process, which accommodates varying levels of analyst skill (more or less experienced in each skill) in practice. The use of matching score mechanism within our framework allows us to establish investigation times based on the degree of alignment between an analyst's skills and the alert's skill requirements. Through our experiments conducted on the SRT personnel, we found out that the alert investigation time associated with the primary level of investigation is approximately 3 min, with a perfect match between the skills possessed by an analyst and the attribute of an alert. Other studies in the literature [3, 9] use a similar estimate of a few minutes for alert investigation, which we use as a baseline value in the experiments. The value of the input parameter E_c is calculated by multiplying the alert count in the cluster c with the estimated investigation time. Table 6 shows the expected time required for alert investigation for each cluster. In our experiments, we considered specific numbers of analysts for each analyst type per 8-h shift: two senior analysts, one intermediate analyst, and two junior analysts. The input variable T_a represents the total time available for each analyst type a, which is calculated by multiplying the number of analysts available for that type with the duration of the shift in minutes. Table 7 shows the total time available for alert investigation by each analyst type.

The discount factor, P, is an algorithmic hyperparameter used to tighten the lower bound on finding the optimal time allocated by the analysts to clusters for alert investigation, thereby assisting the solver in finding the optimal solution faster. It can be observed from Equations (5) and (6) that variable $r_{a,c}$ is not constrained by lower bounds. Without the inclusion of P, the model would fail to assign any analyst to any cluster, yielding

Table 6. Expected Time Required for Alert Investigation for Each Cluster (E_c)

Cluster	Number of Alerts	Time Required (min)
0	195	585
1	156	468
2	189	567
3	49	147
4	99	297
5	107	321
6	86	258
7	50	150
Total	931	2,793

Table 7. Total Time Available for Each Analyst Type (T_a)

Analyst Type	Number of Analysts	Time Available (min)
Senior	2	960
Intermediate	1	480
Junior	2	960
Total	5	2,400

an objective value of the number of deviations to zero, which is theoretically optimal. The value of P could vary based on the number of alerts, the number of analysts available in the shift, and analyst utilization required for the alert-management task. In practice, it must be set to a value greater than 0. The higher it is, without rendering Equation (4) infeasible, the faster the algorithm will find the solution. Based on the input parameters obtained from the CSOC (see Tables 6 and 7), the expected time required for investigating all clusters is equal to 2,793 min (total time required in Table 6), and the total time available for all analysts in an 8-h shift is 2,400 min (total time available in Table 7), indicating a shortage of manpower to investigate all alerts in an 8-h shift. Through a sensitivity analysis involving different values ranging from 0.50 to 0.90 in increments of 0.05, we identified $P = 0.85$ as the best value for solving the problem. For instance, when $P \geq 0.90$, the optimization model becomes infeasible, because the lower bound in Equation (4) would be $0.9 * (585 + 468 + 567 + 147 + 297 + 321 + 258 + 150) = 2,513$ min, exceeding the total time available of 2,400 min (as indicated in Table 7). Table 8 shows the assignment summary obtained upon executing the mathematical program (Algorithm 1). Note that one of the skills required by the clusters (1 and 3) was not present across the skill sets of the available analysts. Hence, the optimization model assigned an analyst type to those clusters based on the time availability. This information can be used by the CSOC to upgrade the skill set of the SRT through periodic training or hiring. To show the effectiveness of our approach, we compare the performance of the CSOC using the assignment results obtained from our framework with that of another alert-management method from recent literature.

5 PERFORMANCE COMPARISON AND ANALYSIS OF RESULTS

In this section, we compare the performance of our methodology with the **quantitative value function hierarchy (QVFH)** approach presented in Reference [33], which first triages alerts and then assigns them to analysts in a round-robin manner. Note that such an assignment scheme is primarily used in CSOCs, ensuring a fair distribution of workload among the available analysts. This triaging process involves a preprocessing step to calculate the composite risk scores of the alerts based on organizational factors. We use the information obtained from the alert data, including alert type, OS, criticality of the asset zones, and alert history to compute this score with the following weights assigned to these factors: $W_1 = 0.2$, $W_2 = 0.35$, $W_3 = 0.2$, and $W_4 = 0.25$, respectively

Table 8. Assignment Summary of Analysts to Alert Clusters ($x_{a,c}, r_{a,c}$)

Analyst Type	Assigned Cluster(s)	Time Allocated (min)
Senior	0	478
	2	482
Intermediate	4	253
	6	227
Junior	0	36
	1	398
	3	125
	5	273
	7	128
Total Allocated Time		2,400

Table 9. Average Investigation Time Per Alert for Different Matching Scores

Matching Score	Average Investigation Time (min)
0	6
1/3	4.5
2/3	3.5
1	3

(as used in Reference [33]). To make a fair comparison among these two approaches, we use the same data set, containing 931 alerts, obtained after the preprocessing step in the ML component of the framework.

To quantitatively compare our approach to the QVFH approach, we established a matching score metric in discussions with the CSOC. This matching score is defined in Equation (7), in which the denominator represents the number of skills required by an alert, and the numerator represents the degree of match between the skills of assigned analyst type and the skill requirements of the alert. As discussed in Section 3, a perfect match (i.e., a matching score of 1) between the analyst type's skill set and the alert's skill requirements is estimated to take an average of 3 min for investigation. However, if the matching score is less than 1, then the investigation time proportionally increases:

$$\text{Matching score} = \frac{|\text{Analyst skill set} \cap \text{Alert skill set}|}{|\text{Alert skill set}|}. \tag{7}$$

Table 9 presents the average investigation time (in minutes) per alert based on various matching scores. The average investigation times were derived from prior investigations conducted. To illustrate, during investigation rounds where analysts with partial or no matching of skills (2/3, 1/3, and 0) were assigned to handle alerts, the corresponding times for alert investigation were recorded, and an average time for each matching score was computed. These average alert investigation times were primarily utilized to quantify the advantages of aligning the skills of analysts with the attributes of alerts within CSOCs in our experiments. Table 10 shows the performance comparison of our approach with the QVFH method for the 8-h shift. The number of alerts investigated and the average time taken for investigation per alert are reported in this table for both these methods. Note that since QVFH method uses a random order of analysts for assigning alerts for investigation at the start, we simulated 100 runs of the experiment by assigning alerts to a different order of the analysts in each run. We used the average investigation times, as outlined in Table 9, to calculate the time expended in investigating alerts in the simulation. We report the best results obtained from these runs in Table 10 for the QVFH method. On an average, 28% more alerts were investigated using the ML+OPT method compared to the

Table 10. Performance Comparison between ML+OPT and QVFH Methods for the 8-h Shift

Method		Hour							
		1	2	3	4	5	6	7	8
QVFH	Number of alerts investigated	65	69	71	57	66	63	78	60
	Average investigation time per alert	4.61	4.34	4.22	5.26	4.54	4.76	3.84	5
ML+OPT	Number of alerts investigated	84	88	83	87	81	90	85	82
	Average investigation time per alert	3.57	3.41	3.61	3.44	3.7	3.33	3.52	3.65

Table 11. Number of Alerts Collected for Each Day and the Number of Irrelevant/Duplicate and Redundant Alert Data Obtained Using the Preprocessing Step and the Feedback Loop

Day	Total E-mails	Irrelevant/Duplicate	Irrelevant/Duplicate (%)	Redundant	Redundant (%)	Actual Alerts
1	4,481	434	9.6	1,563	34.8	2,484
2	4,226	403	9.5	1,458	34.5	2,365
3	4,129	375	9	1,609	38.9	2,145
4	4,801	425	8.8	1,833	38.1	2,543
5	3,892	325	8.3	1,536	39.4	2,031
6	4,013	381	9.4	1,454	36.2	2,178
7	4,247	397	9.3	1,594	37.5	2,256

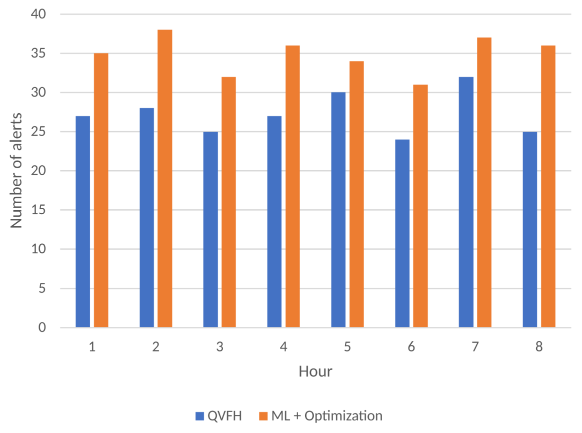
Table 12. Performance Comparison between ML+OPT and QVFH Methods for the One-week Period

		Day						
		1	2	3	4	5	6	7
Method	Number of new alerts assigned	2,484	2,365	2,145	2,543	2,031	2,178	2,256
QVFH	Total number of alerts for investigation	2,484	3,216	3,801	4,599	5,045	5,725	6,401
	Number of alerts investigated	1,633	1,560	1,745	1,585	1,498	1,580	1,644
	Average investigation time per alert	4.41	4.61	4.12	4.54	4.8	4.56	4.37
	Backlog	851	1656	2056	3014	3,547	4,145	4,757
ML+OPT	Total number of alerts for investigation	2,484	2,852	2,970	3,540	3,322	3,607	3,876
	Number of alerts investigated	1,997	2,027	1,979	2,249	1,893	1,987	2,005
	Average investigation time per alert	3.6	3.55	3.63	3.2	3.8	3.62	3.59
	Backlog	487	825	997	1,291	1,429	1,620	1,871

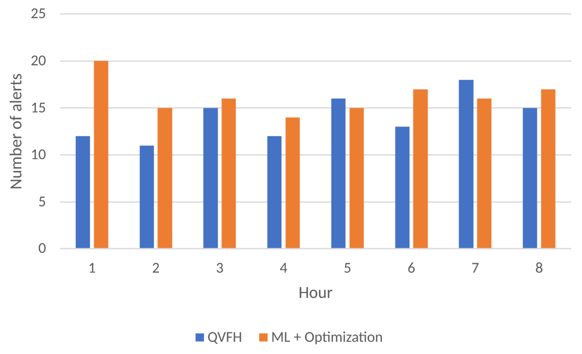
QVFH method. Our method also took, on an average, 23% less time to investigate an alert over the 8-h shift when compared to the QVFH method. Figures 6(a)–6(c) show the total number of alerts investigated by each analyst type for every hour of the shift using the two methods. It can be seen that all analyst types are able to investigate more alerts using the ML+OPT method compared to the QVFH method. By leveraging ML and optimization techniques, our approach streamlines the investigation process by maximizing the assignment of analysts to alerts with matching attributes.

5.1 Scalability Experiment for One-week Period

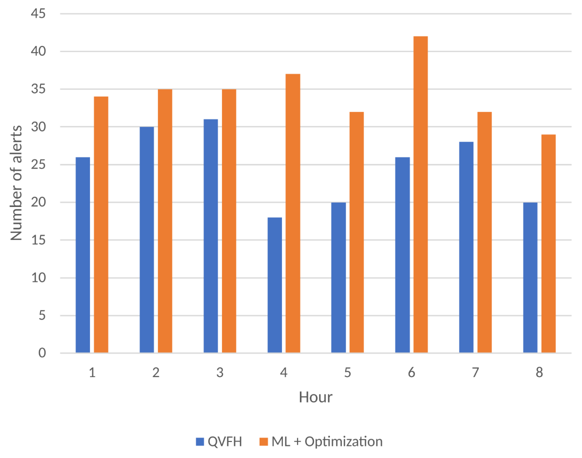
We conducted a numerical experiment by scaling it with multiple analysts, totaling 10 analysts, who worked in shifts throughout the week. Each day of the week, alert data was collected, clusters were formed, and analysts



(a) Senior

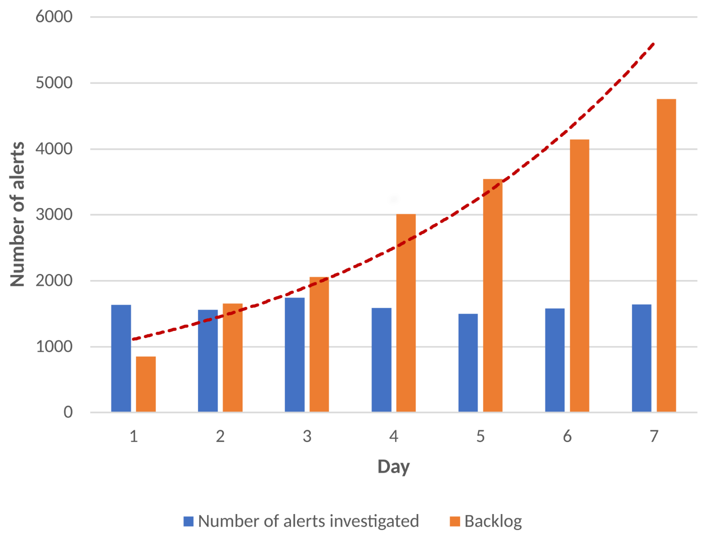


(b) Intermediate

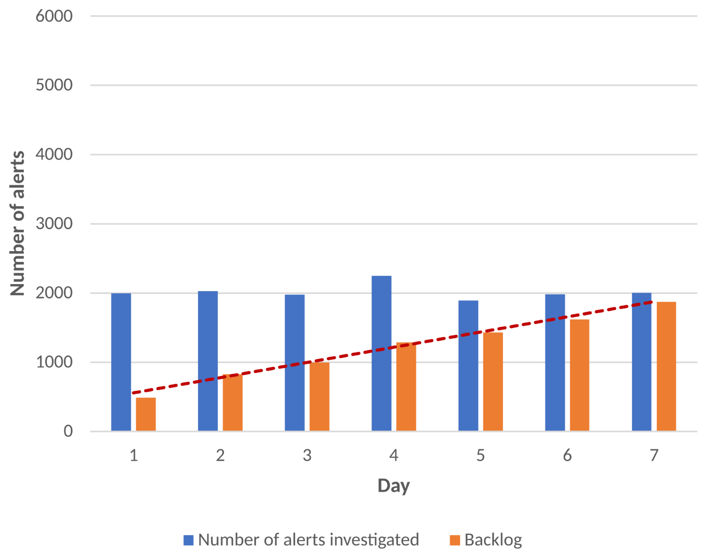


(c) Junior

Fig. 6. Total number of alerts investigated by each analyst type in the 8-h shift using ML+OPT and QVFH methods.



(a) QVFH



(b) ML+OPT

Fig. 7. Total number of alerts investigated and backlog size during the one-week period using ML+OPT and QVFH methods.

were assigned to clusters based on their skill requirements for each shift. The proportion of analysts per type remained consistent across the daily shifts. As described earlier in Section 4, the preprocessing step encompasses data cleaning, removal of irrelevant or duplicate information, information extraction, and a feedback mechanism that incorporates information from recently completed alert investigations. Table 11 displays the total number of actual alerts assigned to analysts after applying the preprocessing techniques and the feedback loop. This feedback system effectively identified and removed approximately 30–40% of benign alerts on a daily basis. Table 12 presents the performance comparison between the ML+OPT and QVFH methods. The results displayed in the table demonstrate a significant improvement when using the ML+OPT method. At the end of the week, the backlog of alerts was reduced from 4,757 alerts (using the QVFH method) to 1,871 alerts (using the ML+OPT method), indicating a 60% decrease in the backlog size. Furthermore, the average time taken to investigate an alert each day of the week was lower for the ML+OPT method compared to the QVFH method. This resulted in a higher alert investigation rate using our approach. Figures 7(a)–7(b) illustrate the trend lines representing the backlog of alerts for the QVFH and ML+OPT methods over the course of the seven days. It can be observed that our methodology with the optimal matching mechanism helps to prevent the backlog from growing exponentially, leading to improved response times and enhanced network protection against cyber threats.

6 CONCLUSIONS AND FUTURE DIRECTIONS

In summary, we proposed a novel framework that integrates ML and optimization to enhance the efficiency of the alert-management process in CSOCs. We demonstrated the effectiveness of our approach using real-world CSOC alert-management data. Our methodological framework involves preprocessing the alert data and engineering features to develop an unsupervised learning model. The ML model then forms clusters of alerts based on feature similarities. Next, we match the skills required to investigate the alerts in these clusters with the respective skills of the analysts using a mathematical program. We develop an optimization model to maximize the matching pairs of analysts and alert clusters, thereby facilitating efficient alert investigations. The experiment results indicate that our approach, which groups similar alerts and assigns them to analysts based on their skill sets, significantly increases the alert investigation rate of the SRT in a CSOC. Additionally, through feature engineering and cluster formation, we can identify missing analyst skills. This information can be utilized to enhance the skill set of the analyst workforce by providing targeted training or improving the hiring and staffing processes.

Future research efforts could focus on utilizing false positive decisions from analyst investigations (feedback component) to refine the training data for customizing the IDS for the CSOC, thereby reducing false alerts. Exploring optimal training strategies to enhance analyst skill sets in alignment with evolving CSOC needs represents another promising avenue. Additionally, a valuable study would involve developing a methodology that automates the remediation of alerts in the investigation queue using historical data. Such an approach has the potential to significantly improve the alert-management process within a CSOC, benefiting both researchers and practitioners in the cybersecurity community.

Acknowledgments

The authors thank the CSOC team for providing their alert data to compare the various methods used in this research study.

REFERENCES

[1] CIS. 2017. Center for Internet Security (CIS). Retrieved from https://www.cisecurity.org/ms-isac. [Accessed: 2023-02-29].

[2] Python. 2022. Regular Expression Python Package. Retrieved from https://pypi.org/project/regex/. [Accessed: 2023-01-15].

[3] Douglas S. Altner, Anthony C. Rojas, and Leslie D. Servi. 2018. A two-stage stochastic program for multi-shift, multi-analyst, workforce optimization with multiple on-call options. *J. Schedul.* 21 (2018), 517–531.

[4] Muhamad Erza Aminanto, Lei Zhu, Tao Ban, Ryoichi Isawa, Takeshi Takahashi, and Daisuke Inoue. 2019. Automated threat-alert screening for battling alert fatigue with temporal isolation forest. In *Proceedings of the 17th International Conference on Privacy, Security and Trust (PST'19)*. 1–3. https://doi.org/10.1109/PST47121.2019.8949029

[5] Giovanni Apruzzese, Michele Colajanni, Luca Ferretti, Alessandro Guido, and Mirco Marchetti. 2018. On the effectiveness of machine and deep learning for cyber security. In *Proceedings of the 10th International Conference on Cyber Conflict (CyCon'18)*. 371–390. https://doi.org/10.23919/CYCON.2018.8405026

[6] Giovanni Apruzzese, Pavel Laskov, Edgardo Montes de Oca, Wissam Mallouli, Luis Brdalo Rapa, Athanasios Vasileios Grammatopoulos, and Fabio Di Franco. 2023. The role of machine learning in cybersecurity. *Digital Threats* 4, 1, Article 8 (Mar. 2023), 38 pages. https://doi.org/10.1145/3545574

[7] Giovanni Apruzzese, Pavel Laskov, and Aliya Tastemirova. 2022. SoK: The impact of unlabelled data in cyberthreat detection. In *Proceedings of the IEEE 7th European Symposium on Security and Privacy (EuroS&P '22)*. 20–42. https://doi.org/10.1109/EuroSP53844.2022.00010

[8] Giovanni Apruzzese, Fabio Pierazzi, Michele Colajanni, and Mirco Marchetti. 2020. Detection and threat prioritization of pivoting attacks in large networks. *IEEE Trans. Emerg. Top. Comput.* 8, 2 (2020), 404–415. https://doi.org/10.1109/TETC.2017.2764885

[9] Haipeng Chen, Andrew Duncklee, Sushil Jajodia, Rui Liu, Sean Mcnamara, and V. S. Subrahmanian. 2022. PCAM: A data-driven probabilistic cyber-alert management framework. *ACM Trans. Internet Technol.* 22, 3, Article 67 (Jan. 2022), 24 pages. https://doi.org/10.1145/3511101

[10] IBM ILOG Cplex. 2009. V12. 1: User's manual for CPLEX. *Int. Bus. Mach. Corp.* 46, 53 (2009), 157.

[11] Anita D'Amico and Kirsten Whitley. 2008. The real work of computer network defense analysts: The analysis roles and processes that transform network data into security situation awareness. In *Proceedings of the Workshop on Visualization for Computer Security (VizSEC'07)*. Springer, 19–37.

[12] Robert F. Erbacher and Steve E. Hutchinson. 2012. Extending case-based reasoning to network alert reporting. In *Proceedings of the International Conference on Cyber Security*. IEEE, 187–194.

[13] Charles Feng, Shuning Wu, and Ningwei Liu. 2017. A user-centric machine learning framework for cyber security operations center. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI'17)*. 173–175. https://doi.org/10.1109/ISI.2017.8004902

[14] Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2017. Optimal scheduling of cybersecurity analysts for minimizing risk. *ACM Trans. Intell. Syst. Technol.* 8, 4 (2017), 1–32.

[15] Rajesh Ganesan, Sushil Jajodia, Ankit Shah, and Hasan Cam. 2016. Dynamic scheduling of cybersecurity analysts for minimizing risk using reinforcement learning. *ACM Trans. Intell. Syst. Technol.* 8, 1 (2016), 1–21.

[16] Rajesh Ganesan and Ankit Shah. 2018. A strategy for effective alert analysis at a cyber security operations center. In *From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday*. Springer, 206–226.

[17] Rajesh Ganesan, Ankit Shah, Sushil Jajodia, and Hasan Cam. 2017. A novel metric for measuring operational effectiveness of a cybersecurity operations center. In *Network Security Metrics*. Springer, 177–207.

[18] Rajesh Ganesan, Ankit Shah, Sushil Jajodia, and Hasan Cam. 2019. Optimizing alert data management processes at a cyber security operations center. In *Adversarial and Uncertain Reasoning for Adaptive Cyber Defense: Control-and Game-Theoretic Approaches to Cyber Security*. Springer, 206–231.

[19] Nitika Gupta, Issa Traore, and Paulo Magella Faria de Quinan. 2019. Automated event prioritization for security operation center using deep learning. In *Proceedings of the IEEE International Conference on Big Data (BigData'19)*. 5864–5872. https://doi.org/10.1109/BigData47090.2019.9006073

[20] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. Retrieved from https://www.gurobi.com

[21] Eider Iturbe, Erkuden Rios, Angel Rego, and Nerea Toledo. 2023. Artificial intelligence for next generation cybersecurity: The AI4CYBER framework. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*. 1–8.

[22] Ravindran Kannan and Clyde L. Monma. 1978. On the computational complexity of integer programming problems. In *Optimization and Operations Research*. Springer, 161–172.

[23] Georgia Killcrece, Klaus-Peter Kossakowski, Robin Ruefle, and Mark Zajicek. 2003. *State of the Practice of Computer Security Incident Response Teams (CSIRTs)*. Technical Report CMU/SEI-2003-TR-001. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. Retrieved from http://resources.sei.cmu.edu/library/asset-view.cfmAssetID=6571

[24] Aron Laszka, Yevgeniy Vorobeychik, Daniel Fabbri, Chao Yan, and Bradley Malin. 2017. A game-theoretic approach for alert prioritization. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. AI Access Foundation, 195–202.

[25] Steven McElwee, Jeffrey Heaton, James Fraley, and James Cannady. 2017. Deep learning for prioritizing and responding to intrusion detection alerts. In *Proceedings of the IEEE Military Communications Conference (MILCOM'17)*. 1–5. https://doi.org/10.1109/MILCOM.2017.8170757

[26] Samuel Ndichu, Tao Ban, Takeshi Takahashi, and Daisuke Inoue. 2023. AI-assisted security alert data analysis with imbalanced learning methods. *Appl. Sci.* 13, 3 (2023), 1977.

[27] Christos H. Papadimitriou. 1981. On the complexity of integer programming. *J. ACM* 28, 4 (1981), 765–768.

[28] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst.* 42, 3 (2017), 1–21.

[29] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2018. Adaptive reallocation of cybersecurity analysts to sensors for balancing risk between sensors. *Serv. Orient. Comput. Appl.* 12 (2018), 123–135.

[30] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2018. Dynamic optimization of the level of operational effectiveness of a CSOC under adverse conditions. *ACM Trans. Intell. Syst. Technol.* 9, 5 (2018), 1–20.

[31] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2018. A methodology to measure and monitor level of operational effectiveness of a CSOC. *Int. J. Info. Secur.* 17 (2018), 121–134.

[32] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2019. Optimal assignment of sensors to analysts in a cybersecurity operations center. *IEEE Syst. J.* 13, 1 (2019), 1060–1071. https://doi.org/10.1109/JSYST.2018.2809506

[33] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2019. A two-step approach to optimal selection of alerts for investigation in a CSOC. *IEEE Trans. Info. Forens. Secur.* 14, 7 (2019), 1857–1870. https://doi.org/10.1109/TIFS.2018.2886465

[34] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2022. Maintaining the level of operational effectiveness of a CSOC under adverse conditions. *Int. J. Info. Secur.* 21 (2022), 637–651.

[35] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, Hasan Cam, and Steve Hutchinson. 2023. A novel team formation framework based on performance in a cybersecurity operations center. *IEEE Trans. Serv. Comput.* 16, 4 (2023), 2359–2371. https://doi.org/10.1109/TSC.2023.3253307

[36] Julie Steinke, Balca Bolunmez, Laura Fletcher, Vicki Wang, Alan J. Tomassetti, Kristin M. Repchick, Stephen J. Zaccaro, Reeshad S. Dalal, and Lois E. Tetrick. 2015. Improving cybersecurity incident response team effectiveness using teams-based research. *IEEE Secur. Privacy* 13, 4 (2015), 20–29.

[37] Yuan-Hsiang Su, Michael Cheng Yi Cho, and Hsiu-Chuan Huang. 2019. False alert buster: An adaptive approach for NIDS false alert filtering. In *Proceedings of the 2nd International Conference on Computing and Big Data (ICCBD'19)*. Association for Computing Machinery, New York, NY, 58–62. https://doi.org/10.1145/3366650.3366657

[38] Sathya Chandran Sundaramurthy, Alexandru G. Bardas, Jacob Case, Xinming Ou, Michael Wesch, John McHugh, S Raj Rajagopalan, and Lorrie Faith Cranor. 2015. A human capital model for mitigating security analyst burnout. In *Proceedings of the 11th Symposium On Usable Privacy and Security (SOUPS '15)*. 347–359.

[39] Sathya Chandran Sundaramurthy, John McHugh, Xinming Ou, Michael Wesch, Alexandru G. Bardas, and S. Raj Rajagopalan. 2016. Turning contradictions into innovations or: How we learned to stop whining and improve security operations. In *Proceedings of the 12th Symposium on Usable Privacy and Security (SOUPS'16)*. USENIX Association, 237–251.

[40] Liang Tong, Aron Laszka, Chao Yan, Ning Zhang, and Yevgeniy Vorobeychik. 2020. Finding needles in a moving haystack: Prioritizing alerts with adversarial reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 946–953.

[41] Kristijan Vidović, Ivan Tomičić, Karlo Slovenec, Miljenko Mikuc, and Ivona Brajdić. 2021. Ranking network devices for alarm prioritisation: Intrusion detection case study. In *Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM'21)*. 1–5. https://doi.org/10.23919/SoftCOM52868.2021.9559086

[42] Manfred Vielberth, Fabian Böhm, Ines Fichtinger, and Günther Pernul. 2020. Security operations center: A systematic study and open challenges. *IEEE Access* 8 (2020), 227756–227779.

[43] Ting-Fang Yen, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. 2013. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*. 199–208.

Received 31 July 2023; revised 7 December 2023; accepted 26 January 2024