electronics MDPI

Article

Unsupervised Security Threats Identification for Heterogeneous Events

Young In Jang [1], Seungoh Choi [2], Byung-Gil Min [2] and Young-June Choi [1,*]

[1] Department of Computer Engineering, Ajou University, 206, World Cup-ro, Yeongtong-gu, Suwon-si 16499, Republic of Korea; sungje@ajou.ac.kr
[2] The Affiliated Institute of ETRI, 1, Daejeon Yuseong P.O. Box, Yuseong-gu, Daejeon 34186, Republic of Korea; sochoi@nsr.re.kr (S.C.); bgmin@nsr.re.kr (B.-G.M.)
* Correspondence: choiyj@ajou.ac.kr

Abstract: As cyberattacks targeting industrial control systems continue to evolve, the development of sophisticated technologies to detect these security threats becomes increasingly essential. In addition, it is necessary to update adversarial information constantly. However, this process is complicated by the deployment of heterogeneous equipment, which increases the number of indicators and characteristics that must be analyzed by security administrators. Furthermore, security operation centers often struggle to respond promptly to adversaries because of the high number of false alerts caused by unreliable system labels. These challenges make it difficult to construct reliable detection systems. To address these issues, we propose a robust unsupervised threat-identification method. Our approach involves applying a preprocessing technique tailored to the various data types pertinent to alerts, followed by classifying unlabeled alerts using an autoencoder (AE) model. Despite the presence of numerous false positives, we verified that the proposed model could effectively distinguish between different attack types and identify their relationships with only one round of training in homogeneous and heterogeneous environments within industrial control systems. Moreover, our model can filter and display data classified as actual attacks and generate relational tables.

Keywords: anomaly detection; autoencoder; heterogeneous environment; unsupervised learning

check for updates

Citation: Jang, Y.I.; Choi, S.; Min, B.-G.; Choi, Y.-J. Unsupervised Security Threats Identification for Heterogeneous Events. *Electronics* **2024**, *13*, 4061. https://doi.org/10.3390/electronics13204061

Academic Editors: Li Pan, Ning Liu and Conghui Zheng

Received: 15 August 2024
Revised: 11 October 2024
Accepted: 14 October 2024
Published: 15 October 2024

1. Introduction

To protect assets and data from cyber threats, firewalls, intrusion detection systems (IDSs), and intrusion prevention systems (IPSs) play crucial roles in monitoring and blocking network entry points that could serve as attack paths. As cybersecurity threats grow increasingly complex and sophisticated, new types of security equipment are constantly being developed and the threat signatures within these systems require constant updates. Moreover, several studies report cyber threats that penetrate the operational technology (OT) area [1,2]. Accordingly, as uninterrupted operations are a priority in the industrial domain, security solutions are deployed to safeguard industrial control systems (ICSs) against targeted attacks by various security threats. For example, the National Institute of Standards and Technology (NIST) is preparing guidelines for expanding the domain from ICS to OT and applying the latest cyber security technologies (behavioral anomaly detection, artificial intelligence, machine learning, and zero trust) [3]. As the critical infrastructure is operated over a wide area, security devices differ between vendors for each operational site and alerts generated from each security device are analyzed individually. The bulk of traditional industrial information constitutes time-series numeric data, such as analog and digital signals. In contrast, security information consists of numeric and categorical data (predominantly text-based categorical data as time-series data).

Recently, there have been efforts toward the consolidation of the heterogeneous information generated from these various types, which can be achieved, for example, by the implementation of a security information and event management (SIEM); thus, the amount of data has dramatically increased. Particularly, the need to analyze security alerts raised by the security operation centers (SoCs) in heterogeneous security environments has increased. However, it is challenging for security administrators to analyze large amounts of data and provide timely responses because the data contain different types of alerts from numerous devices. Furthermore, as the number of intelligent attacks increases, unnecessary alerts, such as false alerts, overdetections, and nondetections, which occur owing to the performance of security equipment, increase the security administrator's workload. Among the various available methods, machine learning is the key to exploring new solutions to these problems. Machine learning specializes in processing big data, and its performance is guaranteed in detecting and classifying abnormalities provided that sufficiently large noiseless datasets are available.

Numerous datasets developed from testbeds or virtual simulation environments targeting ICSs are publicly available [4]. Nevertheless, the alert data produced when security equipment detects abnormal cases are insufficient, and most of the dataset targets are based on network packet data collected by specific equipment (such as CICFlowMeter and NetFlow. CICFlowMeter, formerly known as ISCXFlowMeter, is an open-source tool developed by Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick (UNB) in Canada. NetFlow was created by Cisco Systems in San Jose, CA, USA.). To overcome this problem, we obtained security alerts generated by various devices and adopted an unsupervised learning technique to resolve the issues of labeling and categorizing data in practice. Figure 1 presents an overview of the proposed framework. First, we collect heterogeneous alerts from devices and adapt the data for use as deep-learning input through data refinement. The results from the trained model are evaluated in three parts: anomaly detection, alert filtering, and integrated relevance analysis (IRA).

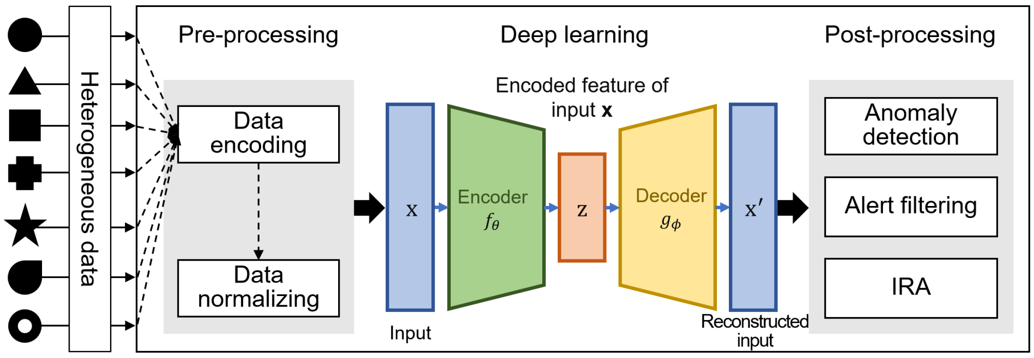


Figure 1. Overview of the proposed framework.

In summary, the contributions of this study are as follows:

- **Robust unsupervised detection model:** We propose a novel method to remove false alarms from the alerts for five types of attacks by converting multi-class classification with an anomaly detection task.
- **Application of optimal preprocessing:** We apply an appropriate preprocessing method while considering the characteristics of the features among the various methods.
- **Filtering alerts to be analyzed:** Using the proposed model, the data classified as true alerts can be filtered from heterogeneous alert data and displayed. The security administrator can respond by first analyzing the filtered data and then checking the excluded data if necessary.
- **Integrated relevance analysis:** We analyze correlations to identify attacks and classify attack types using IRA. This helps the security administrator to respond to occurring attacks.

The remainder of this paper is organized as follows: Section 2 presents the related work. Section 3 describes the methodologies used to implement our framework. In Section 4, we introduce the experimental environment and present the performance of anomaly detection using the proposed model and IRA. The results and scope for future research are discussed in Section 5. Section 6 concludes the paper.

2. Related Works

In this section, we review the datasets commonly used in research on network intrusion detection systems (NIDS), as well as state-of-the-art anomaly detection and classification techniques based on machine learning.

- **Datasets:** Representative datasets include KDD-CUP99 [5] and NSL-KDD. CTU-UNB mixes the botnet-based datasets, CTU-13 [6] and UNB-ISCX-IDS 2012 [7]. UNSW-NB15 [8] comprises network packet data collected by the IXIA PerfectStorm tool and includes data pertinent to nine types of attacks, namely, fuzzers, analysis, backdoors, denial of service (DoS), exploits, generic, reconnaissance, shellcode, and worms. USTC-TFC 2016 [9] contains traffic data comprising ten types of both benign and malware traffic. CSE-CIC-IDS 2018 [10] generated data on web, brute force, DoS, botnet, and distributed denial of service (DDoS)+PortScan attacks as B-profiles and M-Profiles. CIC-DDoS 2019 [11] and CIC-DoS 20177 [12] are network traffic data based on DDoS and DoS, respectively. UGR'16 [13] consists of data on real traffic and up-to-date attacks collected by NetFlow v9 collectors. Most of these datasets comprise network traffic data and not logs from security devices such as enterprise security management (ESM) or SIEM log datasets collected from heterogeneous environments. Additionally, they comprise single attack (for example, DDoS and botnet) data or the scale of the attacks is small; thus, these datasets are limited in terms of the variety and sizes of the attacks.
- **Anomaly classification based on machine learning:** Anomaly-based classification methods that detect intrusions through the identification and classification of attacks have been studied extensively [14–17]. These studies aimed to increase the detection rate and accuracy and lower the false alarm rate through the application of supervised learning methods. However, a disadvantage of supervised learning is that only labeled data can be inputted and learned. Most of the data used for training support labels; however, in practice, it is difficult to label the collected data because various traffic or logs can appear simultaneously, or an unseen attack to which a timely response is not possible, such as a zero-day attack, may occur.
- **Anomaly detection based on machine learning:** Studies on various supervised learning-based detection models include [7,9,18–22]. To solve the problem of labeled data, an alternative solution is unsupervised learning, which does not require labels. Unsupervised learning solutions [11,23–28] have implemented autoencoders, one-class K-means, one-class scaled convex hulls, and isolation forests to identify threats. Nguyen et al. [29] showed that the variational autoencoder can identify various attacks through reconstructed errors when compared with the autoencoder and Gaussian-based thresholding techniques. They adopted gradients as fingerprints to identify or classify the attack types. The gradients identify each attack and show similar results for scan 11 and scan 44 attacks; thus, using this result for reconstructed errors improves the performance. However, the data collected by NetFlow have limitations because only traffic data and five types of attacks are included the following: DoS, port scanning, botnet, spam, and blacklist. Rao et al. [30] proposed the bi-functional autoencoder (BFAE) to reduce the dimensionality of time-series data using autoencoders capturing nonlinear relationships. The application of the BFAE demonstrated superior dimensionality reduction results compared to traditional techniques such as Principal Component Analysis (PCA), AE, and Functional Principal Component Analysis (FPCA). The BFAE method focuses on reconstruction; thus, it is not relevant to our research.
- **Applications in different domains:** Studies have been conducted in other domains to detect outliers [31–33]. Zhang et al. [33] proposed an anomaly detection technique based on unsupervised learning and attention techniques for time-series data. Choi

and Kim [34] proposed anomaly detection using machine learning with a special focus on the HIL-based augmented ICS (HAI) dataset using an autoencoder combining Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks. The study involved two types of data: a synthetic dataset and a real-world power plant dataset. Because both time-series datasets were composed exclusively of numeric data, the direct application of the data as input to the deep learning model was not complicated. However, in heterogeneous security equipment alerts, numerical and categorical data are mixed. Moreover, even if security devices have the same features, some devices use numeric labeling, whereas others use categorical labeling. For example, device A indicates the level of the alert using numbers, whereas device B uses character values such as low, medium, and high. Therefore, we need to account for this heterogeneity by applying a method to convert categorical data to numeric data, after which we can use the various models of anomaly detection regardless of the domain.

3. Methodology

Our proposed framework consists of four parts: data generation, data preprocessing, anomaly detection, and IRA. First, we collected datasets from heterogeneous security devices under various attack scenarios. Second, alerts from heterogeneous security devices were sent in various numerical and categorical formats. However, these alerts could have different data formats or meanings, even when the same feature name is used, because of the security device's policy. Therefore, data transformation was required to normalize the data type and meaning. After data preprocessing, our model can identify abnormal alerts with an autoencoder, regardless of the attack types. Finally, we investigated the correlation among data features using the Pearson correlation coefficient and chi-squared test.

3.1. Dataset Generation

As mentioned in Section 2, most open datasets comprise network traffic data captured using NetFlow or Pcap. Generally, the security operation center builds its own security system with various security devices, and security administrators analyze the network state using information from these security devices. Therefore, we created a dataset using an HAI testbed to handle real-world situations.

3.1.1. Testbed Construction for Generating Dataset

To collect various types of security alerts, we leveraged a testbed comprising five types of commercial security equipment and two types of open-source network intrusion detection systems, as summarized in Table 1. We built an environment including this security equipment and monitoring systems using an elastic stack (https://www.elastic.co/elastic-stack/ accessed on 13 October 2024), shown in Figure 2 [35], and then collected data from this environment. To generate security information from seven different security machines, the network traffic of the control system that is the target of the monitoring exercise is required. We accomplished the surveillance of network traffic by mirroring the configuration at the network switch in each control system based on availability. We leveraged an aggregator capable of traffic aggregation, replication, and distribution to transmit the mirrored traffic from each network switch to each security machine. When the mirrored traffic from each network switch was delivered to the aggregator, it was aggregated and duplicated such that all security machines received identical inputs.

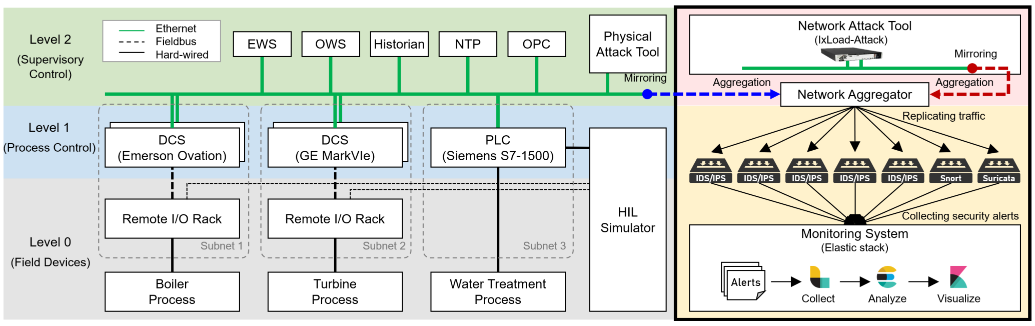


Figure 2. Structure of security data collection using an HAI testbed and security devices.

Table 1. Security alerts collected from seven machines.

ID	Vendor	Type	Installation	ACL	Signature
I	A	IDS/IPS	Physical	✓	✓
II	B	IDS/IPS	Physical	✓	✓
III	C	IDS/IPS	Physical	✓	✓
IV	C	Firewall	Physical	✓	
V	D	IDS/IPS	Physical	✓	✓
VI	Open-source	Snort	Virtual	✓	✓
VII	Open-source	Suricata	Virtual	✓	✓

We also installed Ixia's IxLoad-attack (https://www.keysight.com/us/en/products/network-test/protocol-load-test/ixload.html accessed on 13 October 2024), an attack-traffic-generating device, to reproduce the attack scenarios. The attacker and victim interface settings were configured to reflect the testbed environment. When attack traffic was directly injected into the network switch or the operation and management system in the testbed, the testbed component might be placed in a critical state. Therefore, to eliminate the impact of generating security alerts on the existing operating testbeds, the testbeds and attack-traffic-generating network were physically separated, as shown in Figure 2. Although it looks individually segmented, normal traffic and attack traffic from both networks, as with network traffic generated from one network through the amplifier device, can be integrated and delivered to security equipment.

3.1.2. Configuration of Attack Scenario

It is nearly impossible to account for all the attacks that could potentially target the assets (i.e., IoT systems, workstations, and data centers) used in the infrastructure critical to a diverse range of domains. Generally, an attacker identifies vulnerabilities applicable to a specific ICS and develops an attack tool to compromise its target. Therefore, we identified the functions and operational status of each control system within the testbed and constructed realistic scenarios, including applicable attacks. For the sequence of attacks, we assumed that a service-oriented attack was first conducted. If no service corresponds to a vulnerability or if the attack fails, the attacker finally executes an availability attack. To create such attacks, we composed applicable attacks on the testbed operated by the function and operation state of each control system, as summarized in Table 2. In the case of a service-specific attack, the target and method of the attack were determined. Particularly, we selected 13 DDoS attacks by which an attacker could perform a specific attack on the testbed.

Table 2. Summary of vulnerability attacks.

Category	Service	Attack Type	Attack Duration
Web	HTTP	152	25 min 14 s
OS	SMB	81	13 min 29 s
Remote procedure	DCERPC	65	10 min 55 s
Database	MySQL	18	2 min 21 s
DDoS	Specific protocols	13	2 h 14 min 53 s

3.1.3. Summary of Dataset

Our data contained logs from a testbed with normal and abnormal situations captured over 72 h. To prepare a dataset, we refined the data slightly by excluding alerts with the IP addresses, which cannot appear as attackers and victims in our testbed. Figure 3 illustrates the timeline of the dataset. To avoid the potential interference before an attack attempt, we divided the dataset into training and test datasets 10 min before the attack occurred. Moreover, we accounted for the delay involved in the transmission of the logs by the individual machines by collecting all security alerts for an additional margin of 7 min from the end of the DDoS attack. Table 3 lists the number of alerts per attack in the test, and the total count for each machine in the collected dataset is listed in Table 4 datasets.

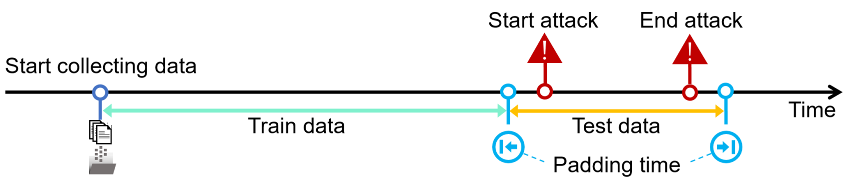


Figure 3. Timeline of dataset.

Table 3. Number of alerts according to attack class by machine in test datasets.

Class	Machine						
	I	II	III	IV	V	Snort	Suricata
Normal	6394 (39.4%)	11,944 (26.0%)	1901 (5.3%)	1448 (23.8%)	27 (0.4%)	2649 (3.8%)	546 (1.5%)
HTTP	4279 (26.4%)	16,565 (36.1%)	8891 (24.7%)	3422 (56.3%)	4579 (67.95%)	51,287 (74.3%)	2074 (5.9%)
MySQL	2222 (13.7%)	3313 (7.2%)	438 (1.2%)	328 (5.4%)	-	589 (0.9%)	172 (0.5%)
SMB	1252 (7.7%)	2435 (5.3%)	861 (2.4%)	284 (4.7%)	3 (0.04%)	401 (0.6%)	98 (0.3%)
DCERPC	587 (3.6%)	1245 (2.7%)	676 (1.9%)	366 (6.0%)	-	735 (1.1%)	248 (0.7%)
DDoS	1476 (9.1%)	10,360 (22.6%)	23,161 (64.5%)	232 (3.8%)	2130 (31.61%)	13,333 (19.3%)	32,286 (91.1%)

Table 4. Dataset summary.

Machine	Train Data	Test Data	Subtotal
I	537,155 (96.5%)	19,421 (3.5%)	556,576 (100%)
II	1,012,399 (95%)	52,777 (5%)	1,065,176 (100%)
III	2,281,683 (91%)	226,882 (9%)	2,508,565 (100%)
IV	1,488,850 (90%)	165,969 (10%)	1,654,819 (100%)
V	4872 (41.4%)	6887 (58.6%)	11,759 (100%)
Snort	456,847 (85.2%)	79,304 (14.8%)	536,151 (100%)
Suricata	33,063 (48.2%)	35,507 (51.8%)	68,570 (100%)
Total	5,814,869 (90.8%)	586,747 (9.2%)	6,401,616 (100%)

False alerts can occur in both the normal and attack intervals. This is caused by differences in the performance of the security equipment, which frequently occur in an actual operating environment. In order to consider real-world scenarios where false alerts are frequent, we did not eliminate the false alerts from the normal interval in this study. Because the false alerts in the training set act as noise and help the model not to overfit the normal data, our model can eliminate the false alerts that appeared in the test dataset. Furthermore, we demonstrated that distinctly different alarms exist because the difference between the loss of true and false alarms was large.

3.2. Data Preprocessing

Security alerts contain a combination of numerical and categorical data. However, the autoencoder uses numerical data as its input; thus, the conversion of categorical data into numeric data is required. We adopted two methods to address this problem: labeling and one-hot encoding.

As illustrated in Figure 4, we used the labeling method for data whose meaning varied depending on the order and size. However, we applied one-hot encoding for data whose meaning is the value itself, such as IP addresses and protocols. Table 5 shows that the number of fields in the collected raw data ranges from a minimum of 20 to a maximum of 69 per machine, including fields from the elastic stack. First, we removed fields added by the elastic stack and fields that were deemed meaningless for each machine. The remaining fields were classified as numerical or categorical based on their type and meaning. For categorical data, we determined whether they were ordinal or nominal to decide whether the appropriate encoding method would be label encoding or one-hot encoding. After the encoding process, the number of fields increased to a maximum of 1122, as shown in Table 5, and the data preprocessing was completed through a normalization process. The next paragraph provides a detailed explanation of the data preprocessing for Snort, an open-source IDS.

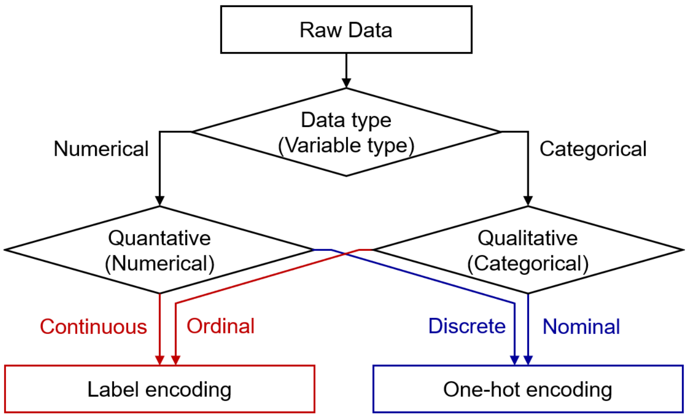


Figure 4. Preprocessing according to data type.

Table 5. Number of fields in data by machine.

Category	**Machine**						
	I	**II**	**III**	**IV**	**V**	**Snort**	**Suricata**
Raw data	20	48	52	38	30	21	69
Encoded data	63	179	99	38	75	144	1122

An example of the data collected from Snort is presented in Table 6. The total number of fields was 21, and the field names beginning with "@" were derived from an elastic stack. The original data from Snort were in the form of raw alerts, and each field was converted from raw alerts into individual features. Further, *timeSent* as a time field, was meaningless because the collected alerts were stacked in sequence. Thus, *@version*, *@timeStamp*, *original*, and *timeSent* were eliminated from the list of relevant fields, leaving 17 fields. Among the remaining features, both numerical and character types existed. Most character-type features were categorical values, whereas some numeric features also had categorical meanings. We considered the meaning of the field and classified numerical data such as *levelRisk*, *portVictim*, *portAttacker*, *id_generator*, and *id_revision*, whereas the other features were considered categorical. As an example of the increased overhead with one-hot encoding, when the value count of *portAttacker* is 13,862, the feature count increases from 1 to 13,862 features when *portAttacker* is included.

The growing feature size increases the operational time and leads to a memory overflow. The number of ports set by the protocol was a numeric value between 0 and 65,535. *LevelRisk* was categorical data consisting of numbers that were processed as numeric data because each number could be classified into the 0-low, 1-medium, 2-high, or 3-severe categories. The protocol feature was categorical and had four values: "udp", "tcp", "icmp", and "igmp" without priority; thus, one-hot encoding was applied. The protocol features increased to four in total: *protocol_udp*, *protocol_tcp*, *protocol_icmp*, and *protocol_igmp*. The features could only have a value of zero or one, depending on the existence or absence of the feature value. If the four features were listed in the order [*protocol_udp*, *protocol_tcp*, *protocol_icmp*, *protocol_igmp*], then one of these is selected in the form of [1,0,0,0], [0,1,0,0], [0,0,1,0], and [0,0,0,1] based on the values stored in the protocol features. For example, if the protocol value of an alert was "icmp", then [0,0,1,0] were allocated to each feature in order.

In a real-world situation, a new value for the IDS/firewall has not been observed before the generation of an alert during a cyber-attack; hence, normalizing both the training and testing datasets together is unsuitable. To manage this condition, we added an extra process to make it more practical as follows: (1) To maximize the case where values from the training dataset did not appear within the test dataset, only the training dataset was normalized. (2) The min–max value of each feature was stored and used for the normalization of the test dataset. (3) The test dataset was normalized to a value over the range of the training dataset. Further, as mentioned above, if a predefined range existed, as in *portAttacker* and *portVictim*, we reflected this in the normalization.

Table 6. Data type of fields in raw data from machine Snort. **E:** Excluded from preprocessing; **N:** Numerical data; **C:** Categorical data.

Field	Data Type	Unique Value	Instance
timeSent	E	82,626	'2020-08-21T07:45:54.000Z', '2020-08-21T07:46:04.000Z', ...
@version	E	1	1
@timestamp	E	98,972	'2020-08-21T07:45:55.295Z', '2020-08-21T07:46:05.295Z', '2020-08-21T07:46:05.296Z', ...
original	E	950,484	'08/21-16:45:54.143014 {[}**{]} {[}1:1234567892:0{]} home -\textgreater external {[}**{]}{[}Priority: 0{]} {UDP} a.b.c.d:### -\textgreater a.b.c.d:###'
levelRisk	N	4	0, 1, 2, 3
portVictim	N	890	138, 1947, 137, 67, 5355, 3991, ...
portAttacker	N	13,862	138, 49157, 56856, 49811, 137, 49155, 60130, 68, 51509, ...
id_generator	N	1	1
id_revision	N	21	0, 1, 18, 9, 17, 13, 10, 5, 7, 11, 8, 2, 6, 3, 15, 4, 12, 22, 16, 14, 25
id_signiture	N	95	1234567892, 1234567893, 41701, 1234567891, 2381, 40046, ...
IPAttacker	C	43	Mixed IPv4, IPv6
IPVictim	C	28	Mixed IPv4, IPv6
location.lat	C	1	latitude
location.lon	C	1	longitude
nameMachine	C	1	'Snort IPS System'
nameAttack	C	92	'home -\textgreater external', 'External -\textgreater external', 'SERVER-IIS cmd.exe access', ...
protocol	C	4	'udp', 'tcp', 'icmp', 'igmp'
nameOperator	C	1	'A'
nameUnit	C	1	'c'
type	C	1	'snort'
categoryModule	C	14	nan, 'Potential Corporate Privacy Violation', 'Attempted Administrator Privilege Gain', 'Web Application Attack', ...

3.3. Anomaly Detection

We focused on the characteristics of the autoencoder, such as data compression and data restoration without noise, to distinguish between normal and abnormal alerts owing to the problem described in Section 3.1.3. The encoder and decoder used in this study consisted of four fully connected layers. The feature size of the first layer output was 100, and as the layers were sequentially connected, their output sizes were reduced to 50, 30, and 10, respectively. Essentially, the feature finally encoded through the encoder became a feature vector with a dimension of 10. The decoder decoded the characteristics in the reverse order of the encoder, and the final output was a vector that was the same as the input.

We set hyperparameters for the learning model as follows: a batch size varying with the volume of data, five epochs, and 0.001 as the learning rate. The optimizer used was Adam with default parameters. As we intended to proceed with anomaly detection for every event, we applied the mean absolute error (MAE) to obtain the loss for individual events as in Equation (1)

$$\frac{1}{n}\sum_{i=1}^{b}\sum_{j=1}^{n}\left|\hat{x}_{ij}-x_{ij}\right| \tag{1}$$

where b is the batch size and n is the feature size.

We selected the maximum and average values of the training loss as static thresholds in individual machines to capture an abnormal alert. The reason for selecting two thresholds is that a value larger than the largest loss learned during training can be considered a definite anomalous alert. However, when the maximum loss of the training dataset is higher than that of other losses produced by outliers, it degrades the performance of the model. Therefore, we selected the average of the losses to solve this problem. Consequently, the model tended to become more sensitive when treating a normal packet as an attack; however, the detection ratio for the actual attacks improved.

3.4. Integrated Relevance Analysis

We defined the IRA to suggest a method for classifying attacks through an integrated analysis, regardless of the data types, because the data obtained in the real world are not only numeric and categorical data. The chi-squared test has two purposes: testing the goodness of fit and determining the dependence between two variables. Thus, we adopted the chi-squared test to determine the independence of the categorical data. To apply the chi-squared test, the numerical data were transformed into categorical data. Numerical values were converted directly to categorical values or converted into grades with each interval containing a range of values being classified as a categorical value.

However, with the chi-squared test, it is possible to confirm the existence of a relationship between each variable, but it is not possible to determine whether the relationship is strong or weak. Therefore, we used Cramér's V to measure the degree of association between two features. The advantage of Cramér's V technique is its ability to obtain the correlation coefficient between features divided into two or more categorical values, and this is independent of the continuity of variables or the distribution of their values. Further, we applied the Pearson correlation coefficient (PCC) to verify the correlation between the values for each feature. Using PCC, it became possible to check whether a feature value existed in the attack type and the positive or negative correlations between values. In summary, the IRA aimed to analyze the correlations and identify attacks using the results.

The chi-squared test was performed using Equation (2), where X is the observed value, $\overline{X}$ is the expected value, k is the number of categories, and i is the "i-th" position in the contingency table.

$$X^2=\sum_{i=1}^{k}\frac{(X_i-\overline{X}_i)^2}{\overline{X}_i}. \tag{2}$$

Cram ér's V was calculated using Equation (3),

$$\phi_c = \sqrt{\frac{\chi^2}{N(k-1))}}, \tag{3}$$

where χ^2 is the chi-squared test value, N is the number of observations, and k is $min(r-1, c-1)$, where r and c are the numbers of rows and columns in the contingency table, respectively. PCC was computed as in Equation (4), where cov is the covariance, σ_X is the standard deviation of X, and σ_y is the standard deviation of Y

$$\rho_{XY} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}. \tag{4}$$

4. Experimental Results

Through experiments, we demonstrated the results of the methodology described in Section 3. The results of IRA were obtained using two datasets. We created a confusion matrix and compared the number of detected alerts, and we collected alerts to evaluate the performance of our model. The distinction between the total number of alerts collected and the number of attack alerts filtered through the model showed that the number of unnecessary alerts was reduced, and the correlation for the attack type was verified using IRA.

4.1. Experimental Environment

The detection model was trained on a server with an Xeon CPU E5-2630 v4, it is manufactured by Intel Corporation in Santa Clara, California, USA @ 2.20 GHz and six Titan Xp GPUs constructed Nvidia Corporation in Santa Clara, CA, USA. We built the model with Python 3.6 and PyTorch, which is an open-source machine learning library developed primarily by Facebook's AI Research lab (FAIR) in Menlo Park, California, USA. Our model used an unsupervised learning algorithm and our collected dataset was unlabeled, sharing the same disadvantage of most IDS/Firewall datasets. Furthermore, it was important to consider the time padding before and after an attack because of the policy of sending alerts, that is, whether the machine sends alerts immediately as they are created, 5 min after the fact, or when the buffer of the log is full.

4.2. Anomaly Detection

4.2.1. Results of Individual Machines

While metrics such as accuracy and F1 score are important in evaluating model performance, False Positive Rate (FPR) and False Negative Rate (FNR) are equally critical. A lower FPR indicates that the model accurately identifies normal behavior, whereas a lower FNR means it is less likely to misclassify actual attacks as benign. In the context of network intrusion detection, minimizing False Negatives, where attacks go undetected, is a bigger concern than False Positives, where normal data are mistakenly flagged as malicious.

To benchmark state-of-the-art anomaly detection methods, we conducted experiments using PCA and LSTM autoencoder. The highest values for each metric are presented in bold, the second-highest are underlined, and the lowest FPR and FNR values are marked in yellow. As shown in Table 7, when the threshold is set to the maximum loss, PCA performs well in terms of FPR. However, it lags behind other models in F1 score and shows significantly higher FNR values on certain machines. The autoencoder and LSTM autoencoder deliver comparable results across most metrics, with the autoencoder achieving the lowest FNR overall.

As observed in the confusion matrix in Table 7, most of the precision and recall values of our model have high scores. However, some machines have low accuracy because of false positives or false negatives. As can be observed in the previous tables, most devices exhibited higher results when the threshold was the average value. When the threshold is adjusted to the average loss, a similar trend is observed. However, with a lower threshold, normal instances are more frequently misclassified as false positives, resulting in a higher FPR. For LSTM, only minor differences are observed compared to the regular autoencoder, suggesting that the time-series nature of the data does not have a significant impact on performance.

Further analyzing the autoencoder's performance, Table 8 lists the number of detection alerts per machine for each attack based on the threshold value and dataset type. Based on this table, Machine I exhibits extreme differences depending on the threshold. Machines III and IV show 100% detection performance when using the average threshold. In the case of Machine I using the average threshold, our model detected almost 80% of both SMB and DCERPC attacks in the dataset. However, when we adopted the maximum threshold, the detection rate was only approximately 30%. Most of the alerts collected in SMB and DCERPC attacks, excluding detected alerts with high loss value as an attack, were likely to be false alerts because their loss values were close to the thresholds. Therefore, the detection rate increased because of false alerts that exceeded the threshold.

Table 7. Confusion matrix with the threshold varying for each machine.

Threshold	Method	Metric	Machine						
			I	II	III	IV	V	Snort	Suricata
Maximum of training loss	PCA	Accuracy	0.5645	0.7015	0.9477	0.7712	0.1186	0.9758	0.9832
		Precision	**0.9723**	**1.0000**	**0.9904**	**1.0000**	**1.0000**	**0.9956**	**0.9972**
		Recall	0.2891	0.5964	0.9541	0.6997	0.1150	0.9792	0.9858
		F1-Score	0.4457	0.7472	0.9719	0.8233	0.2063	0.9873	0.9914
		FPR	0.0127	0.0000	0.1657	0.0000	0.0000	0.1091	0.1795
		FNR	0.7109	0.4036	0.0459	0.3003	0.8850	0.0208	0.0142
	AE	Accuracy	**0.8952**	0.9739	**0.9626**	**0.7776**	**0.9960**	0.9815	**0.9913**
		Precision	0.9654	0.9662	0.9813	0.9881	0.9960	0.9928	0.9951
		Recall	**0.8577**	**0.9996**	**0.9793**	**0.7168**	**1.0000**	**0.9879**	**0.9961**
		F1-Score	**0.9083**	0.9826	0.9803	**0.8308**	**0.9980**	0.9903	0.9956
		FPR	0.0472	0.0992	0.3346	0.0276	1.0000	0.1782	0.3132
		FNR	0.1423	0.0004	0.0207	0.2832	0.0000	0.0121	0.0039
	LSTM-AE	Accuracy	0.8949	**0.9741**	**0.9626**	**0.7776**	**0.9960**	**0.9817**	**0.9913**
		Precision	0.9660	0.9666	0.9813	0.9881	0.9960	0.9936	0.9951
		Recall	0.8566	0.9995	0.9791	**0.7168**	**1.0000**	0.9873	**0.9961**
		F1-Score	0.9080	**0.9828**	0.9802	**0.8308**	**0.9980**	**0.9905**	**0.9961**
		FPR	0.0463	0.0981	0.3340	0.0276	1.0000	0.1582	0.3132
		FNR	0.1434	0.0005	0.0209	0.2832	0.0000	0.0127	0.0039
Average of training loss	PCA	Accuracy	0.5769	0.7243	0.9470	0.7574	0.1186	0.9686	0.9833
		Precision	0.6479	0.8121	0.9687	**0.8099**	**1.0000**	0.9786	0.9909
		Recall	0.6599	0.8161	0.9756	0.8905	0.1150	0.9890	0.9922
		F1-Score	0.6539	0.8141	0.9721	0.8483	0.2063	0.9838	0.9915
		FPR	0.5505	0.5362	0.5650	0.6685	0.0000	0.5428	0.5824
		FNR	0.3401	0.1839	0.0244	0.1095	0.8850	0.0110	0.0078
	AE	Accuracy	0.6590	**0.9518**	**0.9632**	**0.7618**	**0.9960**	0.9719	**0.9901**
		Precision	0.6459	**0.9388**	**0.9687**	0.7618	0.9960	**0.9831**	**0.9928**
		Recall	**0.9671**	**0.9999**	**0.9932**	**1.0000**	**1.0000**	0.9879	0.9972
		F1-Score	0.7745	**0.9684**	**0.9808**	**0.8648**	**0.9980**	0.9854	**0.9950**
		FPR	0.8140	0.1850	0.5739	1.0000	1.0000	0.4266	0.4634
		FNR	0.0329	0.0001	0.0068	0.0000	0.0000	0.0121	0.0028
	LSTM-AE	Accuracy	**0.8317**	0.8393	0.9582	0.7618	**0.9960**	**0.9732**	0.9883
		Precision	**0.8461**	0.8217	0.9671	0.7618	0.9960	0.9732	0.9905
		Recall	0.8826	0.9995	0.9895	**1.0000**	**1.0000**	**0.9925**	**0.9976**
		F1-Score	**0.8640**	0.9019	0.9782	**0.8648**	**0.9980**	**0.9862**	0.9941
		FPR	0.2465	0.6157	0.6023	1.0000	1.0000	0.5108	0.6099
		FNR	0.1174	0.0005	0.0105	0.0000	0.0000	0.0075	0.0024

The highest values for each metric are presented in bold, the second-highest are underlined.

Table 8. Number of detected alerts with the dataset with the average and maximum threshold varying for each machine.

Threshold	Class	Machine						
		I	II	III	IV	V	Snort	Suricata
Maximum of training loss	HTTP	4279 (100%)	16,565 (100%)	8369 (94.13%)	2954 (86.32%)	4579 (100%)	51,267 (99.96%)	2064 (99.52%)
	MySQL	2188 (98.47%)	3311 (99.94%)	292 (66.67%)	0 (0%)	-	204 (34.63%)	104 (60.47%)
	SMB	391 (31.23%)	2435 (100%)	823 (95.59%)	0 (0%)	3 (100%)	0 (0%)	42 (42.86%)
	DCERPC	128 (21.81%)	1234 (99.12%)	676 (100%)	366 (100%)	-	735 (100%)	245 (98.79%)
	DDoS	1433 (97.09%)	10,360 (100%)	23,161 (100%)	0 (0%)	2130 (100%)	13,333 (100%)	32,286 (100%)
Average of training loss	HTTP	4279 (100%)	16,565 (100%)	8763 (98.56%)	3422 (100%)	4579 (100%)	51,267 (99.96%)	2064 (99.52%)
	MySQL	2222 (100%)	3311 (99.94%)	374 (85.39%)	328 (100%)	-	204 (34.63%)	123 (71.51%)
	SMB	1056 (84.35%)	2435 (100%)	823 (95.59%)	284 (100%)	3 (100%)	0 (0%)	58 (59.18%)
	DCERPC	460 (78.36%)	1245 (100%)	676 (100%)	366 (100%)	-	735 (100%)	248 (100%)
	DDoS	1476 (100%)	10,360 (100%)	23,161 (100%)	232 (100%)	2130 (100%)	13,333 (100%)	32,286 (100%)

4.2.2. Results of Individual Machines with Snort

We compared the changes in performance when using only the alert information of a single security device to detect anomalies by combining the detection event results of other security devices. For this experiment, because the same security device had to be used for comparison, we selected Snort, which is the open-source IDS with the largest number of detected events (i.e., responded the most to attacks). Table 9 summarizes the performance of each machine when the machine is used with or without Snort. An operational advantage was obtained by combining Snort with security machines with significantly lower performance, such as machines I and IV. In particular, a marked reduction in the FNR was observed, indicating an enhanced sensitivity of the security machines. This enhancement significantly bolsters the machine's capability to accurately detect and identify attacks without omissions, thereby improving the overall reliability of the security measures in place. However, for the remaining machines with performance similar to Snort, no improvement in performance was observed.

Table 9. Comparison of metrics of individual machines with or without Snort.

Threshold	**Cooperation**	**Metric**	**Machine**					
			I	**II**	**III**	**IV**	**V**	**Suricata**
Maximum of training loss	Without Snort	Accuracy	0.8952	0.9739	0.9626	0.7776	0.9960	0.9913
		Precision	0.9654	0.9662	0.9813	0.9881	0.9960	0.9951
		Recall	0.8577	0.9996	0.9793	0.7168	1.0000	0.9961
		F1-Score	0.9083	0.9826	0.9803	0.8308	0.9980	0.9956
		FPR	0.0472	0.0992	0.3346	0.0276	1.0000	0.3132
		FNR	0.1423	0.0004	0.0207	0.2832	0.0000	0.0039
	With Snort	Accuracy	0.8754	0.9278	0.9001	0.9302	0.9090	0.8708
		Precision	0.9911	0.9805	0.9878	0.9858	0.9946	0.9945
		Recall	0.8755	0.9347	0.9074	0.9312	0.9104	0.8708
		F1-Score	0.9298	0.9570	0.9459	0.9577	0.9506	0.9285
		FPR	0.1269	0.1148	0.2923	0.0753	0.1274	0.1295
		FNR	0.1245	0.0653	0.0926	0.0688	0.0896	0.1292
Average of training loss	Without Snort	Accuracy	0.6590	0.9518	0.9632	0.7618	0.9960	0.9901
		Precision	0.6459	0.9388	0.9687	0.7618	0.9960	0.9928
		Recall	0.9671	0.9999	0.9932	1.0000	1.0000	0.9972
		F1-Score	0.7745	0.9684	0.9808	0.8648	0.9980	0.9950
		FPR	0.8140	0.1850	0.5739	1.0000	1.0000	0.4634
		FNR	0.0329	0.0001	0.0068	0.0000	0.0000	0.0028
	With Snort	Accuracy	0.9603	0.8632	0.9647	0.8490	0.9751	0.9732
		Precision	0.9713	0.8628	0.9647	0.8490	0.9879	0.9803
		Recall	0.9870	1.0000	1.0000	1.0000	0.9862	0.9922
		F1-Score	0.9791	0.9264	0.9820	0.9184	0.9871	0.9862
		FPR	0.4711	0.9820	0.9588	1.0000	0.3082	0.5339
		FNR	0.0130	0.0000	0.0000	0.0000	0.0138	0.0078

4.3. Filtering the Alerts

To accurately recognize and respond to security threats with limited human resources, alerts should be managed by an administrator within a cognitive load range. Thus, for efficient monitoring, it is necessary to filter out false positives in normal situations and select only relevant alerts during an attack. We confirmed the application effect of the proposed model in terms of alert filtering for the analysis. For this, the ratio of the remaining alerts after filtering the alert (c_t) that exceeded the threshold and the total alert (b_t) by the security device during the normal and attack intervals (t) was calculated as follows:

$$\gamma = \sum_{i=0}^{t} \frac{(b_t - c_t)}{a_i}, \text{ where } t \geq 0. \tag{5}$$

Here, γ represents the cumulative ratio of alerts that were not estimated as abnormal by the attack detector (AD) of the proposed model to the total alerts. When γ is 1, no alerts are filtered as exceeding the threshold because the number of alerts that do not exceed the threshold matches the total number of alerts in each time window. Fundamentally, not all the alerts should be analyzed by the administrator. However, when γ is 0, all alerts exceed the threshold in each time window, and the administrator should analyze all the alerts.

Therefore, as γ approaches 1, the number of target alerts that the administrator should analyze approaches 0, and the reverse holds true. As shown in Figure 5, the range of γ increased from 0.016 (Suricata) to 0.6 (Machine III) when the AD threshold of the proposed model was the maximum value of learning loss and from 0 (Machine IV) to 0.33 (Machine I) when the average value of learning loss was calculated. This demonstrates that the overall filtering effect was identified.

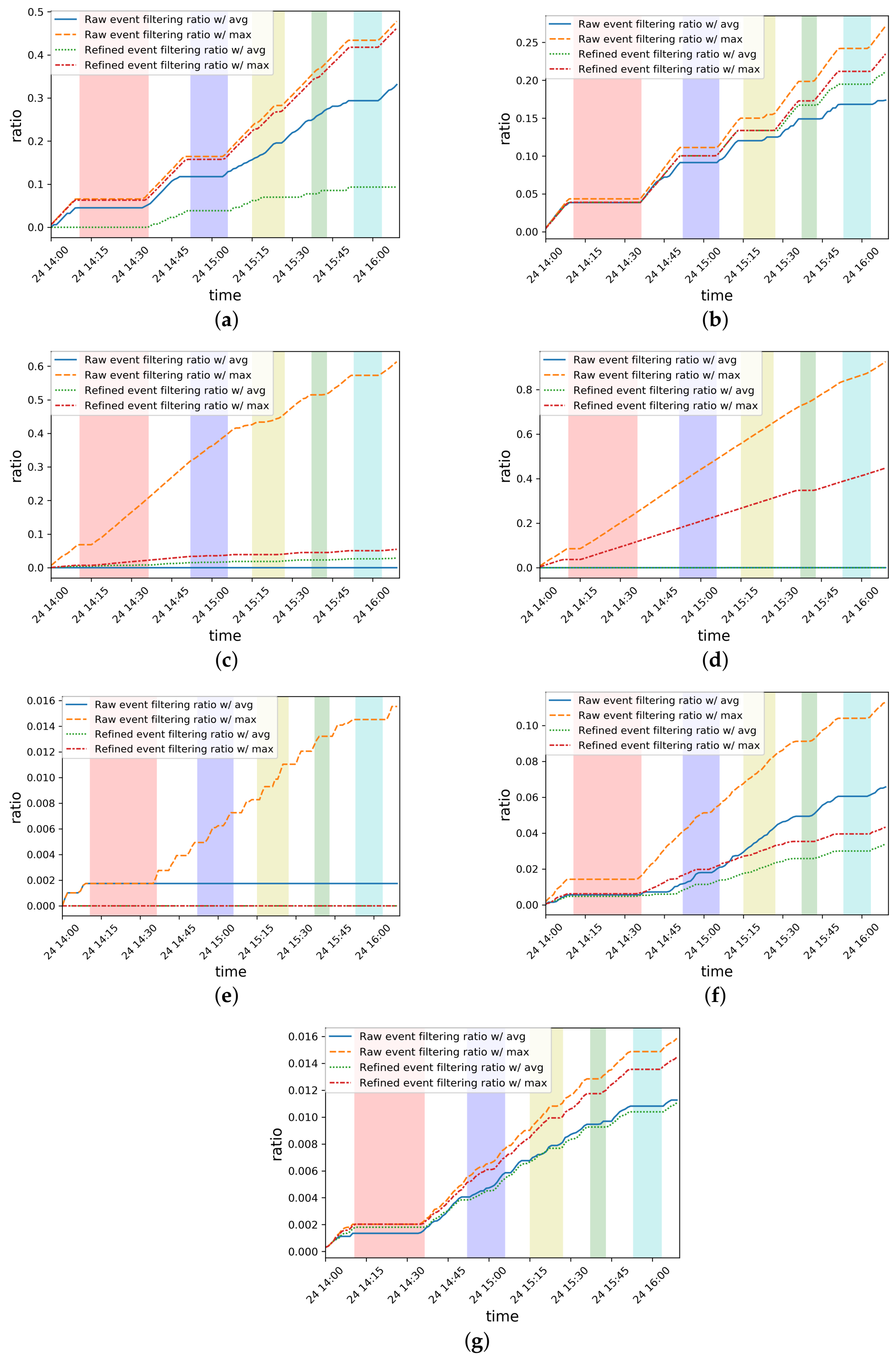


Figure 5. Ratio of security alerts exceeding thresholds in each machine. (**a**) Machine I, (**b**) Machine II, (**c**) Machine III, (**d**) Machine IV, (**e**) Machine V, (**f**) Snort, (**g**) Suricata.

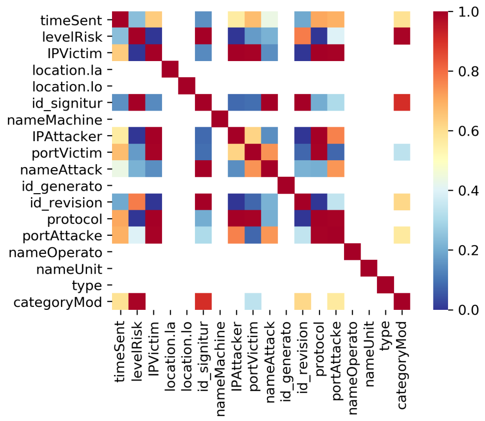
As the F1-score of the AD was significantly high (between 0.96 and 1), and relatively few alerts could be filtered, the alert filtering effect of Machine V, Snort, and Suricata was insignificant. However, in other security devices, γ was extremely high when filtering was based on learning loss, and the gap widened more when the raw alerts were compared with refined alerts. During the attack period, the alerts determined to be related to the attack were not filtered out; hence, γ remained approximately constant, and it continued to increase in other sections, confirming that it can help manage the cognitive load of the administrator. This phenomenon is most clearly observed in Machine II.

4.4. Integrated Relevance Analysis

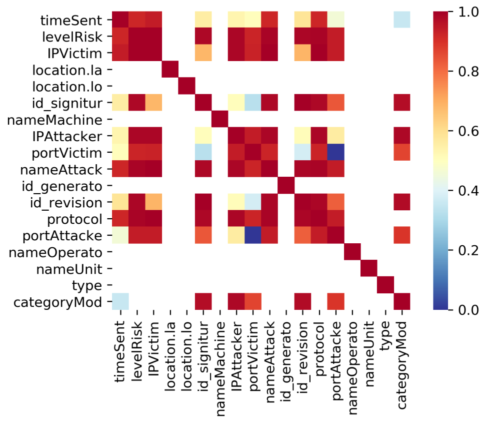
As explained in Section 3.4, we present the correlation between features through the IRA and illustrate the possibility of classifying attack types. Our model succeeded in identifying attack types; however, it was not possible to analyze attacks within an attack type because we tested various attacks in a short time. In addition, we demonstrated that each attack could be classified using the NSL–KDD dataset. The NSL–KDD dataset is divided into four types of attacks and consists of a total of 39 labeled attacks and normal data. The number of attacks and the number of alerts included in each type are listed in Table 10. There are 42 NSL–KDD attributes; eight attributes are categorical data whereas the rest are numeric data. Even in categorical data, only *class*, *protocol_type*, *service*, and *flag* are character types. *class* is used only for data classification, and *protocol_type*, *service*, and *flag* are preprocessed with one-hot encoding for PCC.

The visualization of Cramér's V calculations for different attack types using Snort device data are shown in Figure 6a–d. By identifying variables with a high correlation across all attacks through the Cramér's V results, these variables can be used as indicators to assess whether new input data represent an attack. Additionally, variables that exhibit different correlations depending on the type of attack can help in determining the specific nature of the attack. For example, the correlation between portVictim and portAttacker remains below 0.5 for most attacks, but for DDoS attacks, the value approaches 0.7. This suggests that a high correlation between portVictim and portAttacker within a certain range could be an indicator of a DDoS attack. However, it should be noted that Cramér's V can only be calculated for categorical variables and does not provide insights into relationships at the individual category level. To address this limitation, one-hot encoding is applied to the categorical variables, followed by the calculation of Pearson Correlation Coefficients (PCC). The results are displayed in Figure 6h–k, enabling a deeper understanding of correlations between specific variable values. For instance, if a DDoS attack is detected and PCC is applied to data within the attack window, it can reveal which specific ports between portVictim and portAttacker are correlated. If there is a high correlation between portVictim-3389 and portAttacker-6001, this would indicate that the attack is targeting port 3389, allowing security analysts to respond by either closing port 3389 or blocking connections originating from port 6001. This approach enhances the ability of security teams to respond to threats effectively.

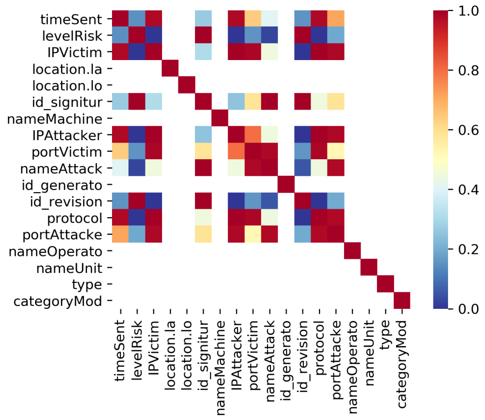
The results of the IRA from the NSL–KDD dataset are shown in Figure 6e–g,l–n according to the chi-squared test and Cramér's V results, where the Apache2 attack shows a high correlation of *logged_in* feature with *duration* and *flag* features. In the case of a multihop attack, similar to Apache2, there is a high correlation with 15 features, including *duration* and *logged_in*; however, the flag feature does not appear. In a snmpguess attack, *protocol_type* and *service* show the same color association, and both features have a high correlation with the following features: *dst_host_same_src_port_rate*, *dst_host_count*, and *dst_host_srv_count*.



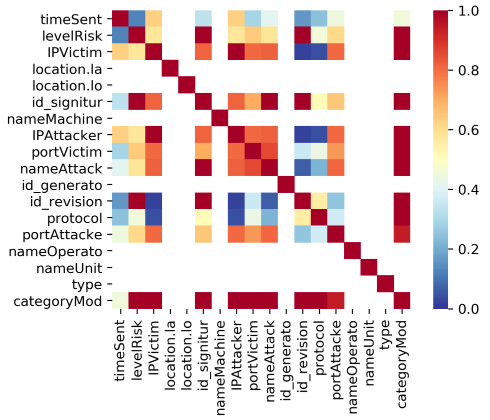
(**a**)



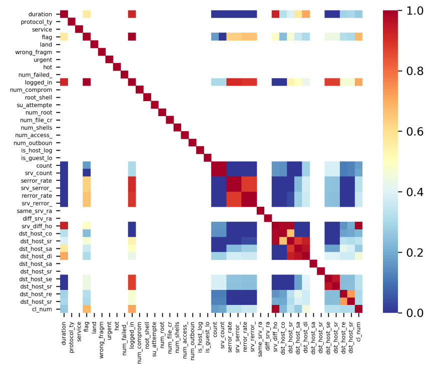
(**b**)



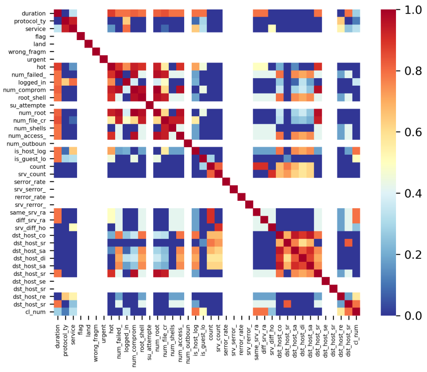
(**c**)



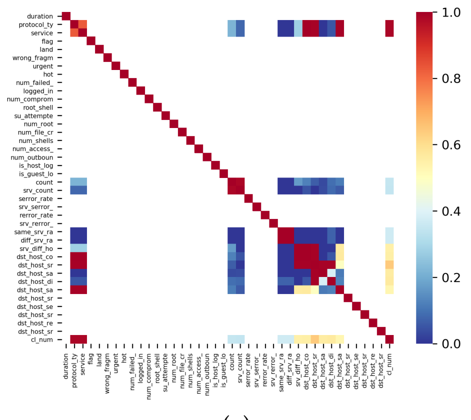
(**d**)



(**e**)

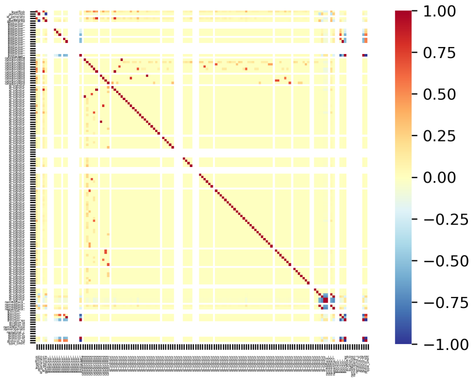


(**f**)

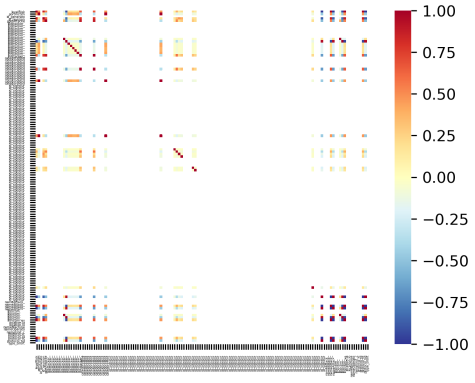


(**g**)

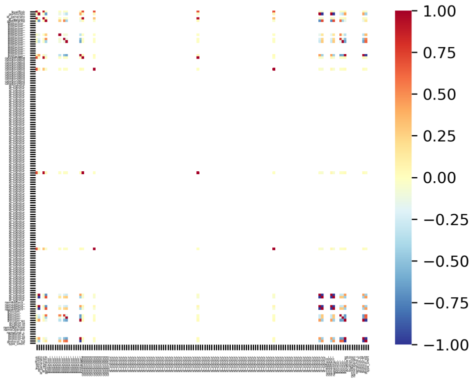
Figure 6. *Cont.*



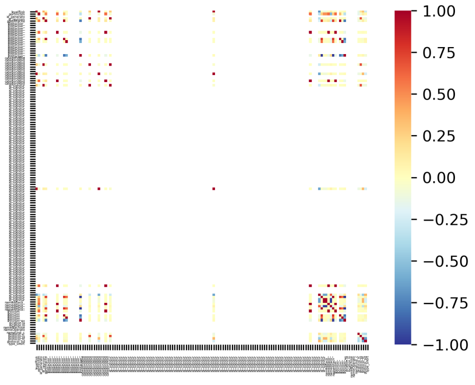
(**h**)



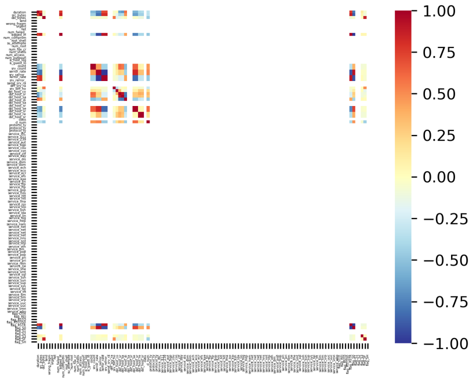
(**i**)



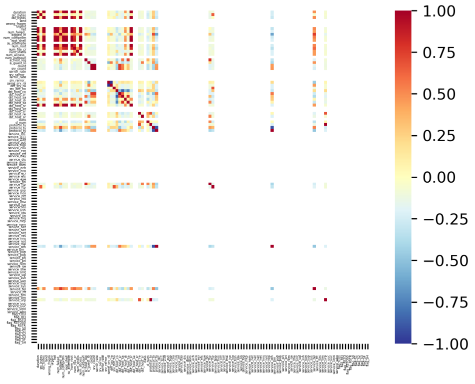
(**j**)



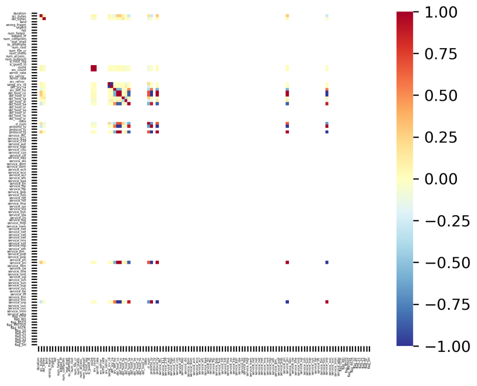
(**k**)



(**l**)



(**m**)



(**n**)

Figure 6. Results of Chi-squared test, Cramér's V and PCC applied to Snort and NSL-KDD. (**a**) HTTP, (**b**) MySQL, (**c**) DCERPC, (**d**) DDoS, (**e**) Apache2, (**f**) Multihop, (**g**) Snmpguess, (**h**) HTTP, (**i**) MySQL, (**j**) DCERPC, (**k**) DDoS, (**l**) Apache2, (**m**) Multihop, (**n**) Snmpguess.

Table 10. NSL-KDD dataset summary.

Class	Distinct Attack	Dataset		
		Train	Test	Subtotal
Normal	-	67,343	9711	77,054
DoS	11	45,927	7460	53,387
Probe	6	11,656	2421	14,077
R2L	15	995	2885	3880
U2R	7	52	67	119
Total	39	125,973	22,544	71,463

We also examined multihop and snmpguess attacks, which show different patterns despite having the same R2L attack type. Nevertheless, in the PCC results, more than one *service* value exists in both the multihop and snmpguess attacks, but it is not observed in the Apache2 attack. Instead, a *flag* feature can be found in the Apache2 attack. Multihop and snmpguess attacks can be assumed to be of the same type because the *service* feature appears in both attacks. Furthermore, the presence or absence of a variable value uniquely appearing in the *service* feature and the values that are not visible in the PCC result of the snmpguess can be used to distinguish between the two types of attacks.

5. Discussion

- **Limitation of the dataset to evaluate performance:** Based on the experimental results, our model detects five different attack situations with one round of training using only normal data. According to Table 9, the model combined with Snort maintains an FNR between 0 and 0.12. Therefore, we expect that this model will be able to detect even the emergence of new attacks. However, as our model was trained on normal data, the performance of the model may vary depending on the state and volume of normal data and the difference between normal and attack data. It is important to collect high-quality normal data or refine the data to increase their quality. In our preprocessing steps, we applied various approaches such as removing IP addresses, which cannot represent attackers or targets, dropping meaningless fields, and determining numeric and categorical fields for data encoding. In addition, our dataset lacks a variety of attack cases, such as multiple attacks of the same type or different types of attacks occurring simultaneously. Furthermore, we could not distinguish attack types because the period between attacks was extremely short. Even if there was a difference in the density of attacks, further research is required to confirm whether the same level of performance can be maintained.
- **Heterogeneous machines and domains:** If a device is located in the outermost part of the network, it generates more logs than those located inside, or if different machines exist in the same location, a different number of logs can be generated depending on the device's policy. Thus, the volume of data can vary because the size of data generation varies according to the devices and locations of the machines set up in the industrial domain. Therefore, we manually set the batch size by checking the corresponding volume; however, we also needed to determine a suitable batch size and the criteria for small data.
- **Static threshold for anomaly detection:** Because we used a static threshold that linearly separates the normal and abnormal as the average and maximum values of the training loss, false alerts were sometimes considered true alerts. As shown in Table 7, the performance differs depending on the threshold; specifically, Machine I shows a difference of approximately 2.0. Some solutions for handling this phenomenon are as follows: (1) Sampling the alerts appearing in the normal interval between attacks for training. (2) Choosing an optimal threshold to minimize false alerts and maximize the detection of actual alerts, for example, by using the median or assigning weights with a histogram of training losses. Additionally, we plan to find a method for the model to optimize batch size and thresholds regardless of the dataset.

- **Operation time:** In a situation where real-time detection is important, a long training time causes problems. In this study, the training time differed depending on the size of the dataset for each machine. In addition, the operation time was affected by both the dataset size and input size. As presented in Table 5, after data preprocessing, the number of fields at least doubles for all machines except Machine IV. When other machines are combined with Snort, the number of fields increases in proportion with the size of Snort, leading to an increase in operation time. When more than two machines are combined, the number of fields increases even further. To address this problem, a more efficient model should be developed or fields that have been proven to have low relevance to attacks should be removed, either through IRA or deep learning analysis.
- **Comparative analysis to create a naive autoencoder:** In this study, we used a naive autoencoder to examine the possibility of its application and to use it as a baseline for future work. It is also necessary to compare its performance with other state-of-the-art models, such as the VAE, Conv-AE, or attention techniques. To implement anomaly detection, we adopted the loss of the batch and, by checking the loss of individual fields in the batch, we also observed that the loss in specific fields had some effect; however, this was not addressed in this study. In future research, we aim to extract the pattern of each attack by combining the IRA and the loss results of individual fields.

6. Conclusions

In this paper, we introduced a method that can detect abnormal situations and classify attack types by targeting alert information from heterogeneous environments. The proposed method preprocesses various types of data from heterogeneous security machines to provide a robust detection model for the actual environment, while also reducing the number of false alerts. This model not only detects abnormal situations but also assists in identifying alerts that require intensive analysis. Therefore, any SoC can detect and respond to incidents in a timely manner based on attacks by drastically reducing the immense volume of alerts. Moreover, we verified that this method is capable of identifying the types of abnormal situations.

Author Contributions: Conceptualization, Y.I.J.; Data curation, S.C. and B.-G.M.; Formal analysis, Y.I.J.; Funding acquisition, B.-G.M.; Investigation, Y.I.J.; Methodology, Y.I.J.; Project administration, Y.-J.C.; Resources, S.C., B.-G.M. and Y.-J.C.; Software, Y.I.J.; Supervision, Y.-J.C.; Validation, Y.I.J., S.C., B.-G.M. and Y.-J.C.; Visualization, Y.I.J.; Writing—original draft, Y.I.J., S.C. and Y.-J.C.; Writing—review and editing, Y.I.J., S.C. and Y.-J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is the result of commissioned research project supported by the affiliated institute of ETRI [2020-046].

Data Availability Statement: Restrictions apply to the availability of these data. Data were obtained from the affiliated institute of ETRI and are not available due to the policy of the affiliated institute of ETRI.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had the role in the decision to publish the results.

References

1. Cybersecurity & Infrastructure Security Agency. DarkSide Ransomware: Best Practices for Preventing Business Disruption from Ransomware Attacks. Available online: https://www.cisa.gov/news-events/cybersecurity-advisories/aa21-131a (accessed on 13 October 2024).
2. Federal Bureau of Investigation, Cyber Division. Conti Ransomware Attacks Impact Healthcare and First Responder Networks. Available online: https://www.cisa.gov/sites/default/files/publications/Conti%20Ransomware%20Healthcare%20Networks.pdf (accessed on 13 October 2024).
3. Stouffer, K.; Pease, M.; Tang, C.; Zimmerman, T.; Pillitteri, V.; Lightman, S. *NIST SP 800-82 rev.3(Draft): Guide to Operational Technology (OT) Security*; National Institute of Standards and Technology Special Publication: Gaithersburg, MD, USA, 2022. Available online: https://csrc.nist.gov/pubs/sp/800/82/r3/ipd (accessed on 13 October 2024).

4. Conti, M.; Donadel, D.; Turrin, F. A Survey on Industrial Control System Testbeds and Datasets for Security Research. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2248–2294. [CrossRef]
5. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
6. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur.* **2014**, *45*, 100–123. [CrossRef]
7. Balkanli, E.; Alves, J.; Zincir-Heywood, A.N. Supervised learning to detect DDoS attacks. In Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Orlando, FL, USA, 9–12 December 2014; pp. 1–8.
8. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
9. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Autoencoder-based feature learning for cyber security applications. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3854–3861.
10. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the ICISSp, Madeira, Portugal, 22–24 January 2018; pp. 108–116.
11. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019; pp. 1–8. [CrossRef]
12. Jazi, H.H.; Gonzalez, H.; Stakhanova, N.; Ghorbani, A.A. Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling. *Comput. Networks* **2017**, *121*, 25–36. [CrossRef]
13. Maciá-Fernández, G.; Camacho, J.; Magán-Carrión, R.; García-Teodoro, P.; Therón, R. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Comput. Secur.* **2018**, *73*, 411–424. [CrossRef]
14. Tama, B.A.; Comuzzi, M.; Rhee, K.H. TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access* **2019**, *7*, 94497–94507. [CrossRef]
15. Qassim, Q.; Zin, A.M.; Ab Aziz, M.J. Anomalies Classification Approach for Network-based Intrusion Detection System. *Int. J. Netw. Secur.* **2016**, *18*, 1159–1172.
16. Atefi, K.; Hashim, H.; Khodadadi, T. A Hybrid Anomaly Classification with Deep Learning (DL) and Binary Algorithms (BA) as Optimizer in the Intrusion Detection System (IDS). In Proceedings of the 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), Langkawi, Malaysia, 28–29 February 2020; pp. 29–34.
17. Gamage, S.; Samarabandu, J. Deep learning methods in network intrusion detection: A survey and an objective comparison. *J. Netw. Comput. Appl.* **2020**, *169*, 102767. [CrossRef]
18. D'hooge, L.; Wauters, T.; Volckaert, B.; De Turck, F. In-depth comparative evaluation of supervised machine learning approaches for detection of cybersecurity threats. In Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS), Heraklion, Greece, 2–4 May 2019; pp. 125–136.
19. Hosseini, S.; Azizi, M. The hybrid technique for DDoS detection with supervised learning algorithms. *Comput. Netw.* **2019**, *158*, 35–45. [CrossRef]
20. Mebawondu, J.O.; Alowolodu, O.D.; Mebawondu, J.O.; Adetunmbi, A.O. Network intrusion detection system using supervised learning paradigm. *Sci. Afr.* **2020**, *9*, e00497. [CrossRef]
21. Kim, M. Supervised learning-based DDoS attacks detection: Tuning hyperparameters. *ETRI J.* **2019**, *41*, 560–573. [CrossRef]
22. Aksu, D.; Üstebay, S.; Aydin, M.A.; Atmaca, T. Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm. In Proceedings of the International Symposium on Computer and Information Sciences, Poznan, Poland, 20–21 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 141–149.
23. Hwang, R.H.; Peng, M.C.; Huang, C.W.; Lin, P.C.; Nguyen, V.L. An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection. *IEEE Access* **2020**, *8*, 30387–30399. [CrossRef]
24. Alom, M.Z.; Taha, T.M. Network intrusion detection for cyber security using unsupervised deep learning approaches. In Proceedings of the 2017 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 27–30 June 2017; pp. 63–69. [CrossRef]
25. Goh, J.; Adepu, S.; Tan, M.; Lee, Z.S. Anomaly detection in cyber physical systems using recurrent neural networks. In Proceedings of the 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), Singapore, 12–14 January 2017; pp. 140–145.
26. Schneider, P.; Böttinger, K. High-performance unsupervised anomaly detection for cyber-physical system networks. In Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy, Toronto, ON, Canada, 15–19 October 2018; pp. 1–12.
27. Tuor, A.; Kaplan, S.; Hutchinson, B.; Nichols, N.; Robinson, S. Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA 4–9 February 2017.

28. Meira, J.; Andrade, R.; Praça, I.; Carneiro, J.; Bolón-Canedo, V.; Alonso-Betanzos, A.; Marreiros, G. Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. *J. Ambient Intell. Humaniz. Comput.* **2020**, *11*, 4477–4489. [CrossRef]
29. Nguyen, Q.P.; Lim, K.W.; Divakaran, D.M.; Low, K.H.; Chan, M.C. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. In Proceedings of the 2019 IEEE Conference on Communications and Network Security (CNS), Washington, DC, USA, 10–12 June 2019; pp. 91–99. [CrossRef]
30. Rao, A.R.; Wang, H.; Gupta, C. Functional approach for Two Way Dimension Reduction in Time Series. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; pp. 1099–1106. [CrossRef]
31. Karimipour, H.; Dehghantanha, A.; Parizi, R.M.; Choo, K.K.R.; Leung, H. A deep and scalable unsupervised machine learning system for cyber-attack detection in large-scale smart grids. *IEEE Access* **2019**, *7*, 80778–80788. [CrossRef]
32. Kundu, A.; Sahu, A.; Serpedin, E.; Davis, K. A3D: Attention-based auto-encoder anomaly detector for false data injection attacks. *Electr. Power Syst. Res.* **2020**, *189*, 106795. [CrossRef]
33. Zhang, C.; Song, D.; Chen, Y.; Feng, X.; Lumezanu, C.; Cheng, W.; Ni, J.; Zong, B.; Chen, H.; Chawla, N.V. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 1409–1416.
34. Choi, W.H.; Kim, J. Unsupervised Learning Approach for Anomaly Detection in Industrial Control Systems. *Appl. Syst. Innov.* **2024**, *7*, 18. [CrossRef]
35. Choi, S.; Yun, J.H.; Min, B.G.; Kim, H. POSTER: Expanding a Programmable CPS Testbed for Network Attack Analysis. In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, ASIA CCS '20, Taipei, Taiwan, 5–9 October 2020; pp. 928–930. [CrossRef]