applied sciences

MDPI

Article

Breaking Alert Fatigue: AI-Assisted SIEM Framework for Effective Incident Response

Tao Ban *, Takeshi Takahashi, Samuel Ndichu and Daisuke Inoue

Cybersecurity Research Institute, National Institute of Information and Communications Technology, Tokyo 184-8795, Japan; dai@nict.go.jp (D.I.)
* Correspondence: bantao@nict.go.jp; Tel.: +81-42-327-7529

Abstract: Contemporary security information and event management (SIEM) solutions struggle to identify critical security incidents effectively due to the overwhelming number of false alerts generated by disparate security products, which results in significant alert fatigue and hinders effective incident response. To overcome this challenge, we propose a next-generation SIEM framework that integrates security orchestration automation and response capabilities and utilizes a divide-and-conquer strategy to mitigate the impact of low-quality IDS alerts. The proposed framework leverages advanced machine learning and data visualization tools—including a cost-sensitive learning method and an event segmenting algorithm—to filter and correlate alerts plus an augmented visualization tool to expedite the triage process. The proposed framework was evaluated experimentally on a dataset collected from a real-world enterprise network, and we report highly convincing results. The alert screening scheme demonstrates significant potential for real-world security operations. We believe that our findings will contributing to the development of a next-generation SIEM system that effectively addresses alert fatigue and lays the foundation for future research in this field.

Keywords: network security; intrusion detection; incident response; SIEM; alert fatigue; AI and machine learning

Citation: Ban, T.; Takahashi, T.; Ndichu, S.; Inoue, D. Breaking Alert Fatigue: AI-Assisted SIEM Framework for Effective Incident Response. *Appl. Sci.* **2023**, *13*, 6610. https://doi.org/10.3390/app13116610

Academic Editor: Wenbo He

Received: 1 April 2023
Revised: 11 May 2023
Accepted: 25 May 2023
Published: 29 May 2023

1. Introduction

Increased usage has exposed the Internet infrastructure and other network-connected systems to cyberthreats with increasing sophistication. In the past few years, the global cybersecurity landscape has seen a steady number of reported incidents ranging from conventional attacks, e.g., ransomware, phishing, and denial of service (DoS) attacks, to novel, hybrid, and emerging threats that exhibit an upward trend and strong impacts [1,2].

Thus, intrusion detection is a critical component to realize the operational and strategic goals of enterprise security operation centers (SOCs) to defend against these threats. Intrusion detection systems (IDS) can provide this capability by monitoring a network or systems and raising alerts when abnormal activities or policy violations are detected [3]. IDSs can be categorized into host-based IDSs (HIDSs) and network-based IDSs (NIDSs) [4]. HIDSs are installed on individual hosts or systems, and they are used to detect intrusions based on internal evidence, e.g., the modification or deletion of critical system files. In contrast, NIDSs are distributed at various network access points, and they detect intrusions by inspecting the communications between network-connected devices. Depending on the difference in their detection mechanisms, NIDSs lean toward proactive detection and prevention of intruding activities before a breach occurs, and HIDSs act as a second layer of defense, i.e., they operate at the endpoint level when a system is breached. In practice, NIDS and HIDS are frequently deployed to work in a collaborative and complementary mode to provide better security.

Due to their comprehensive and proactive nature, enterprise SOCs often rely heavily on NIDSs to acquire a holistic perspective of the security situation and to prevent intrusions

at an early stage [5,6]. Examples of widely adopted NIDSs include open-source projects, e.g., Snort [7], Zeek [8], and Suricata [9], and their commercial counterparts provided by security vendors, e.g., LastLine, which is now part of VMware [10], FireEye [11], and Trend-Micro [12]. However, NIDS effectiveness is frequently hampered by the low quality of the issued alerts. As reported by Vaarandi et al. [13], when deployed in large financial institutions, Snort can produce more than 4,000,000 alerts per day on average. It is obviously unfeasible to process this many alerts manually in a common SOC, and the vast majority of low-quality alerts result in a considerable waste of valuable resources.

Moreover, to maximize the detection rate for crucial incidents, many companies are now employing multiple IDSs in tandem with other security appliances. Nevertheless, this growing number of integrated security appliances generally worsens the security situation due to the so-called *alert fatigue* issue, where security analysts become desensitized to the security alarms (alerts) after constant exposure to a large volume of promiscuous alerts with uncontrolled false positives. This desensitization can result in delayed incident responses and overlooked critical incidents, ultimately leading to increased network security risks and reduced service quality. A recent survey conducted by the Cloud Security Alliance [14] found that half of the enterprises surveyed utilize six or more security tools that generate alerts. Additionally, nearly one-third of IT security professionals who participated in the survey admitted to ignoring alerts due to the frequent occurrence of false positives.

Alert fatigue is a serious barrier to implementing a consolidated security solution by integrating multiple security appliances. Currently, the common practice to mitigate the alert fatigue problem is to leverage a security information and event management (SIEM) system [15] to combine outputs from multiple sources for a consolidated security solution. Ideally, an SIEM should provide integrated functionalities, e.g., log data collection, efficient data retention, log indexing and searching, automated reporting, threat detection alerting, and event correlation. Unfortunately, the effectiveness of contemporary SIEMs is frequently hindered by various challenges that make them as much of a burden as a benefit. In addition, many advanced qualities, including enhanced visualization, open-source intelligence, and artificial intelligence (AI)/machine learning (ML) capabilities, are attractive features for next-generation SIEMs [15].

Thus, in this paper, by leveraging the latest AI, ML, and visualization technologies, we propose a framework for a next-generation SIEM system that integrates genuine security orchestration automation and response (SOAR) capabilities to realize effective incident detection and response. We address the key issue of alert fatigue caused by high numbers of low-quality IDS alerts using a divide-and-conquer strategy. First, we decompose alert fatigue into two causal sub-problems, i.e., filtering and correlation problems [16]. Second, we employ advanced AI and data visualization methods to solve each sub-problem. Specifically, we model the filtering problem as a supervised learning task to screen out false alerts in the IDS output using an instance-weighted support vector machine (IWSVM), which is a cost-sensitive learning method. The correlation problem is addressed by grouping correlated alerts into events using a procedure that exploits their homogeneity and temporal coincidence. We then introduce a novel visualization tool to confirm the results of this grouping procedure.

The proposed solution provides an alternative perspective on the alert fatigue problem, which is commonly recognized as a key issue in enterprise SOC operations. By synergistically exploiting advanced AI/ML and data visualization tools, we provide an integrated solution and prototype for a next-generation SIEM. Our primary contributions are summarized as follows.

- We propose a comprehensive framework to design and implement a next-generation SIEM system, and we demonstrate its effectiveness by prototyping the key components of the framework.
- We formalize a divide-and-conquer strategy to mitigate the alert fatigue problem and introduce practical solutions for each of its component problems, i.e., the filtering and correlation problems. We also present practical solutions to these problems, including the use of the cost-sensitive IWSVM learning method to address the issue

of imbalanced learning in the filtering process and an algorithm that leverages the proximity, spatial, and temporal homogeneity of alerts to effectively summarize them into events for more efficient incident handling.
- We present an augmented visualization tool, which we refer to as augmented tile graph (ATG), and demonstrate its ability to improve the performance of data analysis against security alerts.

Overall, this paper offers a valuable contribution toward the development of a next-generation SIEM system that effectively addresses the challenge of alert fatigue and paves the way for future research in this field.

The remainder of this paper is organized as follows. Section 2 provides the necessary background for efficient incident handling in an SOC and reviews related work on alert fatigue. Section 3 introduces the proposed system framework, and Section 4 discusses the design and implementation of the component systems to address alert fatigue. Section 5 presents experimental results that demonstrate the effectiveness of the proposed approach on real enterprise network data. Section 6 discusses the limitations of the proposed framework and identifies potential directions for further improvement. Finally, the paper is concluded in Section 7.

2. Background and Related Work

In this section, we provide an overview of current practices to detect and mitigate network intrusions. We also discuss the challenges of alert fatigue due to high numbers of false alerts, and we review previous studies on reducing false alerts to improve the effectiveness of incident-handling operations.

2.1. Incident Response with NIDS, SIEM, and SOAR

Computer security incident management involves monitoring and detecting security events on a computer or network and executing appropriate responses. Events detected by NIDSs and HIDSs are observable changes to a system's expected behavior. There are three types of events: normal, escalation, and emergency, each requiring different responses. A normal event does not affect critical components or require change controls prior to the implementation of a resolution. An escalated event affects critical production systems and requires a change to the control process, i.e., participation of senior personnel and notification to the stakeholders associated with the event, before implementing the resolution. An emergency event can impact the health or safety of human beings, breach the primary controls of critical systems, and/or materially affect component performance. NIDS, SIEM, and SOAR are security technologies used at different stages of incident management.

2.1.1. IDSs

It is critical to detect intrusions to handle incidents efficiently. Detection capabilities can be provided by HIDSs installed on critical systems or NIDSs distributed at various network access points, which issue alerts to security incident management when unauthorized activities are detected on monitored systems. Numerous commercial IDS products are available, and research into new IDS implementations is ongoing. IDS products can be signature-based or anomaly-based. Signature-based detection relies on a database of known malicious patterns, and anomaly-based detection uses a model of normal traffic to detect deviations. An NIDS with response capabilities, e.g., blocking traffic when an intrusion is detected, is typically referred to as an intrusion prevention system [17]. As surveyed in the literature [18], recent research on IDSs has focused on applying the most recent AI and ML methods to detect intrusions [19,20]. Noteworthy work includes extending IDS schemes to protect network environments, e.g., the Internet of Things [21,22], cloud environments [23,24], software-defined networks [25], and automation and control systems [26].

2.1.2. SIEM

SIEM solutions are crucial to enhance the security posture of an enterprise network [27]. The goal of these solutions is to collect, integrate, and analyze security-related data from multiple sources, including logs from firewalls, IDSs, and endpoint security tools. As a result, SIEM solutions provide a comprehensive view of the activities occurring in the IT infrastructure, and they can detect potential security incidents before they cause significant damage. In addition to threat intelligence, many SIEM solutions employ user and entity behavior analytics [28] to monitor network activities and detect and mitigate attacks. This helps simplify security operations and reduces the administrative burden of incident response, allowing organizations to enhance their overall security posture and respond to security incidents effectively and efficiently. Recently, increasing interest has been placed on deploying SIEM solutions for security threat detection and response [29]. However, various challenges, e.g., false positive alerts and alert fatigue, remain, and efforts are being made to address these issues using ML and incident response workflows.

2.1.3. SOAR

As a collection of security software solutions and tools for browsing and collecting data from various sources, SOAR systems automate the incident response process, streamline the handling of security incidents, and improve the efficiency and effectiveness of incident response teams [30]. According to Gartner [31], SOAR systems integrate with multiple security tools, e.g., SIEMs, firewalls, and IDSs, to automate various tasks, including triage, investigation, and remediation. These tasks are performed according to defined rules and playbooks, which can be customized to respond to specific types of incidents. Thus, SOAR can help organizations improve their overall security posture, reduce response times, and minimize the risk of human error during incident handling [32]. As summarized by Gupta et al. [33], despite the potential benefits of SOAR, research in this area is still in an early stage, and further work is required to advance capabilities in orchestration, incident response automation, and reporting.

2.2. Previous Studies on Alert Fatigue

The evolving cybersecurity landscape presents challenges for current IDSs due to the sophisticated techniques used by threat actors. Missed critical events and high false alarm rates can result in severe consequences, e.g., data breaches and reputational damage [18]. To address these challenges, enterprises deploy multiple IDSs and complex SIEM solutions; however, this can result in alert fatigue and reduced productivity among security personnel.

2.2.1. Filtering- and Correlation-Based Approaches

Incident handling performance is strongly affected by the alert fatigue issue caused by low-quality IDS alerts. A common solution to false and redundant alerts in IDS outputs is to make them less sensitive. However, this approach conflicts with the need to not overlook incidents and to possess detailed information in post-detection analyses. It also results in experts wasting time tuning IDSs during setup and daily operations. A more effective approach is to allow the IDSs to be overly sensitive and address the filtering and correlation problems in a preprocessing step. Here, the filtering problem refers to the overwhelming number of false alerts in the IDS output that are caused by imperfections in the detection mechanism. The correlation problem refers to the extra number of reported alerts, which are caused by the presence of replicated or correlated alerts in the IDS output due to imperfections in the reporting mechanism. Numerous related works have investigated this approach and have shown it to achieve good results. The proposed solutions have been adopted by commercial products [5,34,35].

One of the approaches is the correlation-based method introduced by Valeur et al. [36], which can reduce the number of alerts by up to 99.2% for honeypot datasets. However, its effectiveness drops to 53.0% on the MIT/LL 2000 dataset, and its dependence on correlation components makes it impractical in scenarios where access rights to the host

may not be available. To tackle the issue of alert fatigue, Hassan et al. [37] proposed the NoDoze method, which incorporates an automated provenance triage process. This method adjusts the suspiciousness level of each event in the provenance graph based on its neighboring events, performs behavioral execution partitioning, and generates a dependency graph of accurate alerts to avoid dependency explosion resulting from prior data provenance. Another effective method is the Isolation Forest (IF) technique, as employed by Sun et al. [38], which detects deviations from normal employee behavior by considering the temporal aspect. This technique involves collecting data for a specific period and conducting anomaly detection on the entire dataset.

2.2.2. Alert Prioritization

By taking too many irrelevant, low-quality alerts as inputs, preprocessing techniques based on filtering and correlation can provide imprecise results due to the noisy input. To solve this problem, alert prioritization attempts to address the difficulty in managing a large number of alerts and improve the accuracy of results by reducing the number of unrelated alerts.

Chakir et al. [39] proposed an alert prioritization model that employs risk assessment to assess the impact of IDS-generated alerts on the security status of an information system. This model incorporates various factors, e.g., priority, reliability, and asset value, to calculate the risk associated with each alert and prioritize the most critical alerts based on their risk levels. The model was evaluated on the KDD Cup 1999 dataset to assess its effectiveness. Aminanto et al. [40] utilized an isolation forest technique to filter out low-level threat alerts. Here, they considered temporal information in alerts, allowing for real-time prediction of incoming threats. This study demonstrated the potential of unsupervised learning in reducing alert fatigue. The method was then improved [41] by adding a one-class stacked autoencoder that reduced false positives by selecting outliers with high reconstruction error rates as candidates.

2.3. Feature Engineering

The alerts generated by various NDSs may have various formats, thereby making it challenging to analyze the data effectively. Effective analyses of such heterogeneous data typically require a preprocessing step to convert the data into a standardized format. However, emerging methods attempt to avoid the need for data transformation and work directly with heterogeneous formats.

2.3.1. Feature-Based Approach

Several studies have aimed to standardize the format of security alert logs to facilitate analysis and improve threat detection. Madani et al. [42] highlighted the challenges associated with handling logs in various formats generated by different security appliances. These formats include the log event extended format (LEEF) [43], common event format (CEF) [44], intrusion detection message exchange format (IDMEF) [45], and the common event expression format (CEE) [46]. The lack of a standard format makes it challenging to analyze and compare threat alerts across different systems.

To address this issue, Azodi et al. [47] proposed a model that can read and normalize different log formats using named-group regular expressions and a knowledge base. Similarly, Sapegin et al. [48] introduced a new common log format that consolidates all necessary information from different formats into a single format. These efforts towards standardization can improve the efficiency and effectiveness of security operations and enable more accurate threat detection.

2.3.2. Featureless Approach

Despite its high performance, the feature engineering process has been criticized for reducing adaptability and being tedious and time consuming [49]. It has been referred to

as a pain point in building trained systems, and it requires a high level of computer science expertise [50].

A recent study [16] proposed a method to correlate and filter data without relying on domain knowledge or feature engineering. Here, the alerts from an IDS are processed as text strings and transformed into numerical representations. These representations are then grouped into representative profiles using clustering methods. The authors presented two implementations of their approach using a long short-term memory recurrent neural network and latent semantic analysis. Although these implementations did not perform as well as existing methods that rely on feature engineering when evaluated using common detection metrics, they demonstrated promise when evaluated using practical metrics for filtering and correlating.

3. Proposed SIEM Framework

In this section, we present the proposed SIEM framework, which is designed to address alert fatigue in security operations. The proposed framework combines AI and data visualization techniques to streamline the incident handling process. As shown in Figure 1, the framework comprises four key modules.

- The alert generation module collects alerts from multiple IDS sources.
- The feature processing module standardizes diverse log formats and encodes relevant information as numerical vectors for uniform representation.
- The ML module employs supervised learning algorithms on labeled alerts to develop a prediction model to detect critical alerts.
- The investigation module assesses the performance of the prediction models and enables rapid incident investigation through data visualization and alert correlation, which summarize alerts into high-priority events.

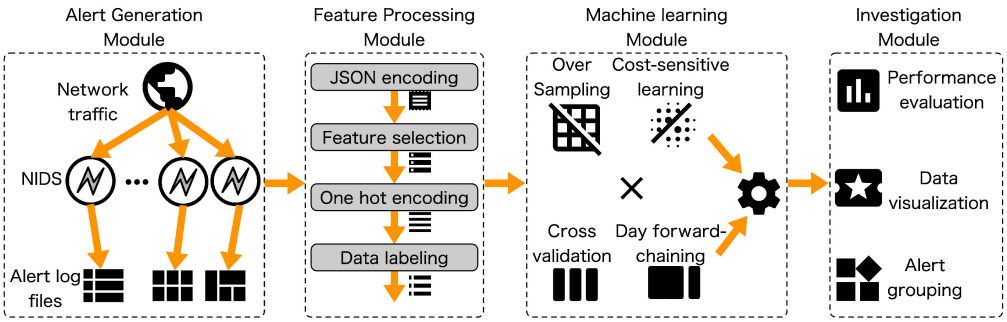


Figure 1. Proposed SIEM system framework The NIDs are represented by circled thunder icons. The orange arrows illustrate the flow of data among the data processing components implemented within the framework. The cross symbol indicates the ability to select and integrate the components based on the specific situation. The other icons in the figure are used purely for illustrative purposes and do not possess specific connotations.

3.1. Alert Generation Module

Although integrating multiple security appliances into an SIEM can aggravate the issue of alert fatigue, our extensive experimental analysis of a substantial number of alerts generated by six IDSs revealed that these systems work together in a complementary manner, enabling the identification and detection of a more extensive range of anomalies within the network. Alerts generated by different IDSs can offer valuable insights and information, even if they are triggered by the same attack, as each IDS has its own unique detection mechanisms. Thus, we believe that integrating multiple security appliances in SIEM can enhance the security of the monitored network as long as the alert fatigue problem can be solved.

The framework we propose advocates for the deployment of multiple security appliances within enterprise networks to ensure comprehensive protection against cyberthreats.

However, previous research [51] has demonstrated that deploying these appliances in a serial manner can result in performance degradation at high packet rates. Thus, we recommend deploying these appliances in parallel, provided it is cost-effective, to maintain multiple security appliances.

3.2. Feature Processing Module

The feature processing module attempts to address integrating security alert logs of varying log formats from different devices. The CEF [44] was introduced by ArcSight and is now part of HPE. The CEF has been widely adopted; however, many IDSs continue to use unique log formats. The proposed data unification component resolves this issue by parsing alert messages and converting them into standard JavaScript object notation (JSON) objects [52]. By consolidating data from different vendors into a single, centralized view, a comprehensive and precise depiction of the network's security status can be obtained. In Appendix A.1, we present a sample alert log in the CEF format and demonstrate how it is parsed and converted into a structured JSON object.

The JSON objects generated by the data unification component contain crucial event information, e.g., event descriptions, unique identifiers, the IP addresses involved in the communication, event impacts, the URLs of risky Internet resources, and the hash values of downloaded files. These attributes are categorized into numerical, categorical, and signature types, with the most relevant attributes selected based on their descriptive and generalizable nature to represent the corresponding event. Each alert message in the log file is transformed into a numerical vector using one-hot encoding, which converts categorical features into binary attributes.

To ensure accurate labeling of critical alerts, the data are initially examined manually by security personnel in the SOC. However, to mitigate the risk of misjudgment and alert fatigue, AI experts review the labeled data using information visualization tools to validate the accuracy of the label information. Additional details about the data used in the evaluation are provided in Section 5.

3.3. Machine Learning Module

The ML module leverages AI and ML algorithms to implement the filtering function, which reduces the number of unrelated alerts in subsequent processing. To achieve this, we must formulate the filtering problem as a common learning problem, and we must select a suitable ML method to solve the problem effectively.

3.3.1. Problem Formulation

The goal of the filtering process is to assign each vector with an indicator of its importance (or, more directly, assess its significance). Given the superior accuracy of supervised learning algorithms compared to other learning models, we propose to formulate the filtering problem as a supervised learning task. The supervised learning model takes the vector representation of the alert as input and predicts a class label indicating whether the sample is a critical alert.

3.3.2. Class Imbalance

With a sufficiently large training set, the filtering problem can be solved by any classification algorithm that can categorize incoming alerts as critical or noncritical. However, class imbalance poses a challenge. An imbalanced dataset refers to a situation where there is a significant disproportion in the number of instances for each class [53]. This problem is prevalent in security alert data, where the frequency of critical alerts is much lower than that of noncritical alerts [54,55]. As a result, conventional classifiers may be biased toward noncritical alerts, thereby hindering the detection of crucial alerts associated with critical incidents. Thus, a method is required to address class imbalance in security alert data to ensure rapid responses to incidents in SOCs.

Class imbalance is commonly addressed by resampling the training data through subsampling, oversampling, or hybrid methods. Subsampling can reduce performance due to information loss in the reduced sample size, and oversampling can be effective through data augmentation algorithms, e.g., the synthetic minority oversampling technique (SMOTE) [56], adaptive synthetic sampling [57], and SVMSMOTE [58]. However, conventional oversampling methods may be ineffective for datasets with a large number of categorical attributes. A previous study [59] demonstrated that incorporating multiple oversampling methods can synthesize samples with more diversity and higher quality. Another approach to address class imbalance is to adopt cost-sensitive approaches, e.g., WSVMs [60,61]. A weighted support vector machine (WSVM) assigns distinct weights to positive and negative samples, which scales the margin in proportion to the class significance. This makes it more efficient in managing imbalanced datasets. In this paper, we propose an improved WSVM, which considers the different weights assigned to samples from different classes and reduces the sample size by summarizing the replicated counts into an instance weight. This results in improved training and prediction performance. We refer to this approach as the IWSVM, which is described in detail in Section 4.

To determine the best approach to mitigate class imbalance in security alert data with a large number of categorical attributes, we compare the proposed method with popular oversampling techniques and the hybrid approach proposed by Ndichu et al. [59]. The experimental results (Section 5) demonstrate the effectiveness of the proposed IWSVM over conventional oversampling methods and the hybrid approach.

3.3.3. Baseline Classification Methods

As baseline methods for comparison, we evaluated six classification methods that can efficiently handle high-dimensional sparse data [62]: K nearest neighbors (KNN), naive Bayes (NB), linear discriminant analysis (LDA), decision tree (DT), AdaBoost, and SVM. These methods were used to create prediction models for the alert dataset (Section 5). The proposed IWSVM and the reference WSVM are described in Section 4.

KNN is a well-known machine learning algorithm used for classification and regression tasks. It finds the K closest training samples in the feature space and assigns the class of the test sample based on majority vote of these neighbors. It is simple and effective but sensitive to irrelevant features and computationally expensive for large datasets or high-dimensional feature spaces.

NB is a probabilistic machine learning algorithm used for classification tasks. It works by computing the probability of a test sample belonging to each class based on the probabilities of the features given to each class and then assigning the class with the highest probability. It assumes independence between features, making it efficient for high-dimensional datasets. It is simple and fast, but the independence assumption may not hold in some cases.

LDA is a supervised machine learning algorithm used for classification tasks. It works by projecting the data onto a lower-dimensional space while maximizing the separation between the classes. It assumes normality and equal covariance matrices but may not work well if these assumptions do not hold.

DT is a supervised machine learning algorithm used for classification and regression tasks. It works by recursively splitting the data into subsets based on the values of features, with the goal of maximizing the purity of each subset. It is simple and easy to interpret, but can be prone to overfitting and may not generalize well. Techniques such as pruning and ensemble methods can help address these issues.

AdaBoost is a supervised machine learning algorithm used for classification tasks. It works by combining several weak classifiers into a strong one by iteratively training them on weighted versions of the data and giving higher weights to misclassified samples. It is powerful and less prone to overfitting but can be sensitive to noisy data and outliers.

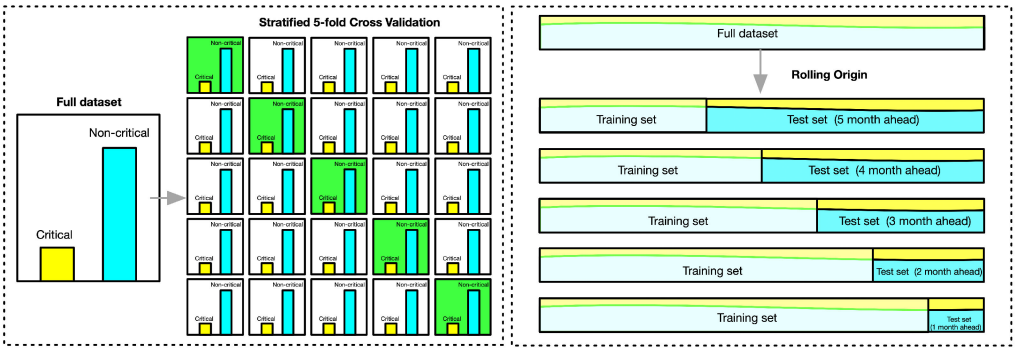
SVM is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane that separates the data into classes with the

largest margin possible. The hyperplane is chosen to maximize the distance between the closest samples of each class, which are called support vectors. SVM is powerful and less prone to overfitting but can be computationally expensive and sensitive to the choice of kernel function and regularization parameters. We chose to use the Radio Basis Function (RBF) kernel-based SVM in the experiment because it is good at dealing with complex, non-linear datasets and has a feature that prevents overfitting.

3.3.4. Evaluation Schemes

Two performance evaluation schemes were considered to assess the performance of our models, i.e., stratified cross validation (SCV) and rolling origin validation (ROV). The SCV [63] method (Figure 2a) evaluates ML models by partitioning the data into folds, where each fold is used as a validation set and the remaining data are used as a training set. The main objective of SCV is to ensure that the distribution of the target variable is represented equally across all folds, which is achieved by ensuring that each fold contains approximately the same proportion of each target class. The validation outcomes are then averaged across all folds to provide an assessment of the model's prediction performance.

ROV is a time-series validation technique frequently used when working with time-series data [64,65]. As shown in Figure 2b, the data are divided into training and validation sets, where the validation set begins at a certain point in time and rolls forward through time, which means that the most recent data are used as the validation set and older data are used as the training set. The advantage of ROV is that it provides a more realistic simulation of how a model would be used in practice because it reflects the scenario wherein the model would be making predictions using the most recent data available.



(**a**) (**b**)

Figure 2. Performance evaluation scheme. (**a**) Cross validation; (**b**) Rolling origin. The subset with a green background indicate it is the testing set for the CV-round. In (**a**), the subset distinguished by a green background signifies its designation as the testing set, while those identified with a white background indicate their inclusion in the training set for each cross-validation round.

3.4. Investigation Module

Security appliances often generate multiple alerts over a long period of time to detect attacks. This can result in an excessive number of alerts, for instance, in the case of a bot-infected host that repeatedly probes multiple targets on a network. While most IDSs log related information about a specific rule violation triggered by certain network communication, not all IDSs effectively group alerts related to the same attack campaign. As a consequence, the number of false alerts may increase unnecessarily. The improper configuration of detection rules and the generation of recurring alerts for persistent unsolicited communications can also significantly increase the number of false positives.

The investigation module incorporates data visualization techniques to simplify the complex process of investigating alerts. By working in tandem with the SOAR tools integrated into the SIEM solution, the security analysis process is streamlined, which empowers analysts to respond to threats with greater speed and precision. The integration of data visualization allows security analysts to identify trends and anomalies in the data

quickly through graphical formats, e.g., charts and graphs. It also provides a user-friendly interface to interact with the SIEM and helps engage stakeholders in the communication of security risks and priorities.

A previously proposed visualization tool [61] has undergone enhancements to visualize correlated alerts, and the updated version (Section 4) offers a clear graphical interpretation of the analyzed alert, thereby helping security analysts discern the links and correlations between various security incidents.

4. Methodology

In this section, we describe the implementation of our approach to addressing the alert fatigue issue with the proposed SIEM framework. Our framework consists of two key components: a filtering module and a correlation module. The filtering module uses IWSVM to identify and remove false alerts, thereby reducing the number of alerts that need to be further investigated. The correlation module groups alerts into events based on their homogeneity and temporal coincidence. To confirm the results of the alert correlation process, we introduce an ATG, which is an improved version of the tile graph [61].

4.1. Filtering Problem

The filtering problem involves identifying critical alerts from a given set of alerts. Recall the three event types defined in Section 2.1: normal, escalation, and emergency. Emergencies are relatively rare in environments controlled by an intrusion response team, and using such events as direct targets for learning may be overly aggressive. Thus, we propose to use escalations as positive samples for the target classification task. As a result, the classifier imitates the process of the intrusion response team by selecting escalations from the outputs of IDSs.

The following two reasons further explain the rationale behind this choice. First, selecting escalations from normal events typically only requires the information contained in the alert, whereas determining if an escalation is an emergency typically requires evidence beyond that provided in the alert. Second, the daily number of escalations reflects the throughput of the intrusion response team; thus, the predicted number of alerts and vigilance levels for ML methods targeting escalations should be reasonably appropriate.

The following definition formally describes the filtering problem as a binary classification task.

Definition 1 (Filtering Problem). *Let* $\{\boldsymbol{x}_i, y_i | i = 1, \ldots, N\}$ *denote a labeled alert set, where* $\boldsymbol{x}_i \in \mathbb{D}$ *is a d-dimensional vector obtained from the one-hot encoding representation of an alert, where* $y_i = 1$ *if the alert is an escalation; otherwise,* $y_i = 0$*. The filtering problem is solved by a decision function* $f(\boldsymbol{x}_i; \theta)$ *with a loss function* $\mathcal{L}(f(\cdot; \theta), \boldsymbol{x}_i, y_i)$*. Specifying the functional form of* f *and* $\mathcal{L}$ *completely specifies the classification algorithm.* $\theta = \arg\min_\theta \mathcal{L}(\theta)$ *defines the classifier. A sample* $\boldsymbol{x}$ *is predicted to be positive (a critical alert) if* $f(\boldsymbol{x}; \theta) > 0$*; otherwise, it is classified as negative (a noncritical alert).*

4.1.1. Filtering by Cost-Sensitive Learning

In this context, the filtering problem can be addressed by utilizing the classifier algorithms discussed in the previous section. However, the classification problem defined by the filtering problem is frequently associated with two main challenges. First, there is an extremely imbalanced distribution of positive and negative classes, which can make it difficult for the classifier to learn from the data effectively. Second, the impact of classification errors varies, with false positives and false negatives having different levels of harm to the system. Here, false positives refer to noncritical alerts that are misclassified as critical, and false negatives refer to critical alerts that are missed or classified as noncritical. This means that it is crucial to use an approach that considers the asymmetric costs of misclassification and the severe class imbalance present in the filtering problem. A promising technique that can address these challenges is cost-sensitive learning [66], which assigns different costs to false positives and false negatives based on their impact on the system, and it can optimize

the overall cost of misclassification. This is consistent with the requirement for setting a higher weight for the minority class, commonly critical alerts, to improve the prediction accuracy for such samples.

Cost-sensitive learning [66] can help encode prior knowledge about the class importance and sample instance importance differently in the classifier. In this study, we adopt the SVM classifier. Assigning a high weight to a class suggests that the learning algorithm should attempt to classify samples from that class correctly, possibly at the expense of misclassifying samples from the other class. This heuristic can be applied successfully to critical alerts because they require higher detection accuracy than noncritical alerts. This weight factor should be considered as class weights. However, information, e.g., source and destination IP addresses, timestamps, file names, and file hashes, that does not generalize well for classification is removed from the alerts during the processing; thus, the one-hot encoding yields many identical numerical vectors with very similar alerts corresponding to the same attack detected in different occasions. Training a classifier with many replicated samples is inefficient, and this can be addressed by assigning weights that reflect the frequency of the alerts to the samples. Note that this weight factor is deemed an instance weight. To differentiate it from the WSVM [61], we refer to this SVM classifier, which considers class weight and instance weight simultaneously, as the IWSVM.

To adopt the proposed IWSVM in the learning process, we first group the numerical vectors to obtain a set of unique vectors and their instance weights following the definition given below.

Definition 2 (**Instance Weight**). *In the context of alert databases, we define the instance weight of a vector* $\boldsymbol{u}_i$ *as follows. Let* $h : S \to \mathbb{N}$ *be a function that maps each vector to its frequency in the database and let* C *be a set of categories. Let* $w : C \to \mathbb{R}$ *be a function that assigns a weight to each category to indicate its importance. Define* $g : S \to C$ *as a function such that* $g(\boldsymbol{u}_i) = c_j$ *if and only if* $\boldsymbol{u}_i$ *belongs to category* c_j*. Then, the instance weight* $W(\boldsymbol{u}_i)$ *for* $\boldsymbol{u}_i$ *is given by* $W(\boldsymbol{u}_i) = h(\boldsymbol{u}_i)w(g(\boldsymbol{u}_i))$.

The instance weight of a sample vector is determined by the frequency of the corresponding alert in the database and the class weight, which is a predefined empirical value indicating the importance of the class. Using the unique sample vectors and their instance weights, an SVM solver optimizes the following weighted loss function [67]:

$$\hat{L}w(f) = \sum_{\boldsymbol{u}_i \in S} W(\boldsymbol{u}_i)\mathcal{L}(y_i, f(\boldsymbol{u}_i)), \tag{1}$$

where $\mathcal{L}(y_i, f(\boldsymbol{u}_i))$ is the loss function that penalizes the difference between the true label y_i and the predicted label $f(\boldsymbol{u}_i)$. In our experiments, we utilized the SVM and WSVM solvers implemented in the LIBSVM toolbox [60]. We adopted the 0-1 loss function, which is widely recognized as the most common loss function, in the experiments. In mathematical terms, it can be expressed as:

$$\mathcal{L}(y_i, f(\boldsymbol{u}_i)) = \begin{cases} 0 & \text{if } f(\boldsymbol{u}_i) = y_i, \\ 1 & \text{if } f(\boldsymbol{u}_i) \neq y_i. \end{cases} \tag{2}$$

4.2. Correlation Problem

In the previous section, we introduced a cost-sensitive learning scheme that filters out noncritical alerts and outputs only those that require further investigation. This method significantly reduces the number of alerts to be processed; however, it only operates on individual alerts and lacks the ability to reduce redundancy caused by duplicate and correlated critical alerts. To address this issue, we employ alert correlation—based on the compactness in terms of behavioral, spatial, and temporal factors—to group alerts into events and summarize them as an "event profile". These groups can be considered a

collection of alerts detected in the same attack campaign, thereby enabling a single-point investigation of the entire event.

When implementing correlation, behavioral similarity is a necessary but insufficient condition for grouping alerts into an attack event. To illustrate this, we consider two Emotet (Emotet, also known as Heodo,is a notorious malware strain and cybercrime operation that has caused havoc in the cybersecurity world since 2014 and is believed to have originated from Ukraine [68]) warnings: one detected a month ago on a finance department server and the other triggered this morning on an HR department server. Although the behavior of these alerts is similar and they should be categorized into the same group, the differences in communication targets and timing require they be treated as separate incidents. Thus, alert correlation requires careful consideration and tuning to ensure that only genuinely related alerts are grouped as part of the same event.

Definition 3 (Correlation Problem). *Let* $\boldsymbol{X} = \{\boldsymbol{x}_i | i = 1, \ldots, N\}$ *denote the alert set, and for each alert in* $\boldsymbol{X}$*, the three-tuple* $([S_i, D_i, P_i])$ *represents the source IP address, destination IP address, and service port. Let* t_i *denote the issuing timestamp associated with alert* $\boldsymbol{x}$ *in* $\boldsymbol{X}$ *and let* τ *represent the maximum duration (in seconds) for an event to exist before being terminated. A correlation problem involves grouping alerts in* $\boldsymbol{X}$ *based on their compactness in terms of behavioral, spatial, and temporal factors. Here, behavioral compactness is achieved by satisfying similarity conditions based on numerical representation of the alerts. Spatial compactness is achieved by satisfying the three-tuple condition defined in (3), where alerts in each event share the same source IP address, destination IP address, and service port. Temporal compactness is realized by satisfying the timeout condition defined in (4), which limits the duration of each event to a maximum value* τ*.*

$$\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{E}, S_i \equiv S_j, D_i \equiv D_j, P_i \equiv P_j, \tag{3}$$

$$\forall \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{E}, |t_i - t_j| \leq \tau. \tag{4}$$

4.2.1. Implementing Alert Correlation

A two-step method can be employed to address the correlation problem efficiently. The first step utilizes a clustering algorithm to group alerts in $\boldsymbol{X}$ based on their similarity. This algorithm partitions alerts into k ($k \leq N$) clusters using the numerical representations in $\boldsymbol{X}$, thereby ensuring that alerts within each cluster are more similar to each other than to alerts in other clusters. In the second step, a segmenting algorithm is employed to divide alerts in each cluster into sets of candidate events (denoted $\boldsymbol{E}_j$) that satisfy the three-tuple and timeout conditions. This procedure ensures that each candidate event comprises alerts exhibiting compactness in behavioral, spatial, and temporal aspects. Finally, to enable effective incident investigation, an event profile is generated for the identified group of alerts. The profile includes various information, such as attack type, attack time, affected assets, and attack methods.

When clustering the filtered alerts into events, selection of the similarity threshold has a critical impact on the resulting groups, where a lower threshold reduces the number of events to be investigated but increases the complexity of alerts within each event, thereby making incident handling more challenging. Conversely, a higher threshold reduces the complexity of alerts in each event but increases the number of events that require investigation. Practically, events with high purity are preferable because they do not involve dealing with multiple types of alerts within a single event, which makes the investigation more efficient.

Using numerical representations provided by one-hot encoding, which can result in a large number of replicated alerts, is equivalent to setting the similarity threshold to 1 in a hierarchical clustering algorithm. In our implementation, we leveraged this feature by requiring alerts in the same cluster to be homogeneous in terms of the one-hot encoding features for the selected categorical fields. This ensures that the resulting events comprise

alerts with consistent attributes, thereby improving the efficiency and accuracy of the incident investigation.

The segmentation algorithm used to divide a group of alerts into events ensures that alerts in the same event exhibit spatial compactness. Many existing NIDSs group alerts into events based on a four-tuple [source IP, destination IP, source port, destination port] or a three-tuple [source IP, destination IP, destination port] for ease of implementation. After carefully observing a substantial number of alerts and using the ATG as a visualization aid, we discovered that using a three-tuple consisting of [source IP, destination IP, service port] combined with a strict similarity criterion resulted in a noteworthy improvement on the grouping of security alerts into separate events. Additionally, our choice of the service port over the source or destination port corresponds to the incident handling expertise of SOC operators, as evidenced by the escalation set utilized during the assessment. Further details can be found in Appendix A.2. Here, the service port refers to the port whose vulnerability is exploited in the attack campaign. The source port on the attacker side is typically selected randomly, which makes it less useful for correlating alerts from the same attack campaign. For example, in a DoS attack against a web server, the source port may differ, but the service port (e.g., port 80 for HTTP services and port 8080 for HTTPS services) remains the same. Grouping alerts based on the proposed three-tuple provides an appropriate level of granularity for segmenting alerts into events.

We propose a simple technique to identify the service port in a group of alert messages produced by an IDS. Here, we assume that the alerts originate from a pair of IP addresses and that they occur within a short time span, which allows them to be merged into a single event. However, at least one of the source or destination ports may have multiple values. To determine the service port, we utilize the heuristic that, as the target of a series of similar attacks, it should appear in the alerts more frequently than randomly selected attacking ports. By comparing the information entropy of the source and destination ports, we can determine the service port as the one with lower entropy (with relatively high confidence). If the comparison does not yield a clear advantage, we utilize a combination of the source and destination ports as the service port identifier to merge alerts into events.

An implementation of the event segmenting algorithm is presented in Algorithm 1. The function takes a set of events, including source ports, destination ports, and issuing timestamps (s_i, d_i, and t_i), as well as a timeout parameter (τ) and threshold parameter (β). First, the algorithm calculates the number of unique source and destination ports, and then it divides the alerts into multiple events if there are several unique source and/or destination ports. Here, it utilizes entropy to measure the level of uncertainty along different ports. A higher entropy value indicates greater uncertainty in the specific port. The algorithm splits the events based on the port with lower entropy. The "SplitByTimeout" function then further segments events with the same service port. This function is straightforward to implement based on the temporal condition in Equation (4). Specific implementation details are omitted here for brevity. The β parameter controls the degree of bias toward one port, and in our experiment, this was set to 0.58.

Definition 4 (**Entropy** [69]). *Let Z be a random variable representing a set of source or destination ports observed in IDS alerts, with individual observations denoted z_i. Typically, z_i takes on integer values in the range 0–65,535. In addition, assume that Z is distributed according to a probability distribution $p : \mathcal{Z} \rightarrow [0, 1]$. Thus, the entropy of Z can be obtained as follows:*

$$H(Z) = -\sum_{z} p(z) \log p(z). \tag{5}$$

The value of the timeout parameter (τ) used to define an event associated with an attack campaign may vary depending on several factors, including the specific use case, the network being monitored, the types of detected attacks, and the alert-issuing IDS settings. Typically, this parameter is used to set a maximum time interval for grouping a series of related alerts or activities as part of the same attack campaign. In common IDSs,

timeout values for reissuing alerts can range from several seconds to several minutes (or even longer in some cases).

When using the timeout parameter to group closely related alerts that satisfy the three-tuple condition, it is more suitable to set a larger value to avoid the negative impact of repetitive escalation candidates, as found in our experiment. For example, we found that a timeout value of 7200 s was suitable to group hourly recurring, identical alerts into a single event and notifying the operator about alerts of the same type only if there is a gap of at least two hours between alerts.

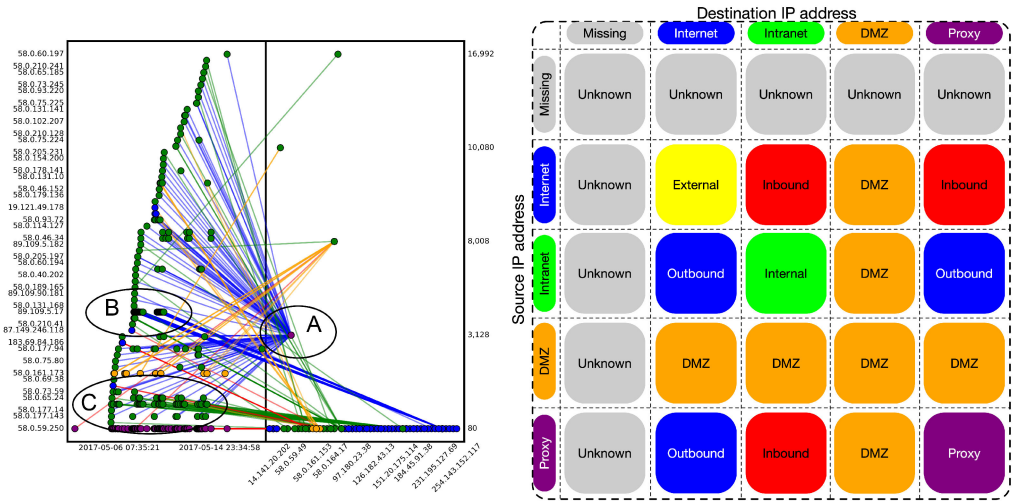
Algorithm 1 Event Segmenting.

Require: $0 \leq t_{i+1} - t_i \leq \tau$ for $i = 2, \ldots, N-1$
function SEGMENTING($\{[s_i, d_i, t_i]\}$, τ, β)
  $n_s \leftarrow$ CountUnique($\{s_i\}$)
  $n_d \leftarrow$ CountUnique($\{d_i\}$)
  if $n_s = 1$ **or** $n_d = 1$ **then** ▷ 1× multi or multi ×1
    $E \leftarrow \{[s_i, d_i, t_i]\}$
  else ▷ multi × multi
    $e_s \leftarrow$ Entropy($\{s_i\}$)
    $e_d \leftarrow$ Entropy($\{d_i\}$)
    if $e_s / e_d < \beta$ **then** ▷ source as service
      $E \leftarrow$ SplitByTimeout($\{s_i\}$, $\{t_i\}$, τ)
    else if $e_d / e_s < \beta$ **then** ▷ dest. as service
      $E \leftarrow$ SplitByTimeout($\{d_i\}$, $\{t_i\}$, τ)
    else ▷ port pair as service
      $E \leftarrow$ SplitByTimeout($\{[s_i, d_i]\}$, $\{t_i\}$, τ)
    end if
  end if
  return E
end function

4.3. Augmented Tile Graph Visualization

In this section, we reintroduce the tile graph method [61] as a way to visualize massive and interconnected alerts. The tile graph divides the plot area into two sub-panels, namely the source panel on the left and the destination panel on the right (Figure 3a). The x-axis of the source panel represents the timestamp of the triggered alert, while the y-axis indicates the source IP addresses involved in the communication. In contrast, the x-axis and y-axis of the destination panel represent the destination IP addresses and destination ports involved in the communication, respectively. To ensure the distinguishability of IP addresses, they are uniformly positioned on the axes based on their occurrence order. A line is drawn between a source IP address and a destination IP address to represent each alert. The color of the line reflects the status of the source and destination IP addresses, as indicated in Figure 3b. For instance, a blue line represents an alert triggered by an internal host communicating with an external host on the Internet.

Figure 3a shows an ATG illustrating the "HTTP login brute-force detected" attack over a one-week period. The ATG reveals that a considerable number of alerts were triggered by communications between internal hosts and the proxy server, as highlighted by the blue lines toward the proxy server enclosed by ellipse A. Although such erroneous communications may not be significant to a security operator, the operator may wish to concentrate on the more distinct attack attempts that display a periodic nature, as highlighted by ellipses B and C. The ATGs provide a valuable perspective of the complexity of cyberattack campaigns, which can help security operators reduce the costs associated with log parsing; thus, they can focus their attention on more critical incidents.



(**a**) (**b**)

Figure 3. Visualization of "HTTP login brute-force detected" attacks using the tile graph. (**a**) Tile graph and (**b**) Color-coded legend illustrating host locations and attack orientations within the tile graph. Hosts are represented by disks, with colors indicating their locations (green for internal, blue for external, orange for DMZ, purple for proxy server, and gray for missing information). Alerts are depicted by lines connecting source and destination hosts in the panels, with colors representing communication orientations (blue for outbound, red for inbound, green for internal, yellow for external, orange for DMZ-related, and gray for unknown). Ellipses A, B, and C in (**a**) highlight distinct host groups. Ellipse A emphasizes a proxy server, exhibiting outbound alerts between source internal hosts and the proxy server as the destination. Ellipse B exemplifies a source host triggering periodic outbound alerts to external destination hosts. Ellipse C demonstrates internal hosts triggering periodic internal alerts between other internal destination hosts. IP addresses in this example were anonymized through random shuffling, ensuring the absence of identifiable information.

The ATG is a powerful framework for visualizing complex datasets, particularly those containing source and destination information found in alert messages. By encoding this information into a set of coordinate variables, the ATG creates an intuitive visualization that facilitates pattern discovery. This approach can visualize various data types, as demonstrated in the context of visualizing the results of the proposed event segmenting algorithm. In this case, alerts occurring between two fixed IP addresses are visualized, thereby freeing up axes previously used to display IP addresses to encode other relevant information.

In the examples shown in Figure 4, the source plane uses the x-axis to represent time and the y-axis to represent source ports, and the destination plane uses the x-axis to represent attack types and the y-axis to represent destination ports. The color of the connecting line in the ATG represents the event label assigned by the algorithm. Here, related events are assigned the same color, which makes it easier to identify event clusters and draw conclusions regarding network behavior.

Figure 4a shows the effectiveness of the event segmenting algorithm. Here, ellipse A contains a group of alerts notifying attacks against port 22, detected as the same type of attack but divided into different events due to exceeding the timeout threshold. Ellipse B contains attacks targeting the same port with varying source ports in a short period, thereby satisfying the conditions of the proximal, spatial, and temporal compactness; thus, these attacks can be merged into a single event. In Figure 4b, ellipse C shows alerts notifying attacks against the same target port that were detected as different types of attacks. These attacks are divided into two events due to a relatively strict proximal criterion, and the same situation is observed with the alerts in ellipse D, indicating that the imposed proximal criterion may be too strong. Thus, there is room to implement a milder criterion to reduce the number of escalation candidates.

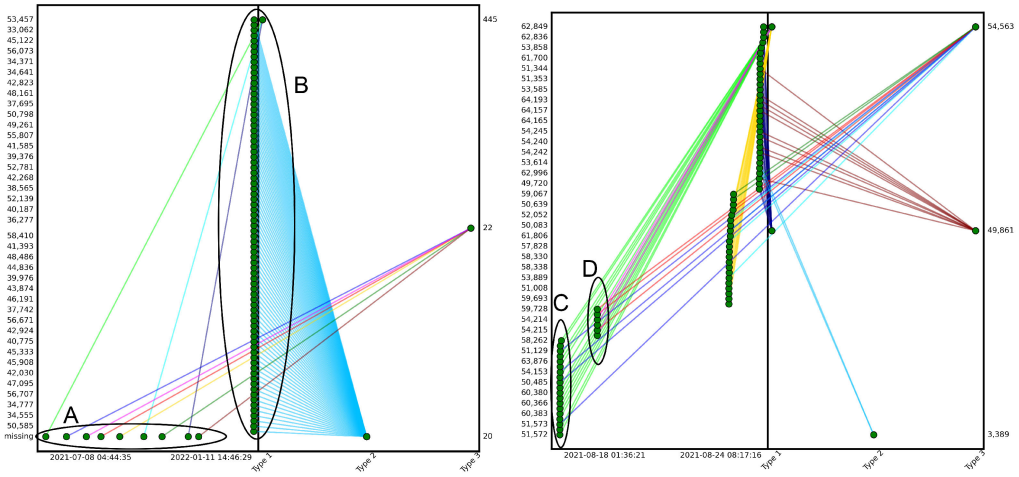


Figure 4. Visualization of the event segmentation algorithm results using ATGs. (**a**) Example 1 of event segmentation; (**b**) Example 2 of event segmentation. In the ATGs, hosts are depicted as disks, adhering to the color scheme defined in Figure 3b. Alerts are illustrated by lines connecting the source and destination hosts in the panels, with colors indicating the grouping outcome—alerts belonging to the same group share a common color. Ellipse A highlights a cluster of alerts indicating attacks on port 22 with prolonged time intervals. Ellipse B exemplifies a cluster of alerts targeting the same destination port but originating from different source ports within a short timeframe. Ellipse C highlights a cluster of alerts targeting the same destination port. However, alerts within this cluster are associated with distinct attack types, resulting in their grouping into multiple events. Ellipse D showcases another example where alerts targeting the same port are distributed across multiple events due to inconsistencies in other attack characteristics.

Overall, the ATG visualizes complex datasets effectively, as demonstrated by the results of the event segmentation algorithm. The power of data visualization is evident in this example because patterns are readily apparent through a visual inspection.

5. Experiments

In this section, we present the results of experiments conducted to evaluate the proposed scheme as applied to log data collected from multiple IDSs deployed in SOC operations in an enterprise network. We conducted two sets of experiments. The first aimed to make a horizontal comparison of the proposed cost-sensitive approach with previous methods to address the filtering problem. To enable comparison and citation, we used the performance evaluation results obtained through five-fold SCV. In the second set of experiments, we examined the effectiveness of the proposed method in a real-world SOC context, where we adopted ROV for the evaluation. In addition, we evaluated the enhancement of incident handling performance resulting from utilizing the proposed event segmenting algorithm to address the correlation problem.

5.1. Data Preparation

In this study, we analyzed real-world data from a class-B network with over 1000 users and 30,000 network devices, e.g., personal computers, servers, and mobile devices [59,61,70]. For anonymity purposes, the six IDSs are referred to as appliances A–F. We used two datasets collected from the same network at different times, which we refer to as the NSOC2017 and NSOC2022 datasets. The NSOC2017 dataset contains alert logs generated by all six IDSs from 1 January to 31 October 2017. The NSOC2022 dataset comprises alert logs from two appliances (A and B) from 18 June 2021 to 30 November 2022. These two appliances are considered the main sources of information for incident handling due to their long operation time and high-quality alert information. Appliances C–F were

discontinued for performance and commercial reasons. Using these datasets, we assessed the performance of the proposed SIEM scheme to improve incident handling.

The SOC security team examined the alerts and escalated those that could potentially impact critical systems. They analyzed the communication content associated with each escalation, checked the accessed URLs against commercial blacklists, analyzed downloaded files using a sandbox, and then recorded the results in the system. To minimize the impact of overlooked critical alerts caused by alert fatigue and single alerts that were escalated accidentally, we performed a review using data grouping techniques, ATGs, and signature-matching using various information, e.g., the URLs and hash values of the downloaded files, to identify alerts that were closely related to escalations. The SOC investigation and subsequent secondary review process helped us accurately identify critical alerts and improve the quality of the dataset.

5.2. Parameter Tuning

The effectiveness of machine learning algorithms is heavily influenced by the hyper-parameters utilized during the training process. In order to tune the classifiers' parameters, we implemented the 10-fold SVC procedure explained in Section 3.3.4. Following this, we chose the optimal setting to create a model based on the entire training set, which was then assessed using the test set. The parameter values we examined for each classifier are presented in Table 1. For standard classifiers such as KNN, NB, LDA, DT, AdaBoost, and SVM, we utilized the parameter tuning function available in MATLAB to select the best parameters for the final assessment [71]. It is important to note that we used this parameter tuning method to find a weight parameter that could effectively balance the loss between positive and negative alerts.

Table 1. Experimental settings for parameter tuning.

Parameter	Classifiers	Physical Meaning	Grid Values
W	WSVM, IWSVM	Weight for positive class	{1, 10, 100, 1000, 10,000}
γ	SVM, WSVM, IWSVM	Width parameter for SVM with RBF kernels	{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1}
C	SVM, WSVM, IWSVM	Penalty parameter for prediction error	{10, 100,1000, 10,000}

5.3. Evaluation Metrics

The generalization performance of the selected algorithms was evaluated using four common supervised learning metrics: accuracy, precision, recall, and false positive rate (FPR). In addition, as the alert filtering task has a highly imbalanced class distribution, we also used the F1-score, true negative rate (TNR), and balanced classification rate (BCR) to provide a comprehensive evaluation [72]. The definitions of these metrics are related to the following intermediate measures.

True positive (TP) refers to the number of records correctly predicted as positive by the classifier, and false positive (FP) is the number of records incorrectly predicted as positive by the classifier. True negative (TN) is the number of records correctly predicted as negative by the classifier, and false negative (FN) is the number of records incorrectly predicted as negative by the classifier.

Accuracy represents the percentage of test label data correctly identified by the classifier, calculated as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N}, \tag{6}$$

where N is the size of the test set.

Precision is the probability that the predicted positive records are classified correctly:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{7}$$

Recall is the probability that a record from the positive class is classified correctly:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{8}$$

The FPR measures the probability that a negative record is predicted incorrectly as a positive record by the classifier:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{9}$$

The F1-score is the harmonic mean of precision and recall:

$$\text{F1} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}. \tag{10}$$

TNR is the number of negative samples classified correctly divided by the total number of negative samples:

$$\text{TNR} = \frac{\text{TN}}{\text{FP} + \text{TN}}. \tag{11}$$

BCR is the average of the recall and the positive class TNR:

$$\text{BCR} = \frac{\text{Recall} + \text{TNR}}{2}. \tag{12}$$

5.4. Experiment I

The statistics for the NSOC2017 dataset are given in Table 2, revealing a wide variation in the number of alerts depending on different detection mechanisms, with some IDSs generating only a few dozen alerts and others producing several hundred thousand. Out of the 133.77 million alerts in the dataset, only 593 were escalated as part of incident handling. To expedite the review process, we downsampled the negative class, resulting in an evaluation dataset with approximately 0.488% negative samples while retaining all positive samples. This process led to a substantial increase in the number of positive samples (from 593 to 2239). The NSOC2017 dataset includes 2239 positive samples and 672,000 negative samples, with a positive sample ratio of approximately 0.333%, highlighting the challenge of class imbalance in the classification task. Note that we adopted a subsampling approach used in previous studies [70] to maintain comparability.

Table 2. Statistics of NSOC2017 dataset.

Appliance	#Total	#Daily Average	#Subsampled	Sampling Ratio (%)	#Positive	Positive Ratio (%)
A	36.68 M	39,005	226,784	0.618	1286	0.567
B	0.56 M	1857	5944	1.053	617	10.380
C	11.86 M	39,005	30,837	0.260	169	0.548
D	15.80 M	51,953	36,145	0.229	20	0.055
E	66.21 M	217,804	372,161	0.562	53	0.014
F	6498	21	129	1.985	94	72.868
Total	137.77 M	452,958	672,000	0.488	2239	0.333

A prominent '#' symbol in the header line indicates that the corresponding item represents a count of a specific entity.

5.4.1. Visualization by *t*-SNE

To gain a deeper understanding of the data distribution, we used the *t*-SNE [73] technique to generate a 2D visualization of the data. The *t*-SNE approach is highly effective at reducing the dimensionality of nonlinear data, enabling the approximation of complex relationships among high-dimensional data in a lower-dimensional space. The method is

computationally efficient and can effectively handle complex data distributions in high-dimensional spaces.

The visualization result is shown in Figure 5, where each colored disk represents a unique sample, and its size is proportional to the number of alerts with the same numerical representation. Positive alerts are marked by black circles around the respective disks. As can be seen, although most IDSs conform to the CEF convention, alerts from different IDSs are spread throughout the space, and those from the same IDSs tend to cluster together. To better understand the formed clusters, we examined the alerts from IDS A (represented by red disks). We found that loose clusters were initially formed by alerts belonging to the same genre, e.g., HTTP communications, which share common features of TCP protocol-based communication. Subsequently, tighter clusters were formed by samples belonging to the "suspicious malware downloading" category. The differences in alert features, e.g., detection rules, download file types, and alert severity, result in clusters with closer proximity within this category.

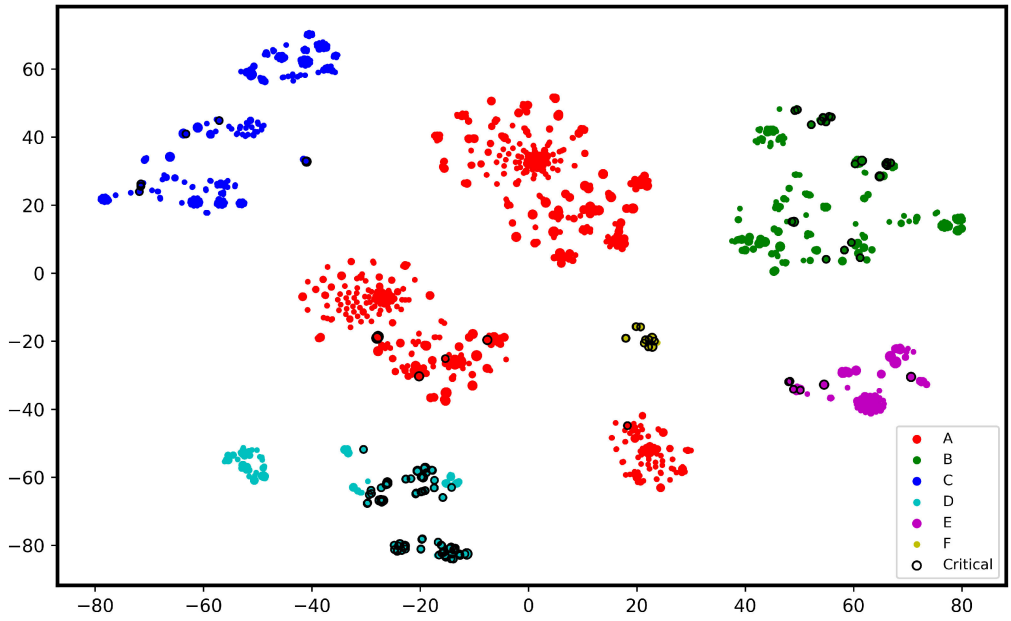


Figure 5. Visualization of unique alerts using *t*-SNE. The size of the disks is directly proportional to the weight of the sample being evaluated. The six NIDSs (A to F) included in the NSOC2017 dataset I are differentiated by the unique colors assigned to them. Disks that are associated with critical alerts are encompassed by a prominent dark boundary.

The most significant observation from this visualization is that positive samples (enclosed in black circles) form tight clusters and are clearly separable from negative samples. This high degree of separability between positive and negative samples suggests that the performance of the malware detection model is likely good.

5.4.2. Numerical Results

Table 3 compares the performance of various ML algorithms on the NSOC2017 dataset. The compared algorithms include unsupervised learning methods [40,41], the common supervised learning methods in Section 3.3, oversampling methods [59,70], and cost-sensitive methods, including the WSVM and IWSVM methods (Section 4.1.1). Note that the unsupervised methods [40,41] (IF and IF + DC) were not designed for the heterogeneous alerts in the NSOC2017 dataset; however, we refer to their results for reference (even though they are not directly comparable to the classification methods used in this study).

The results clearly demonstrate that the highly imbalanced class distribution causes poor generalization performance with the existing classifiers. In terms of the F1-measure,

the top-three performing algorithms are IWSVM, WSVM, and AdaBoost, which outperformed the other methods by a substantial margin.

Among the supervised learning methods, the LDA and AdaBoost methods obtained a very high recall rate of 99.732%, with slightly higher FPR values of 0.016% and 0.002%, respectively. The SVM method obtained the lowest FPR of 0.001% with a relatively low recall rate of 97.901%.

Table 3. Performance comparison on NSOC2017 dataset.

Category	Algorithm	Accuracy (%)	Recall (%)	Precision (%)	FPR (%)	TNR (%)	F1 (%)	BCR (%)
Unsupervised learning	IF [40]	90.705	100.000	0.551	9.300	90.700	1.097	95.350
	IF + DC [41]	94.141	95.876	0.837	5.860	94.140	1.659	95.008
Supervised learning	KNN	99.955	98.749	89.046	0.041	99.959	93.647	99.354
	NB	89.741	93.390	2.950	10.271	89.729	5.719	91.559
	LDA	99.983	**99.732**	95.387	0.016	99.984	97.511	99.858
	DT	99.916	99.285	80.137	0.082	99.918	88.689	99.602
	AdaBoost	99.997	**99.732**	99.466	0.002	99.998	99.599	**99.865**
	SVM	99.992	97.901	99.682	**0.001**	**99.999**	98.783	98.950
Over-sampling	RF-SMOTE [70]	99.914	99.048	95.632	0.072	99.928	97.310	99.488
	DT-SMOTE [70]	99.573	99.286	78.977	0.422	99.578	87.975	99.432
	RF-SVMSMOTE [70]	99.925	98.571	96.729	0.053	99.947	97.642	99.259
	DT-SVMSMOTE [70]	99.562	99.286	78.531	0.443	99.567	87.697	99.426
	XGBoost-MAS-OSS [59]	99.875	99.519	93.034	0.119	99.881	96.167	99.700
Cost-sensitive learning	WSVM	**99.998**	99.598	**99.687**	**0.001**	**99.999**	99.643	99.799
	IWSVM	**99.998**	99.643	**99.687**	**0.001**	**99.999**	**99.665**	99.821

Bold numbers signify the utmost values attained on the particular evaluation criteria.

For the oversampling-based methods, the combination of random forest (RF) and SVMSMOTE yielded comparatively good results, while the DT method did not exhibit significant improvement. The XGBoost-MAS-OSS algorithm [59] leverages four distinct oversampling methods to generate more diverse artificial data and utilizes data cleaning for preprocessing, which results in notable performance enhancements.

The WSVM assigns a high weight to positive samples to address imbalanced class distribution, which allowed it to outperform the conventional SVM method on all seven evaluation criteria, achieving a recall rate of 99.598% and the lowest FPR of 0.001%. IWSVM, which extends the WSVM method by assigning different weights to samples with different importance, further improved the recall rate to 99.643% with the lowest FPR value. In addition, the WSVM and IWSVM methods both achieved recall rates greater than 99.5%, indicating that very few critical alerts were missed by the filtering process, which is a significant advantage for SOC operation. The extremely low FPR values obtained by WSVM and IWSVM suggest that false alerts were minimized to ensure efficient incident response.

5.5. Experiment II

The IDS alerts collected in the NSOC2022 dataset comprise a vast amount of data, with 7.3 million lines of alert logs extracted from only two appliances between 18 June 2021 and 30 November 2022. We employed ROV to evaluate the effectiveness of the proposed SIEM scheme in a real-world SOC context. To present the results clearly, we only report the outcome for IWSVM in this experiment due to its exceptional performance.

Table 4 summarizes the training and testing dataset used to evaluate model accuracy in terms of ROV. In the first configuration, we trained the model using alert data collected between June 2021 and June 2022, and we tested it on data from July 2022 to November 2022. For the subsequent four configurations, we gradually removed the earliest month's data from the testing set and added these data to the training dataset. This allowed us to observe the changes in prediction accuracy when the training and testing datasets had different time intervals.

Table 4. ROV settings to evaluate IWSVM on NSOC2022 dataset.

	Training Set				Testing Set				#Escalated
Setting	**Interval**	**#All**	**#Positive**	**#Negative**	**Interval**	**#All**	**#Positive**	**#Negative**	**#All**
1	June 2021–June 2022	5.112 M	56,530	5.056 M	July 2022–November 2022	2.213 M	25,591	2.187 M	4165
2	June 2021–July 2022	5.648 M	61,250	5.587 M	August 2022–November 2022	1.677 M	20,871	1.656 M	3428
3	June 2021–August 2022	6.130 M	67,116	6.063 M	September 2022–November 2022	1.195 M	15,005	1.180 M	2935
4	June 2021–September 2022	6.546 M	71,758	6.475 M	October 2022–November 2022	0.779 M	10,363	0.768 M	2567
5	June 2021–October 2022	6.989 M	77,692	6.912 M	November 2022	0.336 M	4429	0.331 M	693

A prominent '#' symbol in the header line indicates that the corresponding item represents a count of a specific entity.

5.5.1. Filtering Result

Table 5 summarizes the performance of the IWSVM models obtained using different ROV settings on the NSOC2022 dataset. The last row of the table shows the results obtained by randomly sampling the dataset with 70% of the data used for training and 30% for testing. The results show that as the time interval between training and testing data collection decreases, the predictive accuracy of IWSVM increases. In a follow-up study, we confirmed that the prediction errors are primarily due to alerts triggered by new attacks. Thus, updating the training dataset to reflect the latest attack information is crucial for enhancing predictive accuracy. Across all settings, IWSVM achieved an F1-score of over 99.3% and a recall value approaching 99.0%, conclusively demonstrating its effectiveness in addressing the filtering problem.

Table 5. ROV performance with different prediction intervals by IWSVM on NSOC2022 dataset.

Setting	**Interval**	**Accuracy (%)**	**Recall (%)**	**Precision (%)**	**FPR (%)**	**TNR (%)**	**F1** (%)	**BCR (%)**
1	5 months	99.986	98.945	99.822	0.002	100.000	99.381	99.472
2	4 months	99.995	99.731	99.846	0.002	100.000	99.788	99.865
3	3 months	99.994	99.732	99.812	0.002	99.998	99.772	99.865
4	2 months	99.991	99.611	99.728	0.004	99.996	99.669	99.804
5	1 month	99.999	99.932	99.977	0.000	100.000	99.954	99.966
Random Shuffle	-	100.000	1.000	99.996	0.000	100.000	99.998	50.500

We then obtained a precise estimate of the performance improvements that can be achieved by implementing the proposed SIEM scheme for incident handling. Here, we must minimize the impact of labeling inaccuracies caused by operator alert fatigue. To achieve this objective, we used the preliminary escalation set that was initially selected by the security team. This set comprises alerts that have undergone some level of manual confirmation prior to escalation. This set is best suited as a validation set to measure the effectiveness of the proposed scheme. The information of the preliminary escalation set is summarized in the last column of Table 4 (labeled "Escalated").

Using the experimental setup of the previous trial, we conducted an evaluation on the preliminary escalation set included in the NSOC2022 dataset. The results are shown in Table 6. Consistent with the results given in Table 5, the IWSVM obtained exceptional performance on this dataset, achieving an F1-score of over 98% for all settings. As can be seen, a shorter time interval between the training and testing samples yielded superior pre-

dictive performance. In addition, the table summarizes the alert reduction rate (ARR) [16], which is calculated as follows:

$$\text{ARR} = (1 - \frac{\text{TP} + \text{FP}}{N}) \times 100\%. \quad (13)$$

The ARR assesses the reduction of irrelevant alerts resulting from the application of IWSVM. Across all settings, IWSVM achieved close to or greater than a 50% ARR value.

Table 6. ROV performance with different prediction intervals by IWSVM on preliminary escalations on NSOC2022 dataset.

Setting	Interval	Accuracy (%)	Recall (%)	Precision (%)	FPR (%)	TNR (%)	F1 (%)	BCR (%)	ARR
1	5 months	98.079	97.561	98.940	1.284	98.716	98.246	98.138	45.850
2	4 months	97.987	97.109	99.314	0.872	99.128	98.199	98.119	45.010
3	3 months	98.331	97.361	99.394	0.634	99.366	98.367	98.364	49.400
4	2 months	98.091	96.928	99.292	0.712	99.289	98.096	98.108	50.490
5	1 month	99.423	99.180	99.725	0.306	99.694	99.452	99.437	47.470

5.5.2. Correlation Result

By utilizing the event segmenting algorithm, related alerts can be consolidated into a single event, reducing the burden of the incident handling workload. In our experiment, we applied the proposed event segmenting algorithm to merge alerts from the preliminary escalation set into events. We then evaluated the performance of IWSVM based on the derived events. As shown in Table 7, the proposed correlation algorithm effectively consolidates related alerts, resulting in a significant reduction in the number of events to investigate.

Table 7. ROV performance of preliminary escalation sets on an event basis.

Setting	Interval	#Event	Accuracy (%)	Recall (%)	Precision (%)	FPR (%)	TNR (%)	F1 (%)	BCR (%)	GARR (%)
1	5 months	1512	96.958	97.171	98.611	3.606	96.394	97.886	96.783	74.070
2	4 months	1173	96.249	96.261	98.397	3.779	96.221	97.317	96.241	76.340
3	3 months	888	97.072	97.152	98.472	3.093	96.907	97.808	97.030	79.930
4	2 months	718	96.379	96.288	98.000	3.462	96.538	97.137	96.413	82.470
5	1 month	206	99.029	99.099	99.099	1.053	98.947	99.099	99.023	83.980

To evaluate the efficacy of combining filtering and correlation methods to reduce the number of alerts requiring investigation, we introduced the generalized ARR (GARR) concept:

$$\text{GARR} = \left(1 - \frac{\text{TP}_\text{e} + \text{FP}_\text{e}}{N}\right) \times 100\%, \quad (14)$$

where TP_e and FP_e are TP and FP events, respectively. GARR assesses the rate at which alerts are safely reduced by combining the filtering and correlation methods. As shown in Table 7, combining these techniques reduced the alert volume by nearly or greater than 75% on the preliminary escalation set, indicating the effectiveness of the proposed methodology.

In terms of generalizability, IWSVM maintained high accuracy, with an F1-score of over 97% under all settings throughout the event-based evaluation.

6. Discussion, Limitations, and Future Work

In this section, we discuss the limitations of the proposed framework and identify potential directions for further improvement.

6.1. Limitation of Scope

In this paper, we have focused on using AI/ML and visualization techniques to enhance the efficiency of SIEM systems by eliminating redundant and irrelevant alerts.

While the discussion is limited to this specific issue, the proposed framework can potentially be applied to other security tasks and integrated into various security systems, and these possibilities will be explored in future work.

6.2. Ethical Implications

Integrating the outcomes of NIDSs and other security devices can enhance the effectiveness of the proposed SIEM scheme by reducing irrelevant and duplicate alerts, leading to improved incident handling performance. However, the use of an NIDS raises ethical concerns that must be considered. Below we list some ethical implications of conducting integration and analysis on security alerts using SIEMs [74,75].

- Privacy: Security alerts contain sensitive data, e.g., personal information, usernames, and passwords. Thus, analyzing security alerts can raise privacy concerns if these alerts are stored without appropriate safeguards.
- Data security: NIDS-collected data integrated in the SIEM can be of great value to attackers. Thus, the SIEM must have robust security measures in place to protect the collected data.
- False positives: The proposed SIEM scheme attempts to reduce the number of FPs; however, some may remain. This can result in unnecessary investigations and harm innocent users. To ensure accountability, the proposed visualization technique can summarize the security situation associated with the alert.
- Legal compliance: Similar to NIDSs, the SIEM must comply with applicable laws and regulations, particularly those related to data protection and privacy.

Organizations can mitigate these ethical concerns by implementing appropriate policies and procedures for NIDS and SIEM usage. These policies should include comprehensive data collection and retention guidelines as well as procedures to handle FPs and protect user privacy. In addition, organizations must ensure that NIDSs and SIEMs comply with legal and regulatory requirements, and that these systems are audited and reviewed regularly.

6.3. Technical Limitations

In the following, we discuss the technical limitations of the proposed SIEM scheme and provide guidelines for its deployment in real enterprise SOCs.

6.3.1. Bias in the Results

In this study, we analyzed real-world data in a SOC using the proposed scheme and evaluated it based on an escalation set obtained through actual daily operations. The reported improvements in the escalation results indicate a significant enhancement in the system's performance. However, incident response practices are influenced by various factors. In particular, the escalation set was initially obtained from installed security appliances, integrated by the SIEM system currently in use, and then manually selected by the security operators. The results are heavily reliant on the security policy and implementation of the enterprise network as well as the experience and condition of the technicians and analysts involved in the process, e.g., the occurrence of alert fatigue.

Thus, rather than pursuing an all-encompassing security solution, we presented a SIEM scheme that shows promise for a specific SOC implementation as well as easy adaptability and customization. Achieving a fully objective and impartial comparison on a larger scale dataset that covers more attack types and different case studies considering other network environments is beyond the scope of this paper. These issues will be pursued in future work.

6.3.2. Coping with Detection Errors

The primary objective of this study is to improve the efficiency of incident handling by reducing false positives in the escalation set. Therefore, performance evaluation in the experiment focused on the positive class, i.e., critical alerts that serve as the escalation set in incident response. However, it is crucial to note that eliminating all false positives may not always be feasible and could even lead to an increased risk of false negatives, which could

compromise the security of the network infrastructure. In such scenarios, we recommend setting a high threshold value for the expected recall rate, such as 99%, and opting for a detection model that meets this requirement while having the fewest number of false positives.

In addition, incident response is primarily a manual process conducted by security operators, who can help identify false positives by manually reviewing alerts selected by the SIEM system. Integrating feedback from the security operations can refine the algorithm and reduce false positives over time. To achieve this, it is essential for the security operators to have a clear understanding of the SIEM system's mechanism and provide valuable feedback for performance improvement.

6.3.3. Emerging Threats and Adversarial Attacks

The effectiveness of the proposed framework relies heavily on the accuracy and reliability of input data. However, changes in the network infrastructure or new attack vectors, including intentional adversarial attacks, can negatively impact the framework's performance by resulting in false negatives. To address this limitation, it is important to fine-tune the NIDS detection rules and use machine learning-based NIDS with anomaly detection capabilities to ensure that emerging threats and adversarial attacks are successfully detected. Additionally, advanced machine learning techniques and data cleaning methods can be employed to further improve the framework's effectiveness. Moreover, continuously monitoring for and adapting to new threats is also crucial to ensure that the framework remains effective in the long term.

6.3.4. Guidelines for Model Updating

Experiment II showed that the detection performance gradually degrades as the interval between the training and testing sets increases. To keep an AI model up-to-date and to adapt to new threats, regularly updating the detection model is crucial. Here are guidelines to follow:

Regularly update the training dataset with the latest threat intelligence; Incorporate new features to capture emerging attack techniques; Periodically retrain the AI model with the latest dataset and new features; Continuously evaluate the model's performance and identify areas that require improvement; Incorporate feedback from security teams to refine the AI model and improve its detection capabilities.

6.4. Future Work

While this study has provided insights into the potential benefits of utilizing AI and visualization techniques in the SOC alert management context, there are several avenues for future work. For example, the proposed framework should be implemented and evaluated in a real-world environment. Thus, to address concerns about implementation and scalability, we plan to collaborate with industry partners to conduct a pilot study in a real-world SOC environment. This will provide an opportunity to gather feedback from security analysts and IT professionals and to assess the framework's ability to integrate with existing security systems and to scale in order to satisfy the requirements of large organizations.

Finally, we recognize the need to conduct a more comprehensive evaluation of the proposed framework using a larger dataset and to cover a wider range of attack types and network environments. We believe that these future directions will further advance the field of AI in cybersecurity and contribute to the development of more effective and efficient SOC solutions.

7. Conclusions

This paper has proposed an effective solution to the alert fatigue problem in enterprise SOC operations. The proposed framework utilizes advanced AI/ML and data visualization tools to address the filtering and correlation problems. By breaking down the problem into these components and introducing practical solutions, including a cost-sensitive learning method and an algorithm to summarize alerts into closely related events, the proposed

framework effectively reduces FPs and increases the recall rate. In addition, an ATG visualization tool was introduced to improve the performance of data analysis against security alerts. This study offers valuable contributions toward the development of a next-generation SIEM system and opens opportunities for further research in this field.

Author Contributions: Conceptualization, T.B., T.T. and D.I.; methodology, T.B.; software, T.B. and S.N.; validation, T.B.; formal analysis, T.B.; investigation, T.B.; resources, T.B., and D.I.; data curation, T.B.; writing-original draft preparation, T.B.; writing-review and editing, T.B.; visualization, T.B.; project administration, T.T.; funding acquisition, T.T. and D.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministry of Internal Affairs and Communications grant number JPJ000254.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The enterprise datasets produced and examined in this study are not accessible to the public, and we are unable to provide them under any circumstances. The source code will be available upon request.

Acknowledgments: This research was conducted under a contract of "MITIGATE" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Appendix A

Appendix A.1. Example of Security Log in CEF and JSON Format

In this appendix, we provide an example alert issued by an NIDS and its corresponding JSON object parsed from the alert with a unified list of item and value pairs. Figure A1a shows the CEF log of the alert, which contains event information such as unique identifiers, impacted IP addresses, risky URLs, and file hash values. This information is then interpreted and transformed into a structured JSON object, as shown in Figure A1b.


```
CEF:0|XXXXXXXX|Enterprise|9.1|file-download|Suspicious File Download|10|act=LOG cat=Malicious File Download/Malicious File Download cn1=
100 cn1Label=impact cn2=137####cn2Label=IncidentId cn3=100 cn3Label=IncidentImpact cnt=2 cs1=bf3eaa61:########:######## cs1Label=detecti
onId cs2=https://XXXX.XXXXX.XXXXXXX.XXXX.XX.XX/portal#/event/287149####/33094####/1210###?event_time\=2022-10-18 cs2Label=EventDetailLi
nk cs3=http://###.##.175.5/wget.sh cs3Label=EventUrl cs4=Script cs4Label=fileCategory cs5=####adb5729ccae6afb3c1fcb1e19a1a4287#### cs5La
bel=fileSHA1 cs6=https://XXXX.XXXXX.XXXXXXX.XXXX.XX.XX/portal#/analyst/task/####66ea7aeb000001027e2f42a0####/overview cs6Label=FileDetai
lLink deviceExternalId=287149####:33094#### dhost=###.##.175.5 dpt=80 dst=###.##.175.5 end=Oct 19 2022 00:12:00 JST externalId=1210####f
ileHash=####d5d08b659b251ca221cf3642#### fileType=Shell script text fname=/wget.sh fsize=1598 proto=TCP src=###.###.224.2 start=Oct 19 2
022 00:10:55 JST
```


(**a**)


```
{
  "act":"log",
  "attack":"malicious file download/malicious file download",
  "attack detail":"bf3eaa61:########:########",
  "class":"suspicious file download",
  "count":2,
  "dest host":"###.##.175.5",
  "dest ip":"###.##.175.5",
  "dest port":80,
  "dvc id":"28714####:33094####",
  "end time":"oct 19 2022 00:12:00 jst",
  "event type":"file-download",
  "external id":"1210####",
  "file category":"script",
  "file hash":"####d5d08b659b251ca221cf3642####",
  "file name":"/wget.sh",
  "file sha1":"####adb5729ccae6afb3c1fcb1e19a1a4287####",
  "file size":1598,
  ...
    ...
    "file type":"shell script text",
    "file url":"https://XXXX.XXXXX.XXXXXXX.XXXX.XX.XX/portal/...",
    "impact":10,
    "incident id":"137####",
    "incident score":100,
    "ip versioin":4,
    "product":"enterprise",
    "protocol":"tcp",
    "reference url":"https://XXXX.XXXXX.XXXXXXX.XXXX.XX.XX/portal/...",
    "received time":"oct 18 15:12:56",
    "source ip":"###.###.224.2",
    "start time":"oct 19 2022 00:10:55 jst",
    "type":"file-download",
    "url":"http://###.##.175.5/wget.sh",
    "vendor":"XXXXXXXX",
    "version":"9.1"
}
```


(**b**)

Figure A1. Example alert before and after reformatting. To protect sensitive information in the alert, identifying textual content is partially replaced with "X" symbols, and numbers are partially replaced with "#" symbols. (**a**) Example alert in CEF format; (**b**) Example alert formatted as a unified JSON object.

Appendix A.2. Using Service Port for Effective Alert Correlation

This appendix delves deeper into the three-tuple [source IP, destination IP, service port] that is used to correlate and group alerts into events.

In Figure A2, we present several real-life examples to illustrate why the service port is a better option than the source and destination ports in this context. Each ATG in the figure depicts alerts generated during communication between two hosts over a specific period. The x-axis of the source panel shows the time of issuance, while the y-axis represents the source port of the communication. The x-axis of the destination panel denotes the type of alerts (with detailed information omitted for clarity), while the y-axis represents the destination port. To distinguish groups of alerts with different features, we used red and blue circles. The red circles indicate groups of alerts that share the same source ports, while the blue circles highlight groups that share the same destination ports. It is clear that selecting either the source or destination port as one of the three tuples for correlation analysis cannot cover both cases simultaneously. However, by utilizing the technique proposed in Section 4.2.1 to determine the service port and using it to correlate alerts into events, we can effectively address all the cases demonstrated in the figure.

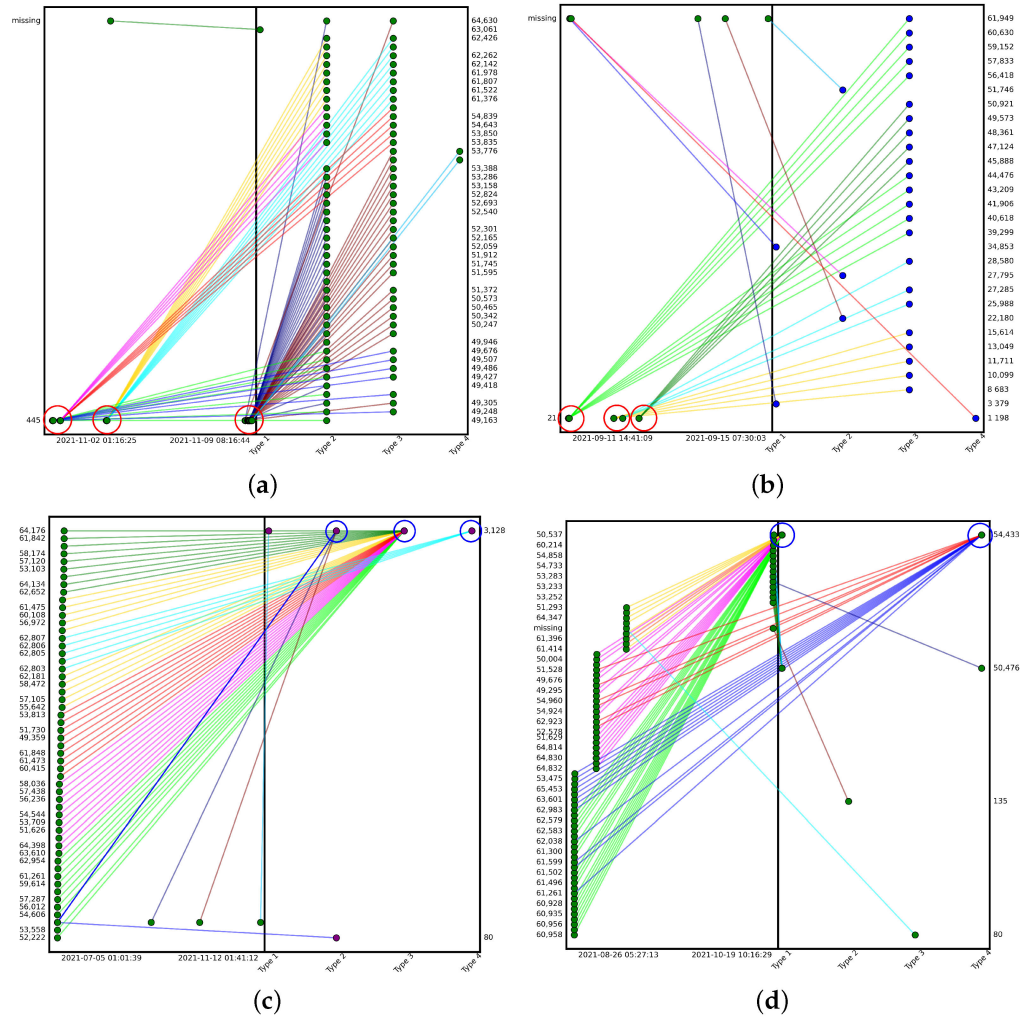


Figure A2. Using ATGs to visualize correlations between alerts. (**a**) Alert correlation example 1; (**b**) Alert correlation example 2; (**c**) Alert correlation example 3; (**d**) Alert correlation example 4. In the ATGs, hosts are represented as disks, adhering to the color scheme defined in Figure 3b. Alerts are depicted as lines connecting the source and destination hosts within the panels, with unique colors signifying the grouping outcome. Alerts belonging to the same group are assigned a consistent color. The red circles emphasize selected alert clusters characterized by shared source ports, while the blue circles highlight selected groups demonstrating shared destination ports.

References

1. European Union Agency for Cybersecurity. ENISA Threat Landscape. 2021. Available online: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021 (accessed on 30 January 2023).
2. European Union Agency for Cybersecurity. ENISA Threat Landscape. 2022. Available online: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022 (accessed on 30 January 2023).
3. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 686–728. [CrossRef]
4. Mohammadpour, L.; Ling, T.C.; Liew, C.S.; Aryanfar, A. A Survey of CNN-Based Network Intrusion Detection. *Appl. Sci.* **2022**, *12*, 8162. [CrossRef]
5. Gu, G.; Zhang, J.; Lee, W. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In Proceedings of the 2008 Network and Distributed System Security Symposium, NDSS, The Internet Society, San Diego, CA, USA, 10–13 February 2008.
6. O'Kane, P.; Sezer, S.; McLaughlin, K. Obfuscation: The Hidden Malware. *IEEE Secur. Priv.* **2011**, *9*, 41–47. [CrossRef]
7. Roesch, M. Snort: Lightweight intrusion detection for networks. In Proceedings of the LISA, Seattle, WA, USA, 7–12 November 1999; pp. 229–238.
8. Paxson, V. Bro: A system for detecting network intruders in real-time. *Comput. Netw.* **1999**, *31*, 2435–2463. [CrossRef]
9. Julien, V. Suricata IDS. 2015. Available online: https://suricata.io/ (accessed on 30 January 2023).
10. VMware. Advanced Threat Prevention with VMware NSX Distributed Firewall. 2021. Available online: https://business-iq.net/assets/8079-advanced-threat-prevention-with-vmware-nsx-distributed-firewall (accessed on 30 January 2023).
11. Fireeye2022. FireEye Network Security: Effective Protection against Cyber Breaches for Midsize to Large Organizations. 2022. Available online: https://docplayer.net/81314407-Fireeye-network-security.html (accessed on 30 January 2023).
12. TrendMicro. Machine Learning and Next-Generation Intrusion Prevention System (NGIPS). 2022. Available online: https://documents.trendmicro.com/assets/wp/WP01_Machine_Learning_170608US.pdf (accessed on 30 January 2023).
13. Vaarandi, R.; Podins, K. Network IDS alert classification with frequent itemset mining and data clustering. In Proceedings of the International Conference on Network and Service Management, Niagara Falls, ON, Canada, 25–29 October 2010; pp. 451–456.
14. McAfee. Alert Fatigue: 31.9% of IT Security Professionals Ignore Alerts. 2017. Available online: https://www.mcafee.com/blogs/enterprise/cloud-security/alert-fatigue-31-9-of-it-security-professionals-ignore-alerts/ (accessed on 30 January 2023).
15. González-Granadillo, G.; González-Zarzosa, S.; Diaz, R. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. *Sensors* **2021**, *21*, 4759. [CrossRef]
16. Kidmose, E.; Stevanovic, M.; Brandbyge, S.; Pedersen, J. Featureless Discovery of Correlated and False Intrusion Alerts. *IEEE Access* **2020**, *8*, 108748–108765. [CrossRef]
17. Bijone, M. A Survey on Secure Network: Intrusion Detection & Prevention Approaches. *Am. J. Inf. Syst.* **2016**, *4*, 69–88.
18. Walling, S.; Lodh, S. A Survey on Intrusion Detection Systems: Types, Datasets, Machine Learning methods for NIDS and Challenges. In Proceedings of the 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 3–5 October 2022; pp. 1–7. [CrossRef]
19. Liu, C.; Gu, Z.; Wang, J. A Hybrid Intrusion Detection System Based on Scalable K-Means+ Random Forest and Deep Learning. *IEEE Access* **2021**, *9*, 75729–75740. [CrossRef]
20. Donkol, A.A.E.B.; Hafez, A.G.; Hussein, A.I.; Mabrook, M.M. Optimization of Intrusion Detection Using Likely Point PSO and Enhanced LSTM-RNN Hybrid Technique in Communication Networks. *IEEE Access* **2023**, *11*, 9469–9482. [CrossRef]
21. Jayalaxmi, P.L.S.; Saha, R.; Kumar, G.; Conti, M.; Kim, T.H. Machine and Deep Learning Solutions for Intrusion Detection and Prevention in IoTs: A Survey. *IEEE Access* **2022**, *10*, 121173–121192. [CrossRef]
22. Ali, I.; Enezi, S.; Ali, F.; Kehar, A.; Fatima, K.; Uddin, M.; Karuppayah, S. Detection of Real-Time Malicious Intrusions and Attacks in IoT Empowered Cybersecurity Infrastructures. *IEEE Access* **2023**, *11*, 9136–9148. . [CrossRef]
23. Okey, O.D.; Melgarejo, D.C.; Saadi, M.; Rosa, R.L.; Kleinschmidt, J.H.; Rodríguez, D.Z. Transfer Learning Approach to IDS on Cloud IoT Devices Using Optimized CNN. *IEEE Access* **2023**, *11*, 1023–1038. [CrossRef]
24. Alohali, M.A.; Elsadig, M.; Al-Wesabi, F.N.; Al Duhayyim, M.; Mustafa Hilal, A.; Motwakel, A. Enhanced Chimp Optimization-Based Feature Selection with Fuzzy Logic-Based Intrusion Detection System in Cloud Environment. *Appl. Sci.* **2023**, *13*, 2580. [CrossRef]
25. Bour, H.; Abolhasan, M.; Jafarizadeh, S.; Lipman, J.; Makhdoom, I. A multi-layered intrusion detection system for software defined networking. *Comput. Electr. Eng.* **2022**, *101*, 108042. [CrossRef]
26. Awotunde, J.B.; Folorunso, S.O.; Imoize, A.L.; Odunuga, J.O.; Lee, C.C.; Li, C.T.; Do, D.T. An Ensemble Tree-Based Model for Intrusion Detection in Industrial Internet of Things Networks. *Appl. Sci.* **2023**, *13*, 2479. [CrossRef]
27. Swift, D. A Practical Application of SIM/SEM/SIEM, Automating Threat Identification. 2006. Available online: https://www.sans.org/white-papers/1781/ (accessed on 30 January 2023).
28. Tian, Z.; Luo, C.; Lu, H.; Su, S.; Sun, Y.; Zhang, M. User and Entity Behavior Analysis under Urban Big Data. *ACM/IMS Trans. Data Sci.* **2020**, *1*, 1–19. [CrossRef]
29. Podzins, O.; Romanovs, A. Why SIEM is Irreplaceable in a Secure IT Environment? In Proceedings of the 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 25 April 2019; pp. 1–5. [CrossRef]
30. Shea, S. SOAR (Security Orchestration, Automation and Response). 2019. Available online: https://www.techtarget.com/searchsecurity/definition/SOAR (accessed on 30 January 2023).

31. Gartner. Gartner: 2022 Market Guide for Security Orchestration, Automation and Response Solutions. 2022. Available online: https://swimlane.com/resources/gartner-soar-market-guide-2022 (accessed on 30 January 2023).
32. Johnson Kinyua, L.A. AI/ML in Security Orchestration, Automation and Response: Future Research Directions. *Intell. Autom. Soft Comput.* **2021**, *28*, 527–545. [CrossRef]
33. Gupta, N.; Traore, I.; de Quinan, P.M.F. Automated Event Prioritization for Security Operation Center using Deep Learning. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 5864–5872. . [CrossRef]
34. Tjhai, G.C.; Furnell, S.M.; Papadaki, M.; Clarke, N.L. A preliminary two-stage alarm correlation and filtering system using SOM neural network and K-means algorithm. *Comput. Secur.* **2010**, *29*, 712–723. [CrossRef]
35. Shittu, R.; Healing, A.; Ghanea-Hercock, R.; Bloomfield, R.; Muttukrishnan, R. OutMet: A new metric for prioritising intrusion alerts using correlation and outlier analysis. In Proceedings of the 39th Annual IEEE Conference on Local Computer Networks, Edmonton, AL, Canada, 8–11 September 2014; pp. 322–330.
36. Valeur, F.; Vigna, G.; Kruegel, C.; Kemmerer, R.A. Comprehensive approach to intrusion detection alert correlation. *IEEE Trans. Depend. Secur. Comput.* **2004**, *1*, 146–169. [CrossRef]
37. Hassan, W.U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; Bates, A. NODOZE: Combatting Threat Alert Fatigue with Automated Provenance Triage. In Proceedings of the Network and Distributed Systems Security (NDSS) Symposium, San Diego, CA, USA, 24–27 February 2019.
38. Sun, L.; Versteeg, S.; Boztas, S.; Rao, A. Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study. *arXiv* **2016**, arXiv:1609.06676.
39. Chakir, E.M.; Moughit, M.; Idrissi Khamlichi, Y. An efficient method for evaluating alerts of Intrusion Detection Systems. In Proceedings of the 2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), Fez, Morocco, 19–20 April 2017; pp. 1–6. [CrossRef]
40. Aminanto, M.E.; Zhu, L.; Ban, T.; Isawa, R.; Takahashi, T.; Inoue, D. Combating Threat-Alert Fatigue with Online Anomaly Detection Using Isolation Forest. In *Proceedings of the Lecture Notes in Computer Science, Neural Information Processing (ICONIP) 2019*; Springer: Cham, Switzerlan, 2019; pp. 756–765.
41. Aminanto, M.E.; Ban, T.; Isawa, R.; Takahashi, T.; Inoue, D. Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis. *IEEE Access* **2020**, **8**, 217977–217986. [CrossRef]
42. Madani, A.; Rezayi, S.; Gharaee, H. Log management comprehensive architecture in Security Operation Center (SOC). In Proceedings of the IEEE 2011 International Conference on Computational Aspects of Social Networks (CASoN), Salamanca, Spain, 19–21 October 2011; pp. 284–289.
43. IBM. Log Event Extended Format (LEEF). 2016. Available online: https://www.ibm.com/support/knowledgecenter/SS42VS_DSM/b_Leef_format_guide.pdf (accessed on 9 May 2019).
44. McAfee. McAfee Enterprise Security Manager 10.2.0 Product Guide (Unmanaged). 2017. Available online: https://docs.mcafee.com/bundle/enterprise-security-manager-10.2.0-product-guide-unmanaged/page/GUID-984F5DA6-8D84-4549-855B-C77D53CF96B9.html (accessed on 30 September 2020).
45. Debar, H.; Curry, D.; Feinstein, B. The intrusion detection message exchange format (IDMEF). 2007. Available online: https://www.ietf.org/rfc/rfc4765.txt (accessed on 30 January 2023).
46. MITRE. Common Event Expression—CEE, a Unified Event Language for Interoperability. Available online: http://makingsecuritymeasurable.mitre.org/docs/cee-intro-handout.pdf (accessed on 30 January 2023).
47. Azodi, A.; Jaeger, D.; Cheng, F.; Meinel, C. A new approach to building a multi-tier direct access knowledgebase for ids/siem systems. In Proceedings of the 2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing, Chengdu, China, 21–22 December 2013; pp. 118–123.
48. Sapegin, A.; Jaeger, D.; Azodi, A.; Gawron, M.; Cheng, F.; Meinel, C. Hierarchical object log format for normalisation of security events. In Proceedings of the IEEE 2013 9th International Conference on Information Assurance and Security (IAS), Gammarth, Tunisia, 4–6 December 2013; pp. 25–30.
49. Anderson, M.; Antenucci, D.; Bittorf, V.; Burgess, M.; Cafarella, M.; Kumar, A.; Niu, F.; Park, Y.; Ré, C.; Zhang, C. Brainwash: A data system for feature engineering. *Proc. CIDR* **2013**, *2013*, 1–4.
50. Khurana, U.; Turaga, D.; Samulowitz, H.; Parthasrathy, S. Cognito: Automated feature engineering for supervised learning. In Proceedings of the IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 1304–1307.
51. Li, D.; Kotani, D.; Okabe, Y. Improving Attack Detection Performance in NIDS Using GAN. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 817–825. [CrossRef]
52. Pezoa, F.; Reutter, J.L.; Suarez, F.; Ugarte, M.; Vrgoč, D. Foundations of JSON schema. In Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, Montreal, QC, Canada, 11–15 April 2016; pp. 263–273.
53. Joloudari, J.H.; Marefat, A.; Nematollahi, M.A.; Oyelere, S.S.; Hussain, S. Effective Class-Imbalance Learning Based on SMOTE and Convolutional Neural Networks. *Appl. Sci.* **2023**, *13*, 4006. [CrossRef]

54. Oliveira, N.; Praça, I.; Maia, E.; Sousa, O. Intelligent Cyber Attack Detection and Classification for Network-Based Intrusion Detection Systems. *Appl. Sci.* **2021**, *11*, 1674. [CrossRef]
55. Jadhav, A.; Mostafa, S.M.; Elmannai, H.; Karim, F. An Empirical Assessment of Performance of Data Balancing Techniques in Classification Task. *Appl. Sci.* **2022**, *12*, 3928. [CrossRef]
56. Liu, C.; Cao, L.; Yu, P.S. A hybrid coupled k-nearest neighbor algorithm on imbalance data. In Proceedings of the IEEE 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, China, 6–11 July 2014; pp. 2011–2018. [CrossRef]
57. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328.
58. Nguyen, H.M.; Cooper, E.W.; Kamei, K. Borderline over-sampling for imbalanced data classification. *Int. J. Knowl. Eng. Soft Data Paradig.* **2011**, *3*, 4–21. [CrossRef]
59. Ndichu, S.; Ban, T.; Takahashi, T.; Inoue, D. AI-Assisted Security Alert Data Analysis with Imbalanced Learning Methods. *Appl. Sci.* **2023**, *13*, 1977. [CrossRef]
60. Chang, C.C.; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 1–27. Available online: http://www.csie.ntu.edu.tw/~cjlin/libsvm (accessed on 30 January 2023). [CrossRef]
61. Ban, T.; Samuel, N.; Takahashi, T.; Inoue, D. Combat Security Alert Fatigue with AI-Assisted Techniques. In Proceedings of the 2021 Cyber Security Experimentation and Test Workshop, CSET, New York, NY, USA, 9 August 2021; pp. 9–16. [CrossRef]
62. Duda, R.O.; Hart, P.E.; Stork, D.G. *Pattern Classification*, 2nd ed.; Wiley-Interscience: New York, NY, USA, 2000.
63. Sammut, C.; Webb, G.I. (Eds.) Stratified Cross Validation. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2010; pp. 928–928. [CrossRef]
64. Tashman, L.J. Out-of-sample tests of forecasting accuracy: An analysis and review. *Int. J. Forecast.* **2000**, *16*, 437–450. CrossRef
65. Svetunkov, I.; Petropoulos, F. Old dog, new tricks: A modelling view of simple moving averages. *Int. J. Prod. Res.* **2018**, *56*, 6034–6047. [CrossRef]
66. He, H.; Garcia, E.A. Learning from Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284. [CrossRef]
67. Lapin, M.; Hein, M.; Schiele, B. Learning using privileged information: SVM+ and weighted SVM. *Neural Netw.* **2014**, *53*, 95–108. [CrossRef]
68. Cimpanu, C. Emotet, tOday'S Most Dangerous Botnet, Comes Back to Life. 2019. Available online: https://www.zdnet.com/article/emotet-todays-most-dangerous-botnet-comes-back-to-life/ (accessed on 30 January 2023).
69. Pathria, R.K.; Beale, P.D. *Statistical Mechanics*, 3rd ed.; Academic Press: Cambridge, MA, USA, 2011.
70. Ndichu, S.; Ban, T.; Takahashi, T.; Inoue, D. A Machine Learning Approach to Detection of Critical Alerts from Imbalanced Multi-Appliance Threat Alert Logs. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 2119–2127. [CrossRef]
71. Mathworks. Hyperparameter Optimization in Classification Learner App. 2023. Available online: https://www.mathworks.com/help/stats/hyperparameter-optimization-in-classification-learner-app.html (accessed on 30 January 2023).
72. Powers, D.M.W. Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation. *Int. J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
73. van der Maaten, L.; Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
74. Petrila, J.; Cohn, B.; Pritchett, W.; Stiles, P.; Stodden, V.V.; Humowiecki, M.; Rozario, N. Legal Issues for IDS Use: Finding a Way Forward. 2017. Available online: https://aisp.upenn.edu/wp-content/uploads/2016/07/Legal-Issues.pdf (accessed on 30 January 2023).
75. Quality, N.; Commission, S. NDIS Code of Conduct. 2019. Available online: https://www.ndiscommission.gov.au/about/ndis-code-conduct (accessed on 30 January 2023).