Combat Security Alert Fatigue with AI-Assisted Techniques

Tao Ban
National Institute of Information and Communications Technology
Tokyo, Japan
bantao@nict.go.jp

Ndichu Samuel
National Institute of Information and Communications Technology
Tokyo, Japan
ndichu@nict.go.jp

Takeshi Takahashi
National Institute of Information and Communications Technology
Tokyo, Japan
takeshi_takahashi@nict.go.jp

Daisuke Inoue
National Institute of Information and Communications Technology
Tokyo, Japan
dai@nict.go.jp

ABSTRACT

The main challenge for security information and event management (SIEM) is to find critical security incidents among a huge number of false alerts generated from separate security products. To address the alert fatigue problem that is common for security experts operating the SIEM, we propose a new alert screening scheme that leverages artificial intelligence (AI)-assisted tools to distinguish actual threats from false alarms without investigating every alert. The proposed scheme incorporates carefully chosen learning algorithms and newly designed visualization tools to facilitate speedy alert analysis and incident response. The proposed scheme is evaluated on an alert dataset collected in the security operation center of an enterprise. With a recall rate of 99.598% for highly critical alerts and a false positive rate of 0.001% reported, the proposed scheme demonstrated very promising potential for real world security operations. We believe the proposed scheme is effective in addressing the alert fatigue problem, and therefore paves the way for a consolidated security solution for network security at the enterprise level.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**.

KEYWORDS

Intrusion detection systems, security alert analysis, alert fatigue

ACM Reference Format:
Tao Ban, Ndichu Samuel, Takeshi Takahashi, and Daisuke Inoue. 2021. Combat Security Alert Fatigue with AI-Assisted Techniques. In *Cyber Security Experimentation and Test Workshop (CSET '21) , August 9, 2021, Virtual, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3474718.3474723

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CSET '21, August 9, 2021, Virtual, CA, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9065-1/21/08…$15.00
https://doi.org/10.1145/3474718.3474723

1 INTRODUCTION

Network-based intrusion detection systems (NIDSs) [24] are devices intelligently distributed within networks that monitor networks and systems for malicious activities or policy violations. Aiming to avoid missing crucial events, more and more enterprises are deploying multiple NIDSs and other security appliances in their networks and leverage security information and event management (SIEM) systems [6] to combine outputs from multiple sources for a consolidated security solution. On the other hand, the growing number of integrated security appliances in SIEM have given rise to the so-called *alert fatigue problem*: security operators becomes desensitized to the alarms (alerts) after constant exposure to a large number of frequent security alerts. Desensitization then leads to longer security incident response times and missing important alarms, which result in increased network security risks and a lower level of quality of services. According to a survey conducted by the Cloud Security Alliance, half of enterprises have six or more tools that generate security alerts. Among IT security professionals, 31.9% of analysts do not pay attention to alerts anymore due to the high frequency of false positives [20].

Alert fatigue is a serious barrier to implement a consolidated security solution by integrating multiple security appliances. While it may be difficult for a human operator to distinguish actual threats from false alarms without investigating every alert, advances in data science are making it possible for artificial intelligence (AI)-assisted tools to perform this task with high rates of accuracy. In this paper, aiming at the development of an automated security solution that can support security operations and shorten the time on crucial alert detection, investigation, and remediation, we propose a new framework to address alert fatigue by automating the alert prioritization process using AI and data visualization techniques. As shown in Figure 1, the proposed system framework consists of four modules: The alert generation module aggregates alert data from multiple NIDS sources. The feature processing module unifies logs in different formats into a transparent format and encodes the related information as numerical vectors. The machine learning module applies supervised learning to the labeled numerical vectors to obtain a prediction model. The investigation module carries out a performance evaluation for the prediction models. Moreover, it also incorporates data visualization and alert grouping techniques

to facilitate fast incident investigation for alerts predicted as highly critical.

The rest of this paper is organized as follows: Section 2 details the proposed system framework for alert screening. Section 3 presents the experimental results of the framework on a real enterprise network. Section 4 presents the related works. Section 5 concludes the paper.

2 PROPOSED SYSTEM FRAMEWORK

In this section, we discuss the four component modules comprising the proposed system framework.

2.1 Alert generation module

While the integration of multiple security appliances in SIEM might be a causal factor for the alert fatigue problem, a careful investigation of the 133.77M line log files from six IDS in the studied data reveals that different NIDSs are working complementarily to detect different type of anomalies in the network. Even for alerts caused by an identical attack campaign, an alert from another NIDS might provide complementary information about the attack using different detection mechanisms. Therefore, we believe as long as the alert fatigue problem can be solved, integrating multiple security appliances in SIEM can improve the security condition of the monitored network.

In the proposed system framework, multiple security appliances are deployed in the enterprise network to implement overall protection from cyber-threats. As security appliances may be subject to performance loss at an intensive packet rate [17], we recommend an independent parallel deployment provided that maintaining multiple security appliances is cost effective.

2.2 Feature processing module

A myriad of security alert log formats streaming from different devices provide complex integration for security event management. Despite the efforts of security vendors in adopting the common event format (CEF) [21] introduced by ArcSight, Inc., many NIDSs still use appropriate log formats. By parsing the alert messages and formatting them into standard JavaScript object notation (JSON) objects [15], the proposed data unification component merges log files from different vendors into one, single central view. The unified data provides a more complete and accurate picture of the security situation in the enterprise network, and make the data transparent and transportable.

The JSON objects, as the output of the data unification component, contain the most relevant event information including the description of the event, unique identifier per event-type, Internet protocol (IP) addresses involved in the communication, impact of the event, uniform resource locator (URL) to a risky Internet resource, hash values of a downloaded file, etc. These attributes are further divided into three types: numerical, categorical, and signature. We carefully select the most relevant attributes from the list that are both descriptive and generalizable as the representing features for the event. Each alert message in the log file is formulated as a numerical vector where one-hot encoding [16] is adopted to formulate categorical features into binary attributes.

In the data, highly critical alerts are first manually labeled by security operators in the security operation center (SOC) with deterministic evidence collected. To remove the effect of misjudgment and overlooking of critical alerts because of the alert fatigue problem in SOC operations, the labeled data is reviewed by AI experts who leverage information visualization tools to justify the label information of the data.

2.3 Machine learning module

The machine learning module applies machine learning algorithms to distinguish the highly critical alerts (labeled as positive in the data) from the rest of the corpus (labeled as negative in the data).

Alert prioritization shares the feature of common security problems, known as a severe class imbalance, between positive and negative classes: in our data, only 2,239 samples were labeled as positive by security operators. Such a severe disparity in the frequencies of the observed classes can have a significant negative impact on model fitting. To cope with the skewness in the data, we adopt a cost-sensitive approach known as a weighted support vector machine (WSVM) [9]. By assigning different weights to positive and negative instances, WSVM weighs the margin proportional to the class importance, which makes it more effective on imbalanced datasets.

Another technique for resolving the class imbalance is to subsample the training data in a manner that mitigates the issues. Examples of sampling methods for this purpose include down-sampling, up-sampling, and hybrid methods. In down-sampling, a subset of majority classes in the training set are randomly taken so that their class frequencies match the least prevalent class. In up-sampling, data in the minority class are randomly sampled (with replacements) until the frequencies of the minority classes become the same as those of the majority class. In hybrid methods, techniques such as SMOTE and ROSE down-sample the majority class and synthesize new data points in the minority class. Our data contains a majority number of categorical attributes, which renders sub-sampling based methods ineffective [18]. Plans to implement a sub-sampling method that generalizes categorical data have been set as future work.

Note that the number of positive samples in an alert dataset tends to be very limited. To build a robust classifier and evaluate its generalization performance without the luxury of abundant positive data, we adopt the model validation technique known as cross-validation. One round of cross-validation consists of partitioning a sample of data into complementary subsets, training the prediction model on one subset (i.e., the training set), and validating the model on the rest of the data (i.e., the validation set). To improve stability, multiple rounds of cross-validation are performed using different partitions, with the validation results averaged over the rounds to give an estimate of the model's predictive performance.

2.3.1 Classification methods. In this section, we select six classification methods [12] that can handle high-dimensional sparse data efficiently, i.e., K nearest neighbors (KNN), naive Bayes (NB), linear discriminant analysis (LDA), decision tree (DT), AdaBoost, SVM, and WSVM; an SVM that works better on skewed datasets. We apply the classifiers to build prediction models on the alert dataset introduced in Section 3.

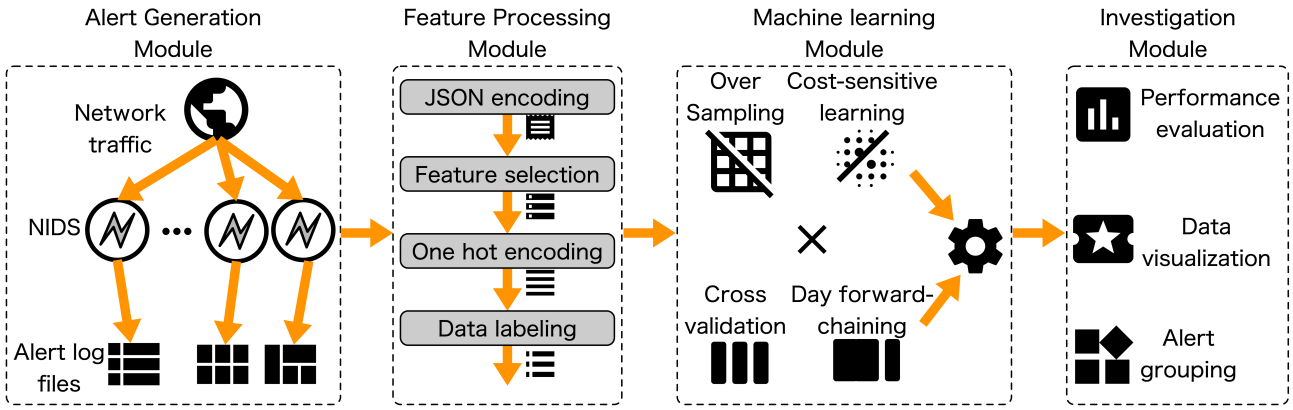

Figure 1: Proposal system framework for alert screening.

*K*NN is a commonly-used classification algorithm—a sample is classified by a majority vote of its K nearest neighbors. The results of *K*NN well indicate how much discriminant information can be captured in the similarity function defined on the feature vectors. As generalization is deferred until a query sample is known, the computational cost of *K*NN in prediction (which depends mostly on a nearest neighbor search for the query sample) tends to be intensive even for datasets with moderate samples size.

NB is a classifier based on Bayes' theorem. It assumes that all the features are statistically independent—the value of a particular feature is independent of the value of any other feature. While requiring a number of parameters linear in the number of features, NB classifiers are highly scalable, which well meets the requirements of our learning task.

LDA is a generalization of Fisher's linear discriminant to find a linear combination of features that characterizes two or more classes of samples. The resulting combination is used to build a linear classifier to separate samples from different classes. LDA is generally used for linearly separable tasks. As the experiment result shown, the features extracted from the alert messages satisfy LDA's linearly separable assumption approximately.

A DT is a tree-like representation for classifying examples. Each internal node of the tree is labeled with an input feature. For predicting the class label for a record (an alert message in our case), we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node. The process is repeated until a leaf node that is associated with a probability distribution of predicted classes is reached. Then, the class with the highest probability is taken as the output of the prediction. The decision mechanism of DT well meets the categorical nature of the alert data. Therefore, its result provides a baseline of prediction methods based on simple rules.

AdaBoost, short for Adaptive Boosting, is a statistical classification meta-algorithm often used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms, so called 'weak learners', is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are adjusted in favor of instances misclassified by previous learners.

An SVM is a supervised learning model that determines a hyperplane with a maximized margin to distinguish samples from two classes. SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or the Vapnik-Cervonenkis theory [29].

When it comes to a classification task with a high class imbalance, many SVM implementations address this by assigning different weights to positive and negative instances. They essentially weigh the samples so that the sum of the weights for the positives will be equal to that of the negatives. We adopt the SVM and WSVM implementation in the LIBSVM toolbox [9]. Referenced classification methods other than SVM and WSVM are based on the implementations in Matlab's *Statistics and Machine Learning toolbox*.

2.4 Investigation module

Depending on the nature of attack detection, many security appliances tend to issue multiple alerts for a lasting attack, e.g., a bot-infected host that constantly probes multiple devices in the network could raise many alerts. While a single line in the log file for most NIDSs contains related information for a particular rule violation triggered by some unsolicited network communication, not all NIDSs provide an effective mechanism to group multiple alerts associated with the same attack campaign. This could significantly increase the number of unnecessary alerts. Other factors that lead to the increase of superfluous false alerts include inappropriate detection rule settings and periodically issued alerts for lasting unsolicited communications.

We can leverage data visualization techniques to reduce the complexity in investigating these alerts and enable a fast incident response. In this paper, we introduce tile graphs for visualizing massive correlated alerts. As shown in Figure 2(a), the plot area of a tile is divided into two sub-planes: the source panel on the left and the destination panel on the right. On the source panel, the x axis represents the time stamp of the triggered alert, and the y axis represents the source IP addresses involved in the communication. On the destination panel, the x and y axes represent the destination

IP addresses and destination ports involved in the communications, respectively. To make the IP addresses discernible from each other, they are uniformly placed on the axes based on the indices in the order of their occurrence. An alert is represented by a line connecting a source IP address to a destination IP address. The color of the line is determined by the status of the source and destination IP addresses defined in Figure 2(b). For example, an alert that is triggered by an outbound communication from an internal host to an external host on the Internet is drawn as a blue line in the tile.

In Figure 2(a), we draw the tile graph for "HTTP login brute-force detected" type of attack detected during a period of one week. A large portion of these alerts are triggered by communications between internal hosts and the proxy server as shown by the blue lines toward the proxy server highlighted by circle A. While a security operator may not be so interested in such occasional erroneous communications, he/she might be more interested in investigating the cases where the attack intention is clearer, e.g., intensive attack attempts that show a periodic nature as highlighted by circles B and C. In this way, tile graphs can provide an insightful look into the complexity of cyber-attack campaigns and help to release the security operators from tedious work of log parsing and enable them put more focus on highly critical incidents.

3 EXPERIMENTS

In this section, we report the experimental results on applying the proposed scheme to NIDS logs collected from SOC operations in an enterprise network.

3.1 Data preparation

The alert logs used in this study were generated by six NIDSs, hereafter referred to as appliances A to F for anonymity purposes, deployed in a class-B network connecting more than 1,000 users and 30,000 network devices, including personal computers, servers, and mobile devices.

The traffic collected at multiple access points to the network are directed to the NIDS to generate security alerts for SOC operations. Alerts logs recorded over ten months, from January 1 to October 31, 2017, are used in the numerical study to verify the proposed scheme. In accordance with the standard process of alert management, the alerts were carefully investigated by security experts to identify key incidents that are labeled as highly critical alerts (positive samples). The rest of the alerts are taken as less critical alerts (negative samples). During the investigation of an alert message, the security experts examined the contents of communications associated with the alerts captured in archived packet capture (PCAP) files. Then, the label of the alert was determined on the basis of evidence gathered after examining the contents of accessed URLs, matching them with commercial URL black lists, performing the sandbox analysis of downloaded files, etc.

Table 1 shows the statistics of the dataset. Depending on the detection mechanisms implemented by the appliances, the number of alerts varies significantly from a few dozen to a few hundred thousand. Among the 133.77M alerts, only 593 were confirmed by security experts as positive and the rest were taken as negative. It is worth noting that as gathering deterministic evidence for the highly critical alerts takes much effort, there could be many crucial alerts overlooked by the security operators due to alert fatigue. Therefore, after the SOC operation, we carried out a review process that made use of tiles visualization and data grouping techniques on the basis of the signature features including URL and hash values of downloaded files to identify alerts identical to positive alerts. To speed up the evaluation process, we perform a down-sampling on the negative classes in our experiment so that approximately 0.488% of negative samples are picked up in the evaluation dataset. We keep all the samples from the positive class in the evaluation dataset. Identifying alerts that were somehow overlooked during the SOC operation in the down-sampled data resulted in a significant boost in the number of positive samples from 593 to 2,239. Finally, the data used for numerical evaluation contains 672K negative samples and 2,239 positive samples. The ratio of positive samples in the evaluation set is about 0.333%, which renders the classification task still a class-imbalanced one.

Note that on the basis of the reviewed alert label, we can roughly estimate the performance of all security appliances. In regard to detection precision, appliance D provided the highest detection precision of 72.868%, while other appliances achieved a detection precision as low as 0.014%.

3.2 Visualize data distribution using t-SNE

To better understand the data distribution, we adopt t-SNE [28] to visualize the data in a 2-D embedding space. t-SNE is a nonlinear dimension reduction method that can be used to approximate the relationship of high-dimensional data in a low-dimensional layout. It is computationally efficient and can handle complicated data distributions in high-dimensional spaces.

Figure 3 shows the 2-D visualization result of the unique samples in the dataset. In the figure, colored disks represents a unique sample and the size of a disk is in proportion to the number of alerts that share the same numerical representation. The positive alerts are emphasized by an enclosing black circle around the disks. The figure illustrates that although most of these appliances follow the CEF convention, alerts raised by different appliances tend to be spread out while those from the same appliances tend to cluster together. For a better understanding of the cluster formations in the layout, we further investigated the alerts from appliance A, which are shown as red disks. Loose clusters are first formed by alerts that belong to the same genre such as HTTP communications because they share common features such as TCP-protocol-based communication, and then tighter clusters are formed from samples that belong to the "suspicious malware downloading" category. Under the same category, information such as detection rules, download file types, and alert severity leads to minor differences in the alerts, giving rise to more compact clusters. The most important observation from the figure is that the positive alerts—enclosed by black circles—form tight clusters and are highly separable from the negative samples. This good separability between positive and negative alerts implies good prediction performance for malware detection.

3.3 Evaluation Metrics

To measure the generalization performance of the chosen supervised learning algorithms, we used four widely adopted metrics for supervised learning: accuracy, precision, recall, and false positive

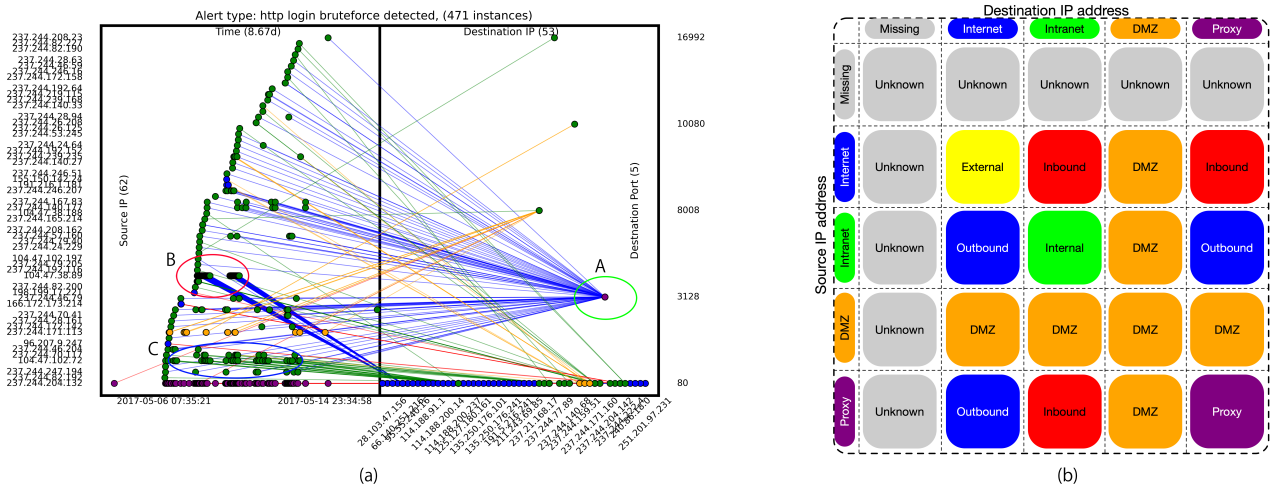

Figure 2: Visualization of "HTTP login bruteforce detected" type of attacks using a tile. (a) A tile graph, (b) the color code book of lines (attacks) and points (hosts) in a tile. Note: the IP addresses in this example are anonymized using random shuffling and do not have particular information.

Table 1: Statistics of experiment data

Appliance	#Total	#Daily Average	#Subsampled	Sampling Ratio (%)	#Positive	Positive Ratio (%)
A	36.68M	39,005	226,784	0.618	1286	0.567
B	11.86M	39,005	30,837	0.260	169	0.548
C	15.80M	51,953	36,145	0.229	20	0.055
D	0.56M	1,857	5,944	1.053	617	10.380
E	66.21M	217,804	372,161	0.562	53	0.014
F	6,498	21	129	1.985	94	72.868
Total	137.77M	452,958	672,000	0.488	2,239	0.333

rate (FPR). Moreover, as the alert screening has highly skewed class distribution, we also adopt F-score, true negative rate (TNR), and balanced classification rate (BCR) in the evaluation. The definitions of these metrics are related to the following intermediate measures.

- True Positive (TP): the number of records correctly predicted as positive by a classifier.
- False Positive (FP): the number of records incorrectly predicted as positive by a classifier.
- True Negative (TN): the number of records correctly predicted as negative by a classifier.
- False Negative (FN): the number of records incorrectly predicted as negative by a classifier.

The accuracy is defined as the percentage of the test label data that are correctly identified by a classifier, i.e.,

$$Accuracy = \frac{TP + TN}{n}. \tag{1}$$

where n represents the size of the test set.

The precision is the probability that predicted positive records are correctly classified, i.e.,

$$Precision = \frac{TP}{TP + FP}. \tag{2}$$

The recall is the probability that a record from the positive class is correctly classified, i.e.,

$$Recall = \frac{TP}{TP + FN}. \tag{3}$$

The FPR is the probability that a negative record is incorrectly predicted to be a positive record by a classifier, i.e.,

$$FPR = \frac{FP}{FP + TN}. \tag{4}$$

The F-score is the harmonic mean of the precision and recall and can evaluate the balance of the two, i.e.,

$$F - score = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}. \tag{5}$$

The TNR is the opposite of FPR; it is defined as the number of negative samples classified correctly divided by the total number of negative samples, i.e.,

$$TNR = \frac{TN}{FP + TN}. \tag{6}$$

The BCR is the average of recall and on positive class TNR, i.e.,

$$BCR = \frac{Recall + TNR}{2}. \tag{7}$$

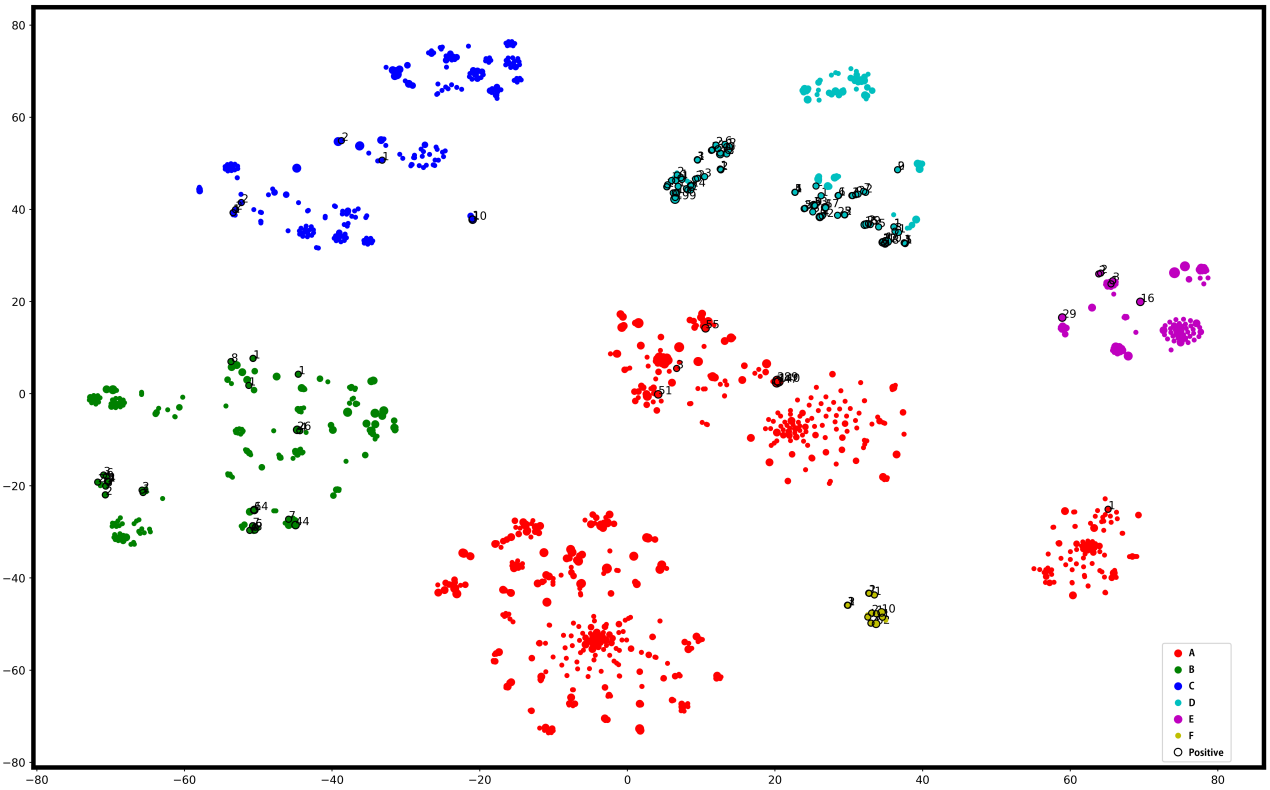

Figure 3: Visualization of unique alerts using t-SNE.

1

3.4 Numerical results

We compared the aforementioned classifier's performance on the prepared alert dataset. Table 2 shows the criteria introduced in the previous section are used for the evaluation. In the algorithm list, isolation forest (IF) [5] and IF embedded with day-forward chaining (IF+DC) [1] are unsupervised learning methods and the rest are supervised learning methods. As the implementation of IF and IF+DC may not fit to our dataset, we cite the results from [5] and [1], which consist of alerts collected from a single appliance for reference. Their numerical values are not directly comparable with the classification methods.

As can be seen from the table, due to the highly skewed class distribution, commonly-used classifiers do not provide a very high generalization performance. The best three algorithms are WSVM, SVM, and AdaBoost. While AdaBoost and LDA reported the highest recall rate of 99.732%, they showed slightly higher FPsR of 0.016% and 0.002%, respectively. SVM yielded the lowest FPR of 0.001% and a comparably high recall rate of 97.901%. By incorporating weight parameters to cope with the imbalanced class distribution, WSVM outperformed SVM on all of the seven criteria. In particular, it yielded a 3rd place recall rate of 99.598% and the lowest FPR of 0.001%. The extremely low FPR yielded by and SVM and WSVM indicates that the number of false alerts can be suppressed to a very low level, which can ensure a high potential for speedy SOC operation. Both SVM and WSVM yielded recall rates higher than 97.9%, which indicates that very few critical alerts will be missed by the detection engine. This is also an advantage of the proposed scheme.

Comparing the supervised learning methods with unsupervised learning ones, i.e., IF and IF+DC, we can see that unsupervised learning tends to have much higher FPRs, which negatively affects the effectiveness of the SOC operation. This is partially because unsupervised learning such as IF does not make use of the labeling information in the data; therefore, it might be suitable to detect alerts corresponding to extremely rare patterns. However, for critical alerts that do not show strong discriminant characters from common alerts, detection misses occur. The numerical results suggest that due to the complexity of the different type of alerts raised by different security appliances, supervised learning might be a more preferable technique for alert screening.

4 RELATED WORK

Among the many studies on alert screening, several focused on creating a common format of alert logs. For example, Madani et al. [19] discussed the challenges of categorizing, parsing, transferring, and filtering logs in various formats. Existing security appliances generate logs in different formats, including the log event extended format (LEEF) [14], CEF [21], intrusion detection message exchange format (IDMEF) [10], and common event expression (CEE) [22]. The absence of a common format makes the analysis of threat

Table 2: Performance comparison of supervised and unsupervised learning.

Algorithms	Accuracy (%)	Recall (%)	Precision (%)	FPR (%)	TNR (%)	F-score (%)	BCR (%)
IF	90.705	100.000	0.551	9.300	90.700	1.097	95.350
IF+DC	94.141	95.876	0.837	5.860	94.140	1.659	95.008
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
WSVM	**99.998**	99.598	**99.687**	**0.001**	**99.999**	**99.643**	99.799

alerts a challenging task. Azodi et al. [7] [8] proposed a model for reading and normalizing various log formats using named-group regular expressions (NGRE) and a knowledge base. Sapegin et al. [23] proposed a new common format that contains all information needed from different log formats.

Valeur et al. [27] proposed a correlation-based method for reducing the number of alerts. The performance of this method depends on the characteristics of datasets. In particular, a reduction of 99.2% could be achieved for the honeypot dataset but only 53.0% for the MIT/LL 2000 dataset. Despite the convincing performance, a number of correlation components may be impractical, which requires the security operator to have access rights to the host.

Hassan et al. [13] proposed the NoDoze method that uses an automated provenance triage to overcome the alert fatigue problem. NoDoze adjusts the suspiciousness level of each event in the provenance graph on the basis of that of neighboring events in the graph. The method performs behavioral execution partitioning by distinguishing between benign and malicious behaviors. It also generates the dependency graph of true alerts (most malicious) to prevent the dependency explosion due to previous data provenance.

Sun et al. [25] used IF to identify deviations from normal employee behaviors. They considered the temporal factor, where they gathered data for a period of time and then performed anomaly detection on the entire dataset. In contrast, Ding and Fei [11] used IF to analyze data coming in a form of streams using a sliding window. Tuor et al. [26] used a recurrent neural network to reduce the number of alerts regarding employee behaviors.

Aminanto et al. used a stacked autoencoder (SAE) to output transformed features that helped improve the detection rate [3] and performance of the *k*-means clustering algorithm [4]. The method proposed in the paper uses an SAE to reduce the number of false positives by calculating the reconstruction error values. In [2], Aminanto et al. addressed the two main drawbacks of IF by leveraging an SAE. In particular, a one-class SAE is applied to reduce the number of false positives by calculating reconstruction errors. They also used a day-forward-chaining analysis to conduct a time-series analysis in real time while considering the underlying patterns of previous data.

5 CONCLUSION

This paper presented a new system framework addressing the alert fatigue problem. The proposed system framework leverage machine learning and data visualization techniques to analyze alert logs generated from multiple security appliances and distill alerts that are highly critical. The scheme was demonstrated to be effective in reducing the number of false positives (as low as a FPR of 0.001%), and thus leaving only a small number of threat alerts for security investigation. Moreover, its high recall rate also guarantees that fewer critical alerts will be overlooked in the analysis. This study focused on an all-at-once alert analysis across multiple security appliances. In the future, the proposed method can be customized to specific appliances so that even better generalization performance can be expected.

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.

REFERENCES

[1] Muhamad Erza Aminanto, Tao Ban, Ryoichi Isawa, Takeshi Takahashi, and Daisuke Inoue. 2020. Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis. In *IEEE Access*. IEEE, 217977–217986.

[2] Muhamad Erza Aminanto, Tao Ban, Ryoichi Isawa, Takeshi Takahashi, and Daisuke Inoue. 2020. Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis. *IEEE Access* 8 (2020), 217977–217986. https://doi.org/10.1109/ACCESS.2020.3041837

[3] Muhamad Erza Aminanto, Rakyong Choi, Harry Chandra Tanuwidjaja, Paul D Yoo, and Kwangjo Kim. 2017. Deep abstraction and weighted feature selection for Wi-Fi impersonation detection. *IEEE Transactions on Information Forensics and Security* 13, 3 (2017), 621–636.

[4] Muhamad Erza Aminanto and Kwangjo Kim. 2017. Improving detection of Wi-Fi impersonation by fully unsupervised deep learning. In *International Workshop on Information Security Applications*. Springer, 212–223.

[5] Muhamad Erza Aminanto, Lei Zhu, Tao Ban, Ryoichi Isawa, Takeshi Takahashi, and Daisuke Inoue. 2019. Combating Threat-Alert Fatigue with Online Anomaly Detection Using Isolation Forest. In *Lecture Notes in Computer Science, Neural Information Processing (ICONIP) 2019*. Springer, Cham, 756–765.

[6] Igor Anastasov and Danco Davcev. 2014. SIEM implementation for global and distributed environments. In *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. IEEE, 1–6. https://doi.org/10.1109/WCCAIS.2014.6916651

[7] Amir Azodi, David Jaeger, Feng Cheng, and Christoph Meinel. 2013. A new approach to building a multi-tier direct access knowledgebase for ids/siem systems. In *2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 118–123.

[8] Amir Azodi, David Jaeger, Feng Cheng, and Christoph Meinel. 2013. Pushing the limits in event normalisation to improve attack detection in IDS/SIEM systems. In *2013 International Conference on Advanced Cloud and Big Data*. IEEE, 69–76.

[9] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (2011), 27:1–27:27. Issue 3. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[10] Herve Debar, David Curry, and Benjamin Feinstein. 2007. The intrusion detection message exchange format (IDMEF).

[11] Zhiguo Ding and Minrui Fei. 2013. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes* 46, 20 (2013), 12–17.

[12] Richard O. Duda, Peter E. Hart, and David G. Stork. 2000. *Pattern Classification (2nd Edition)* (2 ed.). Wiley-Interscience.

[13] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. 2019. NODOZE: Combatting Threat Alert Fatigue with Automated Provenance Triage. In *Network and Distributed Systems Security (NDSS) Symposium 2019* (San Diego).

[14] IBM. 2016. Log Event Extended Format (LEEF). https://www.ibm.com/support/knowledgecenter/SS42VS_DSM/b_Leef_format_guide.pdf. [Online; accessed 9th May 2019].

[15] ECMA International. 2013. The JSON Data Interchange Format. https://www.ecma-international.org/wp-content/uploads/ECMA-404_1st_edition_october_2013.pdf. [Online; accessed 30th January 2021].

[16] Eric Jackson and Rajeev Agrawal. 2019. Performance Evaluation of Different Feature Encoding Schemes on Cybersecurity Logs. In *2019 SoutheastCon*. IEEE, 1–9. https://doi.org/10.1109/SoutheastCon42311.2019.9020560

[17] Dongyang Li, Daisuke Kotani, and Yasuo Okabe. 2020. Improving Attack Detection Performance in NIDS Using GAN. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 817–825. https://doi.org/10.1109/COMPSAC48688.2020.0-162

[18] Chunming Liu, Longbing Cao, and Philip S Yu. 2014. A hybrid coupled k-nearest neighbor algorithm on imbalance data. In *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2011–2018. https://doi.org/10.1109/IJCNN.2014.6889798

[19] Afsaneh Madani, Saed Rezayi, and Hossein Gharaee. 2011. Log management comprehensive architecture in Security Operation Center (SOC). In *2011 International Conference on Computational Aspects of Social Networks (CASoN)*. IEEE, 284–289.

[20] McAfee. 2017. Alert Fatigue: 31.9% of IT Security Professionals Ignore Alerts. https://www.mcafee.com/blogs/enterprise/cloud-security/alert-fatigue-31-9-of-it-security-professionals-ignore-alerts/. [Online; accessed 30th January 2021].

[21] McAfee. 2017. McAfee Enterprise Security Manager 10.2.0 Product Guide (Unmanaged). https://docs.mcafee.com/bundle/enterprise-security-manager-10.2.0-product-guide-unmanaged/page/GUID-984F5DA6-8D84-4549-855B-C77D53CF96B9.html. [Online; accessed 30th September 2020].

[22] MITRE. [n.d.]. Common Event Expression — CEE, A Unified Event Language for Interoperability. http://makingsecuritymeasurable.mitre.org/docs/cee-intro-handout.pdf. [Online; accessed 9th May 2019].

[23] Andrey Sapegin, David Jaeger, Amir Azodi, Marian Gawron, Feng Cheng, and Christoph Meinel. 2013. Hierarchical object log format for normalisation of security events. In *2013 9th International Conference on Information Assurance and Security (IAS)*. IEEE, 25–30.

[24] Aumreesh Ku. Saxena, Sitesh Sinha, and Piyush Shukla. 2017. General study of intrusion detection system and survey of agent based intrusion detection system. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 417–421. https://doi.org/10.1109/CCAA.2017.8229866

[25] Li Sun, Steven Versteeg, Serdar Boztas, and Asha Rao. 2016. Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study. *CoRR* abs/1609.06676 (2016). arXiv:1609.06676 http://arxiv.org/abs/1609.06676

[26] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. 2017. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

[27] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A Kemmerer. 2004. Comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on dependable and secure computing* 1, 3 (2004), 146–169.

[28] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html

[29] Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.