Cross-Vendor Knowledge Transfer for Managed Security Services with Triplet Network

Toshiki Shibahara
NTT Secure Platform Laboratories
toshiki.shibahara.de@hco.ntt.co.jp

Hirokazu Kodera
NTT Secure Platform Laboratories
hirokazu.kodera.dh@hco.ntt.co.jp

Daiki Chiba
NTT Secure Platform Laboratories
daiki.chiba@ieee.org

Mitsuaki Akiyama
NTT Secure Platform Laboratories
akiyama@ieee.org

Kunio Hato
NTT Secure Platform Laboratories
kunio.hato.gm@hco.ntt.co.jp

Ola Söderström
NTT Security
Ola.Soderstrom@nttsecurity.com

Daniel Dalek
NTT Security
Daniel.Dalek@nttsecurity.com

Masayuki Murata
Osaka University
murata@ist.osaka-u.ac.jp

ABSTRACT

Managed security services detect incidents, i.e., successful attacks such as malware infection, in real time from a large number of alerts based on vendors' and security operations center's (SOC's) detection rules. To immediately find incidents, professional analysts in a SOC prioritize alerts if their indicators, i.e., meta-information of detection rules in alerts, are highly correlated with incidents. Indicators are typically divided into two priority levels, i.e., primary and secondary. However, levels of new indicators are difficult to accurately determine with a conventional system. Such a system determines an indicator's level as primary if the conditional probability of incidents occurring given an observation of the indicator's alert is high. Therefore, we propose a system for accurately determining levels of new indicators by focusing on alerts not recognized as incidents. With this system, we analyze the correlation between indicators made by different vendors then transfer knowledge of incidents between different vendors with a triplet network. We evaluate the effectiveness of the proposed system using 4,919,791 alerts collected from a large-scale SOC for one month. Our system identified 24.3% more primary indicators undiscovered at the time of data collection than a system without correlation analysis at a 5% false positive rate.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Computing methodologies → Neural networks**.

KEYWORDS

alert analysis, security information and event management, metric learning, deep neural network

AISec'19, November 15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6833-9/19/11…$15.00
https://doi.org/10.1145/3338501.3357367

ACM Reference Format:
Toshiki Shibahara, Hirokazu Kodera, Daiki Chiba, Mitsuaki Akiyama, Kunio Hato, Ola Söderström, Daniel Dalek, and Masayuki Murata. 2019. Cross-Vendor Knowledge Transfer for Managed Security Services with Triplet Network. In *12th ACM Workshop on Artificial Intelligence and Security (AISec'19), November 15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3338501.3357367

1 INTRODUCTION

To minimize damage caused by cyber attacks, incidents, i.e., successful attacks such as malware infection, must be immediately detected after the occurrence. Managed security services (MSS) are aimed at detecting incidents in real time by monitoring customers' traffic in a security operations center (SOC) [20, 22]. To manage large-scale traffic, MSS leverages three different levels of analysis: security appliances, security information and event management (SIEM), and professional analysts. Customer traffic is first analyzed using security appliances such as unified threat management (UTM), which has vendor's detection rules and a logging function. If attacks are detected by the rules (e.g., malware hashes, regular expression signatures, machine-learning-based classifications), alerts are sent to SIEM. In parallel, communication logs, such as HTTP requests, are also sent to SIEM by the logging function. Next, logs sent from appliances are analyzed in SIEM with detection rules developed by a SOC. The SOC's detection rules are applied to logs from all customers and enhance the detection capabilities of MSS. If attacks are detected, alerts are sent to analysts along with alerts from appliances. Finally, professional analysts in a SOC manually analyze alerts detected with both vendors' and SOC's rules. Since an alert does not include sufficient information to determine whether an attack has succeeded, analysts have to manually confirm it. An alert consists of an indicator and identifier. An indicator is a tuple of meta-information of a vendor's or SOC's detection rule such as a rule name and communication direction. An identifier is a tuple of information identifying a communication related to an alert such as the timestamp and source/destination of a communication. Analysts manually find attack artifacts in communication logs related to alerts. If analysts confirm successful malware infection or exploitation of vulnerability on servers, alerts are recognized

as incidents. If analysts confirm failure of malware download or exploitation, alerts are not recognized as incidents.

Incidents must be immediately found from a large number of alerts to respond to and recover from incidents as soon as possible [15]. To reduce time required to find incidents, analysts intensively and carefully analyze alerts whose indicators are determined to be high priority levels. Indicators are typically divided into two priority levels, i.e., primary and secondary, in terms of correlation with incidents.

The number of indicators increases on a daily basis because vendors' detection rules are frequently updated once new common vulnerabilities and exposures (CVEs), malware variants, and exploit kits are discovered [21]. However, levels of new indicators are difficult to accurately determine with a conventional system [18]. This system can be generalized as that for determining a level of an indicator on the basis of the conditional probability of incidents occurring given that an alert of the indicator is observed. If the conditional probability is high, the level is determined as primary. In fact, incidents regarding some new indicators do not occur a few weeks or months after the indicators started to be observed in a SOC because incidents generally do not frequently occur. In this case, their conditional probabilities are 0, and their levels are determined as secondary at this time even if incidents regarding them will occur in the near future. We call such indicators, which are determined as secondary indicators but incidents regarding them will occur in the near future, *potential primary indicators*. In the worse case, alerts of potential primary indicators are not analyzed in a SOC, and the corresponding incidents remain undetected.

Therefore, we propose a system for identifying potential primary indicators by focusing on alerts not recognized as incidents, i.e., failed attacks. Such alerts of potential primary indicators are definitely observed, and we can infer traits of their indicators from increase, decrease, and duration of the alerts. Using these alerts, we analyze the correlation between indicators made by different vendors then transfer knowledge of incidents between different vendors. For example, we consider the case in which an incident regarding a newly discovered CVE occurred in a customer's network using an appliance of a certain vendor, but no incident regarding the same CVE occurred in other customers' networks using appliances of another vendor. Even if both vendors have made indicators regarding the CVE, a conventional system determines the indicator of the former vendor related to an incident as primary but determines the indicator of the latter vendor as secondary. We identify the latter vendor's indicator as a potential primary indicator by inferring that these indicators are related to the same type of attacks, e.g., attacks regarding the same CVE, from the similarity in increase, decrease, and duration of alerts.

The same types of attacks are observed at different vendors' appliances, and alerts are triggered by vendors' detection rules. However, alerts are biased by customers because the number of alerts triggered in a customer's network depends on whether attackers select the customer as targets of the attacks. To compare indicators of different vendors on the basis of their alerts, we need to reduce the bias of their alerts. To this end, we aggregate alerts of the same indicator from different customers. We expect the bias to be reduced by expanding observation range and increasing the number of alerts. On the basis of the above insight, we aggregate alerts from many customers by indicators and extract statistical traits of indicators from timestamps, sources, and destinations in the aggregated alerts, e.g., frequency, burstiness, and the number of unique sources/destinations [11]. We call a numerical vector consisting of the extracted traits *a detection pattern*. Even though biases are reduced by aggregating alerts, detection patterns of different vendors' indicators regarding the same type of attacks are not completely identical because different vendors' appliances are used by different customers.

We aim at eliminating difference in detection patterns resulting from vendors by leveraging indicators of a SOC's detection rules, which we call *SOC-made indicators*. We aggregate alerts of SOC-made indicators by vendors where the originated logs are observed then extract their detection patterns. Since SOC-made indicators are applied to all vendors' logs, we can obtain detection patterns based on the same indicators but different vendors. By comparing the detection patterns, we can quantitatively measure the difference in detection patterns resulting from their originated logs observed at different vendors' appliances. We transform detection patterns into low-dimensional vectors (representations), where we can calculate similarities between indicators of different vendors without being affected by the difference resulting from vendors.

To eliminate the difference resulting from vendors but keep other traits of detection patterns in representations, we simultaneously conduct the following two optimizations.

(1) preserve the similarity relationship between detection patterns of the same vendor's indicators
(2) minimize the difference between SOC-made indicators resulting from vendors

For the first optimization, we preserve the order of similarities between the same vendor's indicators not to contradict the second optimization. Optimization with the same setting is studied in metric learning, and a method called a triplet network was proposed [7]. A triplet network optimizes representations that preserve the order of similarities between samples using similar and dissimilar pairs of samples. Therefore, we apply a triplet network to our system in a very unique/new setting in terms of analyzing alerts observed in a SOC. By adding constraints for the second optimization to the representation learning of the triplet network, we can obtain the representations where similarities between indicators of different vendors can be calculated without being affected by the difference resulting from vendors.

We evaluate the effectiveness of our system using 4,919,791 alerts collected from a large-scale SOC for one month. Our system outperformed a conventional system that identifies potential primary indicators on the basis of the similarities between detection patterns. Specifically, our system identified 24.3% more primary indicators undiscovered at the time of data collection at a 5% false positive rate (FPR). From analyzing optimized representations, we also show that identification performance improved by decreasing the difference between SOC-made indicators resulting from vendors.

We make the following contributions:

- We propose a system that identifies potential primary indicators by transferring knowledge of incidents and show its effectiveness using alerts collected from a large-scale SOC.

Table 1: Examples of indicators

Indicator	Vendor	# of incidents	# of alerts	Level
V_1^A	A	20	100	Primary
V_2^B	B	1	100	Secondary
V_3^B	B	0	10	Secondary

- We show that representations for calculating similarities between different vendors' indicators are effectively optimized with a triplet network.

2 BACKGROUND

In this section, we describe motivating examples, notations, and a triplet network.

2.1 Motivating Example

To facilitate understanding of our motivation, we explain how we collect and analyze alerts and how our system works. Figure 1 shows the flow of our alert analysis. Each customer of an MSS deploys at least one security appliance such as UTM (❶). Generally, the number and vendors of appliances differ depending on the customer. For example, in Fig. 1, customers 1 and 2 use appliances of vendor A, while customer 3 uses an appliance of vendor B. Appliances, which have vendor's detection rules and a logging function, send alerts with vendor-made indicators and communication logs to SIEM (❷). Communication logs include HTTP requests, DNS queries/responses, and so on. By leveraging the communication logs, a SOC enhances its detection capability of incidents. SOC's detection rules are deployed to SIEM and raise alerts with SOC-made indicators from the communication logs. As a result, alerts with vendor-made and SOC-made indicators are sent to a SOC (❸). The circles, squares, and triangles represent different indicators, and their colors represent the vendors of appliances from which alerts and logs originate. For example, indicators in blue originate from traffic observed at the appliances of vendor A, and indicators in orange originate from that of vendor B. Indicators not filled with color represent vendor-made ones, and indicators filled with color represent SOC-made ones. Vendor-made indicators differ depending on the vendor. For example, all circle and triangle indicators are in blue and orange, respectively. On the other hand, SOC's detection rules are applied to all customers' logs in SIEM; thus, SOC-made indicators, i.e., square indicators, are in both blue and orange.

To efficiently analyze alerts in a SOC, indicators are divided into two levels: primary and secondary. Their levels are conventionally determined on the basis of their correlation with incidents [18]. For example, alerts regarding an indicator are collected during a period and confirmed if they are recognized as incidents. If the ratio of the number of alerts recognized as incidents to that of collected alerts is high, the indicator is determined as primary. In other words, the level is determined by a conditional probability of incidents given an indicator's alert. The conditional probability p is defined by the number of incidents n_i and the number of alerts n_a: $p = n_i/n_a$. Table 1 shows examples of vendor-made indicators. V in an indicator denotes a vendor-made indicator, its superscript denotes a vendor of an appliance from which alerts originate, and its subscript denotes an index of an indicator. The above conditional probabilities of V_1^A and V_2^B are $p(V_1^A) = 20/100$ and $p(V_2^B) = 1/100$. Since $p(V_1^A)$ is high but $p(V_2^B)$ is low, their levels are determined as primary and secondary, respectively. The conditional probability of V_3^B is $p(V_3^B) = 0/10 = 0$. Since $p(V_3^B)$ is low, its level is determined as secondary. As mentioned above, on the basis of the conventional system, indicators whose related incidents have not occurred are determined as secondary. However, V_3^B may be a potential primary indicator because the number of its alerts is small and sufficient data (i.e., alerts and related incidents) have not been collected. Its level will change to primary if its related incidents will occur in the near future. Our system predicts whether it is a potential primary indicator on the basis of the correlation with different vendors' indicators.

To analyze the correlation between indicators, we aggregate alerts from all customers by indicators and vendors (❹) then extract statistical values representing the traits of indicators (e.g., increase, decrease, and duration of attacks) as detection patterns (❺). Detection patterns of different vendors cannot be directly compared by distance metrics, such as L2 distance, because different vendors are used by different customers. Detection patterns of different vendors originate from different customers' traffic; thus, their elements (e.g., increase, decrease, and duration of attacks) differ resulting from the originated traffic. More precisely, different vendors are used by different numbers and scale of customers and used in different network structures. Figure 2(a) shows an example of a distribution of detection patterns. SOC-made indicators are drawn in Fig. 2(a) as well as vendor-made indicators. SOC-made indicators are denoted as S. For example, S_1^A is a SOC-made indicator regarding a SOC's detection rule and vendor A's communication logs, and S_1^B is a SOC-made indicator regarding the same detection rule but vendor B's communication logs. Similarly, S_2^A and S_2^B are regarding the same detection rule but originating from vendor A's and B's communication logs, respectively. In this figure, similar detection patterns are drawn close to each other. S_1^A and S_1^B are ideally expected to be similar but in fact dissimilar in detection patterns due to the difference resulting from vendors. Analogously, S_2^A and S_2^B are expected to be similar but are in fact dissimilar. This difference causes inaccurate similarity calculation between indicators of different vendors. V_1^A and V_3^B seem to be related to the same attack because both are in the middle of the SOC-made indicators. To identify similar indicators on the basis of detection patterns, V_1^A and V_3^B are ideally expected to be similar. However, in Fig. 2(a), V_1^A and V_3^B are dissimilar, but V_1^A and V_2^B are similar due to detection patterns' differences resulting from vendors.

Our system transforms detection patterns into representations where similarities of different vendor's indicators can be calculated without being affected by the difference resulting from vendors (❻). Figure 2(b) shows an example of an ideal distribution of representations optimized so that S_1^A and S_1^B are similar, S_2^A and S_2^B are similar, and the similarity relationship between indicators of the same vendor are preserved. As a result of the optimization, V_1^A and V_3^B become similar in representations. On the basis of these representations, we can identify potential primary indicators that are similar to known primary indicators of different vendors (❼).

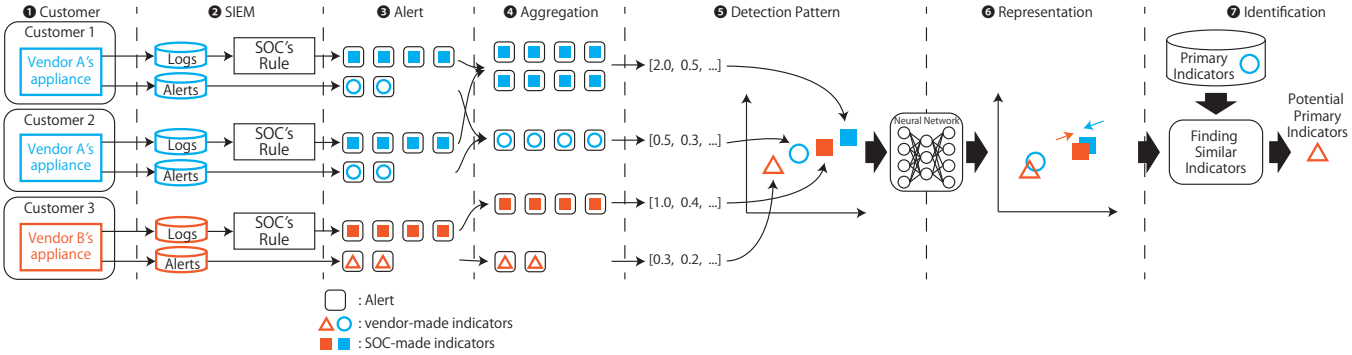

Figure 1: Flow of alert analysis in our system

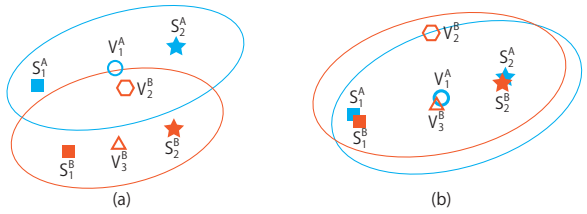

Figure 2: Distribution of (a) detection patterns and (b) representations

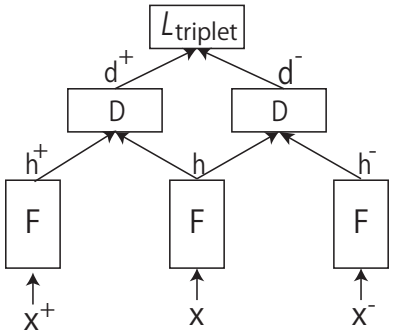

Figure 3: Triplet network

To obtain such representations, we need to optimize representations from two perspectives: (1) preserving the similarity relationship between indicators of the same vendor and (2) minimizing the difference between SOC-made indicators resulting from vendors. Since two optimizations need to be conducted at the same time, we have to design their constraints to not be inconsistent with each other. Specifically, we cannot define the constraint of the first optimization using the similarity values between detection patterns. If we constrain to preserve the distance in detection patterns between S_1^A and S_2^A and between S_1^B and S_2^B for the first optimization, the second optimization is impossible because the distance between S_1^A and S_2^A differs from that between S_1^B and S_2^B. Therefore, we preserve the similarity order of the same vendors for the first optimization. This constraint is sufficient because we can maintain V_1^A and V_1^B in the middle of the SOC-made indicators. Optimizations of representations for calculating the similarities between samples have been studied in metric learning. We apply a triplet network [7] to our system because it conducts optimization on the basis of the similarity order of samples. Our system conducts two optimizations at the same time by adding a constraint of the second optimization to the optimization of the triplet network.

2.2 Notations

A multi-layer neural network (NN) is denoted as F, and its input sample, such as a matrix or vector, is denoted as $\mathbf{x}$. An output of an NN is denoted as $\mathbf{h} \in \mathbb{R}^k$, which we call *representation*. A representation is a projection of a sample into a k-dimensional vector and is also known as embedding. A representation is calculated as $\mathbf{h} = F(\mathbf{x}; \boldsymbol{\theta}_F)$, where $\boldsymbol{\theta}_F$ is the parameters of the NN. For convenience, we use $F(\mathbf{x}; \boldsymbol{\theta}_F)$ and $F(\mathbf{x})$ interchangeably in this paper. Distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, e.g., L2 distance, is denoted as $D(\mathbf{x}_i, \mathbf{x}_j)$, and a loss function is denoted as $\mathcal{L}$.

2.3 Triplet Network

We describe a triplet network [7, 25] applied to our system. It was proposed to optimize a low-dimensional representation given pairs of similar and dissimilar samples. Specifically, a triplet network uses combinations of three samples (triplets) as training data $\mathbf{X} = \{(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)\}$. Samples $\mathbf{x}$, $\mathbf{x}^+$, and $\mathbf{x}^-$ are called reference, positive, and negative samples, respectively. Triplets are selected to satisfy a condition under which reference and positive samples are similar and reference and negative samples are dissimilar. Using these samples, representations are optimized so that the distance between representations of reference and positive samples is smaller than that between reference and negative samples. Since training data $\mathbf{X}$ include many triplets, representations are optimized to preserve the distance relationship between many samples. As a result of the optimization, similarities between samples can be calculated using their representations with distance metrics such as L2 distance.

Figure 3 shows the NN for optimizing representations. Each sample of triples $\mathbf{x}$, $\mathbf{x}^+$, and $\mathbf{x}^-$ is input to a shared NN F, and their representations $\mathbf{h}$, $\mathbf{h}^+$, and $\mathbf{h}^-$ are output, respectively. They are used for calculating their distances. Specifically, the distance between representations of reference and positive samples $d^+ = D(\mathbf{h}, \mathbf{h}^+)$

and that between representations of reference and negative samples $d^- = D(\mathbf{h}, \mathbf{h}^-)$ are calculated. On the basis of these distances, a loss function of the triplet network is defined as follows:

$$\mathcal{L}_{triplet} = \sum_{X} \max\{d^+ + \alpha - d^-, 0\}, \quad (1)$$

where α is a hyperparameter that controls the differences between distances of similar and dissimilar pairs. This loss is called triplet loss. The loss becomes 0 when the distance of a dissimilar pair is larger than the sum of α and distance of a similar pair. Otherwise, the loss becomes positive, which depends on the differences between distances of similar and dissimilar pairs. Representations are optimized on the basis of the differences between distances, but distances of representations are not directly optimized. This optimization satisfies the requirement of our representation learning. Specifically, optimization is conducted on the basis of the similarity order of samples not similarity values of samples.

3 PROPOSED SYSTEM

We describe our system for identifying potential primary indicators by dividing it into the following four steps. In step 1, we collect alerts sent to a SOC (❶,❷,❸ in Fig. 1). In step 2, we extract detection patterns using the collected alerts (❹,❺). In step 3, we optimize representations to calculate the similarities between indicators of different vendors (❻). In step 4, we identify potential primary indicators using known primary indicators and the representations (❼).

3.1 Data Collection

In step 1, we collect alerts sent to a SOC. As mentioned in Section 2.1, customer traffic is analyzed at security appliances, then communication logs and alerts triggered by vendors' detection rules are sent to SIEM (❶). In SIEM, SOC's detection rules trigger alerts if attacks are detected in the communication logs (❷). At the end of this step, alerts with vendor-made and SOC-made indicators are sent to a SOC (❸).

We describe an alert in detail. An alert consists of an indicator and identifier:

$$alert = (indicator, identifier).$$

We assume that an indicator consists of the following four attributes:

$$indicator = (name, direction, action, vendor),$$

where *name* is a detection rule name defined by a vendor or SOC, *direction* is communication direction, *action* represents whether a communication is blocked by an appliance, *vendor* is a vendor name of a security appliance where the alert originates. Communication directions are divided into outbound (from a customer to the Internet), inbound (from the Internet to a customer), and internal (in a customer's network). We use *direction* to define an indicator because its severity greatly differs depending on the communication direction even if the same attack is detected [13, 18]. For example, an indicator regarding network scan is severe when the direction is outbound because it indicates that a possible malware-infected host exists in the customer's network. We also use an action because blocked communication indicates a failure of an attack. We include *vendor* in the definition of an indicator because we aggregate alerts of SOC-made indicators by vendors. This definition of an indicator simplifies the following part of the paper.

We assume that an identifier consists of the following three attributes:

$$identifier = (timestamp, source, destination),$$

where *timestamp* is the time when an alert is raised, and *source*/*destination* are source/destination of a communication. A source and destination are typically represented as IP addresses. In some alerts regarding network scan or denial of service (DoS) attack, multiple IP addresses are aggregated and used as a source and destination.

Table 2: Detection pattern

Type	No.	Features
Timestamp	1	Number of total alerts
	2	Burstiness of alerts
Source	3	Number of unique sources
	4	Ratio of sources represented as multiple IP addresses
	5	Burstiness of unique sources
Destination	6	Number of unique destinations
	7	Ratio of destinations represented as multiple IP addresses
	8	Burstiness of unique destinations

3.2 Detection-Pattern Extraction

In step 2, we calculate a detection pattern of an indicator using the collected alerts. First, we aggregate alerts by indicators (❹) then extract detection patterns (❺).

We aggregate alerts related to the same indicator. As a result, we obtain a set of alerts A_i regarding $indicator_i$:

$$A_i = \{alert_j \mid indicator_j = indicator_i\}.$$

From A_i, we extract a detection pattern representing traits of an indicator. We focus on statistical traits such as burstiness and frequency to distinguish attacks related to indicators. For example, attacks regarding some indicators burst for a short period after the related CVEs are discovered, but attacks regarding other indicators are constantly observed. For another example, a malware sample launching network scan communicates with multiple external hosts, but a malware sample stealing confidential information communicates to a C&C server. We make sets of each identifier's attribute from A_i and extract eight traits; two are extracted from a set of timestamps, three are from a set of sources, and three are from a set of destinations, as shown in Table 2. From a set of timestamps, we extract the number of total alerts and burstiness of alerts. To calculate burstiness [5], we split a collection period of alerts by time windows of size W and count the number of alerts in each window. The burstiness b_W is defined using an average μ_W and variance σ_W^2 of the number of alerts in each window: $b_W = \frac{\sigma_W^2}{\mu_W}$. We use a day as the time window to ensure that a few alerts are included in each time window on average. This is because a meaningful value is not extracted if no alert is included in many time

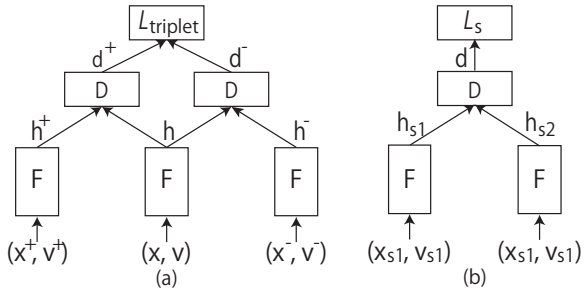

Figure 4: Neural network used in our system

windows. From sets of sources and destinations, we extract the number of unique sources/destinations, ratio of source/destinations represented as multiple IP addresses, and burstiness of the number of unique sources/destinations. Because extracted values of some indicators are very large, we use log scale for elements of detection patterns. In other words, when an extracted value is x_i, the element of the detection pattern is $\log(x_i)$. We extract an 8-dimensional vector as a detection pattern of an indicator. Note that each element of a detection pattern is normalized so that its average and standard deviation are 0 and 1, respectively, when it is input to an NN.

3.3 Representation Learning

In step 3, we optimize representations so that (1) they preserve the similarity order of the detection patterns of the same vendor and (2) the difference in SOC-made indicators resulting from vendors decreases (❻). Since we conduct the optimization from two perspectives, we use two datasets for training. For the first optimization, we use a triplet network. Hence, we prepare triplets as the first dataset by selecting three indicators whose vendors are the same. These indicators are assigned to reference, positive, and negative samples (indicators) on the basis of the similarities of their detection patterns. In this manner, we prepare a sufficient number of triplets as a triplet dataset $\mathbf{X}_{triplet}$. For the second optimization, we prepare a dataset consisting of pairs of SOC-made indicators that are different in vendors but the same in the other attributes (e.g., *name*). We call it the *SOC-made indicator dataset* $\mathbf{X}_S = \{(\mathbf{x}_{S1}, \mathbf{x}_{S2})\}$, where the vendor of $S1$ and $S2$ is different, but the other attributes are the same. We calculate the distance of pairs in the SOC-made indicator dataset and use it as a loss function. On the basis of the above two optimizations, we can obtain representations where the similarity relationship of detection patterns of the same vendor are preserved and the difference in SOC-made indicators resulting from vendors is small. Using these representations, we can calculate the similarities of indicators made by vendors without being affected by the difference resulting from vendors.

We use the two NNs shown in Fig. 4 for representation learning. Since the shared NN F needs to project a detection pattern to a representation in a different manner depending on the vendor, we include a vendor of an indicator in an input of F. More precisely, detection patterns of SOC-made indicators typically differ depending on the vendor. To make their representations similar, projections of detection patterns must differ depending on the vendor. Therefore, we input a vector consisting of a detection pattern $\mathbf{x}$ and a vector representing a vendor $\mathbf{v} \in \mathbb{R}^{N_v}$ to F, where N_v is the number of vendors. We use one-hot encoding for $\mathbf{v}$. Each element of $\mathbf{v}$ corresponds to each vendor. An element corresponding to a vendor of an indicator is 1, and the others are 0. A representation is calculated using a vendor of an indicator as well as a detection pattern: $\mathbf{h} = F(\mathbf{x}, \mathbf{v})$. To calculate the distance of representations, we use L2 distance: $D(\mathbf{h}_1, \mathbf{h}_2) = ||\mathbf{h}_1 - \mathbf{h}_2||_2^2$.

In the optimization of F, we minimize the weighted sum of two loss functions:

$$\min_{\theta_F} \mathcal{L}_{triplet} + \beta \mathcal{L}_S, \tag{2}$$

where $\mathcal{L}_{triplet}$ is a loss function regarding the triplet dataset, $\mathcal{L}_S$ is a loss function regarding the SOC-made indicator dataset, and β is a hyperparameter that controls the effect of $\mathcal{L}_S$. $\mathcal{L}_{triplet}$ is a triple loss defined by (1) and calculated using the NN shown in Fig. 4(a). The loss function of SOC-made indicator dataset $\mathcal{L}_S$ is defined as the distance of the representations of a pair in this dataset. Given $\mathbf{h}_{S1} = F(\mathbf{x}_{S1}, \mathbf{v}_{S1})$, and $\mathbf{h}_{S2} = F(\mathbf{x}_{S2}, \mathbf{v}_{S2})$, $\mathcal{L}_S$ is defined as follows:

$$\mathcal{L}_S = \sum_{\mathbf{X}_S} D(\mathbf{h}_{S1}, \mathbf{h}_{S2}). \tag{3}$$

Note that the hyperparameters of losses can be optimized as mentioned in Section 4.1.

3.4 Potential Primary Indicator Identification

In step 4, we identify potential primary indicators using the representations and levels of indicators determined by a SOC (❼). Potential primary indicators are determined as secondary by a SOC because no relevant incident has occurred yet. Therefore, we identify potential primary indicators from SOC's secondary indicators. For each secondary indicator, we identify the most similar primary indicator among primary indicators of different vendors and use the similarities between them as the score. Secondary indicators that have high scores are identified as potential primary indicators.

Let a set of primary and secondary indicators be PI and SI, one of indicators in the sets be pi_i and si_j. Their detection patterns are denoted as $\mathbf{x}_{pi_i}$ and $\mathbf{x}_{si_j}$, their vendors are denoted as v_{pi_i} and v_{si_j}, and their representations are denoted as $\mathbf{h}_{pi_i}$ and $\mathbf{h}_{si_j}$. The score of the secondary indicator y_{si_j} is defined as follows:

$$y_{si_j} = \max_{pi_i \in PI, v_{pi_i} \neq v_{si_j}} sim(\mathbf{h}_{pi_i}, \mathbf{h}_{si_j}), \tag{4}$$

where sim is a function for calculating a similarity of representations. Specifically, we define sim using the distance of representations as follows: $sim(\mathbf{h}_1, \mathbf{h}_2) = \frac{1}{1+D(\mathbf{h}_1, \mathbf{h}_2)}$. Finally, a set of secondary indicators whose scores are higher than a threshold t are obtained as a set of potential primary indicators PPI:

$$PPI = \{si_i \mid y_{si_i} > t, si_i \in SI\}. \tag{5}$$

We do not apply supervised machine learning to our system because the task of our system is to identify potential primary indicators among secondary ones, not to classify primary and secondary indicators. Specifically, indicators are labeled as primary or secondary by professional analysts in a SOC. These labels of secondary indicators are not accurate and cannot be used for training of supervised learning because potential primary indicators whose labels should be primary may be included in secondary ones. Therefore, we selected an algorithm based on only primary indicators

Table 3: Number of indicators of every vendor

Originated vendor	Vendor-made		SOC-made	
	Alerts	Indicators	Alerts	Indicators
A	1,903,123	1,568	6,437	72
B	1,558,267	1,012	228,061	137
C	751,245	506	158	8
D	192,454	323	13,400	56
E	196,223	181	761	13
F	7,652	7	55,352	85
G	5,835	44	85	3
H	14	2	724	30

that have already been accurately labeled. Note that professional analysts may accidentally label secondary indicators as primary ones, but we expect that such a possibility is very low.

4 EVALUATION

We evaluated the effectiveness of our system using alerts collected in a large-scale SOC and levels of indicators assigned by professional analysts in the SOC. We describe the evaluation setup then present the evaluation results. We conducted the evaluation using an Ubuntu machine with 8-core CPU and 32-GB RAM. All NNs were implemented with Keras [3].

4.1 Evaluation Setup

Dataset. We collected alerts sent to a SOC in Jan. 2019 and prepared triplet and SOC-made indicator datasets. The total number of alerts was 4,919,791, number of unique indicators was 4,047, and number of vendors was 8. Table 3 shows the number of alerts and indicators counted by vendor and type of indicator (i.e., vendor-made or SOC-made). The numbers of alerts and indicators of SOC-made indicators were counted by the vendor of the security appliance where their originated communication logs are observed. All vendors' appliances were used by multiple customers and some customers used multiple vendors' appliances. To protect privacy, no information identifying customers was recorded.

Using these alerts, we prepared two datasets for representation learning. For the triplet dataset, combinations of three indicators of the same vendor were prepared as triplets. When we prepared a triplet, we randomly selected three indicators whose originated vendor was the same. Indicators were selected from both vendor-made and SOC-made indicators. Then, we assigned each indicator to a reference, positive, or negative indicator on the basis of their similarities. More precisely, we made three pairs from the three selected indicators and identified the most similar pair and most dissimilar pair. The indicator included in both pairs was assigned to a reference indicator, the other of the most similar pair was assigned to a positive indicator and the other of the most dissimilar pair was assigned to a negative indicator. To calculate similarities, we used the similarities of detection patterns because vendors of the three indicators were the same and can be compared using detection patterns. To consider both the size and direction of detection patterns in calculating similarities, we used L2 distance and cosine similarity. We selected this metric because norms of some detection patterns are small. Similarities between them cannot be adequately calculated with L2 distance because the L2 distances between vectors of small norms are almost zero. Therefore, we also use cosine similarity to take their direction into account. Both of L2 distance and cosine similarity were normalized so that their maximum value was 1 and minimum value was 0, and their average was used. The similarity of detection patterns $\mathbf{x}_i$ and $\mathbf{x}_j$ is denoted as $sim_x(\mathbf{x}_i, \mathbf{x}_j)$ and defined as follows:

$$sim_x(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}\left(\frac{1}{1 + ||\mathbf{x}_i - \mathbf{x}_j||_2^2} + \frac{\cos(\mathbf{x}_i, \mathbf{x}_j) + 1}{2}\right). \quad (6)$$

In the training of a triplet network, a similar pair is assumed to be apparently more similar than a dissimilar pair. Therefore, we included a triplet in the triplet dataset if the similarity of a similar pair was 0.1 or larger than that of a dissimilar pair. We decided the criterion by referring to a distribution of similarities of detection patterns. In this manner, we prepared a sufficient number of triplets, and the triplet dataset consisted of 1,133,139 triplets. Note that we confirmed that the number of triplets is sufficient because the optimized representations are stable without depending on random seeds for selecting triplets.

We prepared the SOC-made indicator dataset using pairs of SOC-made indicators that differ in vendors but the same in the other attributes. However, all pairs could not be used for the SOC-made indicator dataset because some SOC-made indicators in our collected data depend on alerts from appliances. The rules regarding such indicators detect different traffic depending on the vendor. Hence, vendor-dependent indicators are not suitable for the SOC-made indicator dataset. The difference in detection patterns of vendor-dependent indicators is larger than that of vendor-independent ones because vendor-dependent indicators have a difference resulting from vendors' detection rules as well as traffic observed at vendors' appliances. Therefore, we selected relatively similar pairs for the SOC-made indicator dataset to include vendor-independent pairs. Specifically, we used pairs of SOC-made indicators whose similarities are in the top 20%. In this manner, the SOC-made indicator dataset included 109 pairs of SOC-made indicators. Note that the threshold of similarities was optimized as mentioned in Section 4.1.

Conventional System. To evaluate the effectiveness of our system, we used a baseline system that identifies potential primary indicators on the basis of the similarities of detection patterns sim_x. The baseline does not project detection patterns into representations and calculates scores directly using detection patterns. Given a detection pattern of a secondary indicator $\mathbf{x}_{SI_j}$ and a detection pattern of a primary indicator $\mathbf{x}_{PI_i}$, the score of the secondary indicator is calculated as follows:

$$y_{si_j} = \max_{pi_i \in PI, v_{pi_i} \neq v_{si_j}} sim_x(\mathbf{x}_{pi_i}, \mathbf{x}_{si_j}). \quad (7)$$

We compared our system with this baseline to confirm whether representation learning is necessary for our system to achieve high identification performance. We used only this baseline in our evaluation because, to the best of our knowledge, no system has been proposed for the same purpose in the literature.

Hyperparameter Optimization. We describe how to optimize the hyperparemeters of our system and architecture of the shared NN F. Our system optimizes representations from two perspectives:

(1) preserving the similarity order of the same vendors' indicators and (2) minimizing the difference in SOC-made indicators resulting from vendors. High similarities of pairs in the SOC-made indicator dataset indicate the success of the latter optimization. However, high similarities can be achieved even if representations of all indicators are similar. Apparently, such representations do not preserve the similarity order of the same vendors' indicators. If the similarity order is preserved, the similarities between the same vendor's indicators are relatively smaller than that between pairs in the SOC-made indicator dataset. On the basis of above insight, we selected hyperparameters with which similarities between pairs in the SOC-made indicator dataset were the largest compared with similarities between the same vendor's indicators. The index for selection is defined as follows:

$$\frac{\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in X_S} sim(\mathbf{h}_i,\mathbf{h}_j)}{\sum_{\mathbf{x}_k\in\mathbf{X},\mathbf{x}_l\in\mathbf{X},v_k=v_l} sim(\mathbf{h}_k,\mathbf{h}_l)}, \tag{8}$$

where $\mathbf{X}$ is the set of all detection patterns.

The selected hyperparameters of our system were $\alpha = 1.0$ and $\beta = 1.0$. The architecture of the F consisted of three layers. The first layer was a fully connected layer whose output size was 10 and activation function was rectified linear units (ReLU) with L2 regularization. The second layer was a batch normalization layer [10]. The third layer was a fully connected layer whose output size was 5 and activation function was an ReLU with L2 regularization. The batch size was 512, number of epochs was 50, and optimizer was Adam [12].

4.2 Evaluation Results

We report the results from two evaluations. One was conducted to investigate whether transferring knowledge between vendors is effective for identifying potential primary indicators in detail. We also analyzed representations learned with our system to show the reason of the results. The other evaluation was conducted to investigate whether our system can identify primary indicators discovered in the future.

Effectiveness of Transferring Knowledge Between Vendors. We investigated whether transferring knowledge between vendors is effective for identifying potential primary indicators in detail. To this end, we assumed that primary indicators of a vendor are undiscovered (i.e., potential primary indicators) and evaluated whether our system can identify primary indicators of the vendor using the primary indicators of the other vendors. In this manner, we can evaluate our system using a sufficient number of potential primary indicators for precise analysis. Levels of indicators were assigned by the SOC at the same time of collecting alerts, i.e., in Jan. 2019. When calculating evaluation metrics, we treated primary indicators as positive samples and secondary indicators as negative samples.

We compared our system and the baseline in terms of an area under a receiver operating characteristic (ROC) curve (AUC). Table 4 shows the AUCs of our system and the baseline based on scores calculated by (4) and (7), respectively. No indicator of vendor H was assigned to a primary indicator; thus, the AUCs of vendor H were filled in with NA. The AUCs greatly differed depending on the system and vendor. From the results of the baseline, the AUC of vendor C was the highest, and that of vendor D was the lowest.

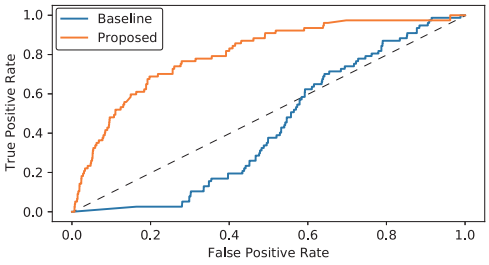

Figure 5: ROC curves of knowledge-transfer evaluation

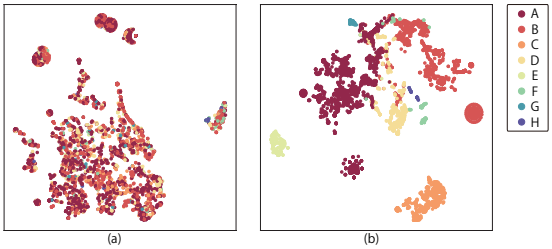

Figure 6: Distributions of (a) detection patterns and (b) representations visualized with t-SNE

From the results of our system, the AUC of vendor A was the highest, and that of vendor E was the lowest. Our system outperformed the baseline in terms of AUCs of all vendors except for vendor C.

We elucidated the reason that AUCs differed depending on vendors. The AUC of vendor D largely was improved by our system, but that of vendor C was not improved. Since the effect of $\mathcal{L}_S$ depends on the number of indicators in the SOC-made indicator dataset, we investigated the number of indicators of each vendor in this dataset and its ratio to all indicators of the vendor (i.e., vendor-made and SOC-made indicators originating from the vendor), as shown in Table 5. The number of SOC-made indicators of vendor D was large, but that of vendor C was small. This is why the effect of $\mathcal{L}_S$ on vendor C was small, and its AUC was not improved by our system. The number of SOC-made indicators of vendors E and G was also small, but the ratio of vendors E and G was larger than that of vendor C. Consequently, the effect of $\mathcal{L}_S$ on vendors E and G was larger than that on vendor C. For this reason, the AUCs of vendors E and G were improved by our system. For more analysis results, please refer to Appendix A.

We compared the overall identification performance of our system with the baseline by drawing ROC curves using all vendors, as shown in Fig. 5. The ROC curve of the baseline was worse than the random prediction (i.e., diagonal line in the figure). This shows that identifying potential primary indicators is very difficult. Our system significantly outperformed the baseline in the range of low FPR. To investigate this, we analyzed the distributions of detection patterns and representations obtained from our system. Figure 6 shows the visualization of the detection patterns and representations by using t-SNE [14]. Almost all regions of detection patterns include multiple vendors, as shown in Fig. 6(a). In contrast, some regions of representations include multiple vendors, but the other regions consist of indicators of one vendor, as shown in Fig. 6(b).

Table 4: AUC of each vendor in knowledge-transfer evaluation

Vendor	A	B	C	D	E	F	G	H
Baseline	0.4053	0.4704	**0.6699**	0.1746	0.3734	0.5798	0.5777	NA
Proposed	**0.8949**	**0.7916**	0.6216	**0.6190**	**0.5863**	**0.7831**	**0.7555**	NA

Table 5: Number of SOC-made indicators in SOC-made indicator dataset

Vendor	#	Ratio	Vendor	#	Ratio
A	36	2.19%	E	7	3.60%
B	43	3.74%	F	28	30.43%
C	7	1.36%	G	3	6.25%
D	26	6.86%	H	13	40.62%

Table 6: Identification performance of potential primary indicator

	FPR=1%		FPR=5%		FPR=10%	
	TP	TPR	TP	TPR	TP	TPR
Baseline	0	0.0%	0	0.0%	0	0.0%
Proposed	2	1.9%	26	24.3%	50	46.7%

Such representations are learned because SOC-made indicators are optimized to be similar to different vendors' SOC-made indicators due to $\mathcal{L}_S$, but other indicators are not forced to be similar to different vendors' indicators. At the same time, we optimized representations so as to preserve the similarity order of indicators of the same vendor. This indirectly forces some indicators that are similar to SOC-made indicators to be similar to indicators of different vendors. Consequently, indicators of different vendors become similar only if they are similar to SOC-made indicators. As a result, our system reduces FPs and outperformed the baseline.

Identifying Primary Indicators Discovered in Future. We evaluated whether our system can identify potential primary indicators that will be discovered as primary indicators by the SOC in the future. We identified potential primary indicators using data and levels in Jan. 2019 and confirmed whether the identified potential primary indicators were determined as primary by the SOC in May 2019. Table 6 shows the numbers of true positives (TPs) and true positive rates (TPRs) at FPRs of 1, 5, and 10%. Since the baseline produced many FPs, it could not identify any potential primary indicator in the range of 1–10% FPRs. On the other hand, our system identified 2, 26, and 50 undiscovered primary indicators at FPRs of 1, 5, and 10%, respectively. Note that the actual TPRs could be higher than those of this evaluation because some indicators could be determined as primary after May 2019.

We analyzed the TPs at an FPR of 5% to investigate what type of attacks they are related to. All TPs are related to attacks to servers such as cross site scripting and SQL injection. Vendors are continuously developing new detection rules to keep up with new attacks and vulnerabilities. As a result, new indicators are being made everyday and multiple indicators are related to the same type of attack. TPs identified with our system are inferred to be newly developed indicators. They were not determined as primary by the SOC because enough incidents related to them did not occur in Jan. 2019. In May 2019, enough incidents had occurred, and they were determined as primary. Since our system identifies potentially primary indicators using alerts not recognized as incidents, we can identify primary indicators four months earlier than a SOC.

5 DISCUSSION

Identification Performance. Our system outperformed the baseline but could not identify potential primary indicators without FPs. Although its accuracy might seem low, our system can contribute to MSS. Specifically, in an actual SOC, analysts spend a large amount of time analyzing alerts of more than 10% of secondary indicators so as not to miss incidents. Therefore, an FPR of 10% is acceptable, and our system achieved sufficiently high identification performance to reduce the time for finding incidents by analyzing alerts of indicators predicted as potential primary prior to alerts of the other indicators compared with that of an actual SOC, where secondary indicators are analyzed in the order of detection time. Moreover, our representations can be used for identifying similar indicators of different vendors. By displaying the similar indicators to analysts in SOCs, they can infer severity and traits of indicators if they are not familiar with the indicators. Therefore, our system can enhance the efficiency of alert analysis in SOCs.

Data Collection. To achieve high identification performance, we need to collect alerts for a certain period, e.g., one month, because our system leverages detection patterns consisting of statistic values. By increasing the collection period, we can improve the identification performance of our system. Conversely, we need longer time to obtain predictions of our system. In the evaluations, we collected alerts for a month; thus, we need a month to obtain reliable prediction after indicators are observed for the first time. Even though we needed a month for prediction, our system identified primary indicators earlier than the SOC. This shows that the collection period of our system does not matter to achieve our goal.

Limitation and Future Work. As shown in Section 4.2, representations of a vendor are not well optimized if the number of indicators in the SOC-made indicator dataset is small. The reason of this is that only a few types of communication logs are sent to SIEM from the security appliances of the vendor. In this case, the number of observed SOC-made indicators is small. Consequently, the number of indicators in the SOC-made indicator dataset is small. Although our system cannot improve identification performance of such vendors, it can improve the performance of many other vendors. In terms of overall identification performance, our system can outperform the baseline. Future work includes designing constraints of representation learning considering more information such as detection rule names, correlation of communication source/destination, and co-occurrence of alerts. The above future work will facilitate associating indicators of different vendors and

mitigate the limitation of our system. As a result, identification performance of our system is expected to improve.

6 RELATED WORK

6.1 Alert Analysis

To enhance the efficiency and detection capabilities of SOCs, some systems have been proposed for analyzing alerts. The most related system is for finding undiscovered incidents on the basis of the relationship between past incidents and indicators [18]. With this system, a bipartite graph is constructed whose vertices are indicators/hosts and edges are alerts regarding indicators and hosts. On this graph, the relationship between incidents and indicators is analyzed with random walk with restart [23]. Random walks start from vertices of hosts where incidents are discovered, then highly related vertices, i.e., indicators and hosts, are identified on the basis of arrival probabilities. Highly related hosts are identified as undiscovered incidents and highly related indicators are identified as primary indicators. This system infers the relationship of indicators using past incidents, but our system infers the relationship of indicators on the basis of aggregated alerts of indicators without using past incidents. Other systems for SOCs are mainly aimed at reducing the labor of analyzing enormous amount of alerts. Since many alerts regarding one incident are sent to a SOC, systems for grouping alerts related to the same attack have been proposed [16, 17, 19, 24]. If alerts are similar in terms of the timestamp, source, and destination, they are aggregated into one meta-alert. From a different perspective, a system was proposed for predicting whether an appliance is useful for finding incidents [2]. None of these systems are for identifying potential primary indicators.

To extract traits of aggregated logs, a system was proposed that extracts log-generation patterns, such as frequency and burstiness, and predicts the status of appliances [11]. By converting aggregated alerts into a numerical vector, we can easily apply statistical analysis such as machine learning. Therefore, we design detection patterns of our system by referring to log-generation patterns [11].

6.2 Metric Learning

To calculate similarities between data, such as images, sentences, and graphs, metric-learning methods have been actively studied. Since deep NNs have achieved significantly high classification performance in many tasks [4, 6], many metric-learning methods are based on NNs, and are called deep metric learning. The basic idea of metric learning was proposed as a Siamese network [1]. Given similar and dissimilar pairs of samples, representations are optimized to increase the similarities of similar pairs and decrease those of dissimilar pairs. A drawback of a Siamese network is the requirement of a carefully prepared dataset to stabilize its training. To overcome this drawback, a triplet network was proposed [7]. As described in Section 2.3, a triplet network accepts triplets consisting of reference, positive, and negative samples and optimizes representations so that similar pairs are more similar than dissimilar pairs. Applications of a triplet network include collaborative filtering [8] and transfer learning [9]. In this study, we used a triplet network so as not to heavily depend on the selection of the triplet dataset.

7 CONCLUSION

To identify potential primary indicators, we propose a system that transforms detection patterns into representations where we can calculate the similarities between indicators of different vendors without being affected by the difference resulting from vendors. On the basis of the representations, we identify potential primary indicators from secondary indicators if secondary indicators are similar to primary indicators of different vendors. In representation learning, we simultaneously conduct the following two optimizations with a triplet network: (1) preserving the similarity order of detection patterns of the same vendor's indicators and (2) minimizing the difference between SOC-made indicators resulting from vendors. We evaluated the effectiveness of our system using 4,919,791 alerts collected from a large-scale SOC for a month. Our system outperformed a conventional system that identifies potential primary indicators on the basis of the similarities between detection patterns. Specifically, our system identified 24.3% more primary indicators undiscovered at the time of data collection at a 5% FPR. Furthermore, by analyzing optimized representations, we show that indicators of different vendors become similar only if they are similar to SOC-made indicators. Consequently, our system reduced FPs and improved identification performance.

REFERENCES

[1] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*. 737–744.

[2] Shang-Tse Chen, Yufei Han, Duen Horng Chau, Christopher Gates, Michael Hart, and Kevin A Roundy. 2017. Predicting Cyber Threats with Virtual Security Products. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. 189–199.

[3] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] U. Fano. 1947. Ionization Yield of Radiations. II. The Fluctuations of the Number of Ions. *Phys. Rev.* 72 (1947), 26–29. Issue 1.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[7] Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. 84–92.

[8] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th international conference on world wide web*. 193–201.

[9] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. 2015. Deep transfer metric learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 325–333.

[10] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[11] Tatsuaki Kimura, Akio Watanabe, Tsuyoshi Toyono, and Keisuke Ishibashi. 2018. Proactive failure detection learning generation patterns of large-scale network logs. *IEICE Transactions on Communications* (2018).

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Ulf Lindqvist and Phillip A Porras. 1999. Detecting computer and network misuse through the production-based expert system toolset (P-BEST). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*. 146–161.

[14] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[15] Alina Oprea, Zhou Li, Robin Norris, and Kevin Bowers. 2018. MADE: Security Analytics for Enterprise Threat Detection. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 124–136.

[16] Roberto Perdisci, Giorgio Giacinto, and Fabio Roli. 2006. Alarm clustering for intrusion detection systems in computer networks. *Engineering Applications of Artificial Intelligence* 19, 4 (2006), 429–438.

[17] Phillip A Porras, Martin W Fong, and Alfonso Valdes. 2002. A mission-impact-based approach to INFOSEC alarm correlation. In *International Workshop on Recent Advances in Intrusion Detection*. 95–114.
[18] Kevin A Roundy, Acar Tamersoy, Michael Spertus, Michael Hart, Daniel Kats, Matteo Dell'Amico, and Robert Scott. 2017. Smoke detector: cross-product intrusion detection with weak indicators. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. 200–211.
[19] Alireza Sadighian, Saman Taghavi Zargar, José M Fernandez, and Antoine Lemay. 2013. Semantic-based context-aware alert fusion for distributed Intrusion Detection Systems. In *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*. 1–6.
[20] SecureWorks, Inc. 2019. Managed Security Services. https://www.secureworks.com/services/managed-security.
[21] Ben Stock, Benjamin Livshits, and Benjamin Zorn. 2015. Kizzle: A signature compiler for exploit kits. In *International Conference on Dependable Systems and Networks*.
[22] Symantec Corporation. 2019. Symantec Managed Security Services. https://www.symantec.com/services/cyber-security-services/managed-security-services.
[23] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Proceedings of the 6th International Conference on Data Mining*. 613–622.
[24] 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.
[25] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1386–1393.

A ANALYSIS RESULTS ON AUC DIFFERENCE

We investigated the reason the AUCs of the baseline differed depending on the vendor. Particularly, the AUC of vendor D was significantly low. We calculated the scores on the basis of the similarities between the indicators of different vendors. If primary indicators of vendor D are significantly distant from the indicators of the other vendors among indicators of vendor D, identifying the primary indicators is quite difficult. We investigated whether it is the reason the AUC of vendor D was low. To this end, for each indicator of a vendor, we identified the most similar indicator of a different vendor and then calculated their similarities. Note that we identified the most similar indicator from all indicators (i.e. both primary and secondary indicators) of different vendors not primary indicators of different vendors to conduct comparison in terms of the distance to distributions of different vendors' indicators. For comparing primary and secondary indicators in terms of the above similarities, we calculate a ratio of an average of the primary indicators' similarities to that of the secondary indicators' similarities. Sets of vendor A's primary and secondary indicators are denoted as PI^A and SI^A, and a set of indicators of vendors other than A is denoted as $I^{\bar{A}}$. The ratio of vendor A is calculated as follows:

$$\frac{\frac{1}{|PI^A|}\sum_{pi_i \in PI^A} \max_{j \in I^{\bar{A}}} sim_x(\mathbf{x}_{pi_i}, \mathbf{x}_j)}{\frac{1}{|SI^A|}\sum_{si_k \in SI^A} \max_{l \in I^{\bar{A}}} sim_x(\mathbf{x}_{si_k}, \mathbf{x}_l)}. \tag{9}$$

Table 7 shows the ratio of every vendor. The vendor D's ratio was the lowest among all vendors' ratios. In other words, the primary indicators of vendor D are particularly distant from the other vendors' distributions compared with the other indicators of vendor D. On the other hand, the ratio of vendor C was the highest. These results indicate that an AUC of vendor D can be improved by adjusting the distribution of this vendor.

Table 7: Ratios of primary indicators' to secondary indicators' averages.

vendor	Detection pattern	Representation
A	0.8766	1.8767
B	0.9249	1.1870
C	1.0508	1.1257
D	0.8154	0.9796
E	0.9737	1.0937
F	0.9029	1.4917
G	0.9969	1.1092
H	NA	NA

Table 8: Similarities of SOC-made indicators

Vendor	Baseline	Proposed	Ratio
A	0.7612	0.2033	0.2671
B	0.7429	0.2184	0.2939
C	0.7023	0.1846	0.2628
D	0.7769	0.2522	0.3246
E	0.6309	0.1607	0.2548
F	0.7647	0.2073	0.2710
G	0.5811	0.2167	0.3729
H	0.7647	0.2023	0.2645

Our system improved the AUC of vendor D. We confirmed whether the improvement was due to change in distributions. Table 7 also shows the ratio calculated using representations as follows:

$$\frac{\frac{1}{|PI^A|}\sum_{pi_i \in PI^A} \max_{j \in I^{\bar{A}}} sim(\mathbf{h}_{pi_i}, \mathbf{h}_j)}{\frac{1}{|SI^A|}\sum_{si_k \in SI^A} \max_{l \in I^{\bar{A}}} sim(\mathbf{h}_{si_k}, \mathbf{h}_l)}. \tag{10}$$

Vendor D's ratio became similar to other vendors' ones. This shows that vendor D's distribution of representations became suitable for identifying potential primary indicators and improved the AUC of vendor D. We further investigated whether such representations were obtained with our representation learning. To this end, we analyzed the similarities between pairs in the SOC-made indicator dataset. Table 8 shows averages of the similarities between detection patterns, those between representations, and ratios of representations' to detection patterns' averages. Larger ratios indicate relatively larger increase in similarities between pairs in the SOC-made indicator dataset. The ratio of vendor D was larger than those of most other vendors. This shows that vendor D's distribution of representations was relatively largely affected by $\mathcal{L}_S$. For this reason, the AUC of vendor D was inferred to be significantly improved by the optimization of our system.