Collaborative Alerts Ranking for Anomaly Detection

Ying Lin* Zhengzhang Chen† Cheng Cao‡ Lu-an Tang† Kai Zhang†
Zhichun Li† Haifeng Chen† Guofei Jiang†

Abstract

Given a large number of low-level heterogeneous categorical alerts from an anomaly detection system, how to characterize complex relationships between different alerts, filter out false positives, and deliver trustworthy rankings and suggestions to end users? This problem is motivated by and generalized from applications in enterprise security and attack scenario reconstruction. While existing techniques focus on either reconstructing abnormal scenarios or filtering out false positive alerts, it can be more advantageous to consider the two perspectives simultaneously in order to improve detection accuracy and better understand anomaly behaviors. In this paper, we propose CAR, a *c*ollaborative *a*lerts *r*anking framework that exploits both temporal and content correlations from heterogeneous categorical alerts. CAR first builds a tree-based model to capture both short-term correlations and long-term dependencies in each alert sequence, which identifies abnormal action sequences. Then, an embedding-based model is employed to learn the content correlations between alerts via their heterogeneous categorical attributes. Finally, by incorporating both temporal and content dependencies into one optimization framework, CAR ranks both alerts and their corresponding alert patterns. Our experiments — using real-world enterprise monitoring data and real attacks launched by professional hackers — show that CAR can accurately identify true positive alerts and successfully reconstruct attack scenarios at the same time.

Keywords: anomaly detection, enterprise security system, alerts ranking, temporal dependency modeling, content dependency modeling.

1 Introduction

In modern information systems or cyber-physical systems, such as social networks, enterprise computer networks, or data centers, a significant challenge is to understand the underlying regularity/irregularity of the vast amount of data that are being generated and collected in a streaming fashion. By identifying underlying anomalies or irregular patterns, critical actionable information can be extracted to facilitate human decision making and mitigate the potential damage.

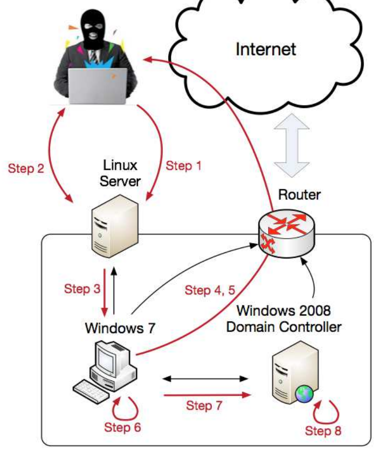


Figure 1: One example of APT attack

Towards this end, a variety of anomaly detection systems (e.g., [3, 9, 12]) have been developed to provide in-depth protection in various information systems. However, in order to fully capture the system status and guarantee sufficient detecting power, a large volume of alert data often needs to be generated from different detectors. On the one hand, this poses significant challenges to end-users in performing effective analysis and initiating timely response; on the other hand, false positive alerts (*i.e.*, alarms that were triggered incorrectly by benign events) can be prevalent, leading to unnecessary system intervention or suspensions. Therefore, it is particularly important to build trustworthy post-processing systems to reduce false positives from massive amount of raw alert data, intelligently correlate and associate alerts from different sources, and uncover interpretable anomaly patterns for understanding the system/user's irregular behaviors.

Building an efficient alerts post-processing system is quite challenging for several reasons. Firstly, most anomaly detection systems focus on low-level interaction between system entities and generate isolated alerts

*University of Washington, Seattle, WA 98195
†NEC Laboratories, America Inc., Princeton, NJ 08540
‡Texas A&M University, College Station, TX 77843

while system/user's abnormal behaviors are high level activities consisting of different low-level events/steps. For example, in an enterprise intrusion detection system, a network attack called Advanced Persistent Threat (APT) as shown in Fig. 1 is composed of a set of stealthy and continuous computer hacking processes, by first attempting to gain a foothold in the environment, then using the compromised systems as the access into the target network, followed by deploying additional tools that help fulfill the attack objective. Thus, it is hard to infer which process/alerts correspond to attack behaviors by looking them independently. Second, the multi-modal and nonlinear correlation between alerts further exacerbates the problem. Alerts can demonstrate complex inter-relations either in the temporal domain (temporal correlation) or in terms of the contents they carry (content correlation). These contents are often represented by heterogeneous categorical attributes that defy existing similarity measurements. Meanwhile, temporally correlated alerts may not occur consecutively but be segmented by false positive ones. Therefore, it is necessary to develop an advanced alerts post-processing system that captures both long-term temporal dependencies and content correlation between alerts.

Existing works on alerts post-processing focus either on modeling causalities between alerts to reconstruct abnormal scenarios [5, 11, 21], or filtering out false positive alerts [15, 16, 20]. Due to aforementioned challenges, it can be more advantageous to consider both factors in order to improve detection accuracy and understand complex abnormal patterns. Filtering out false positive alerts leads to more confident abnormal scenarios being reconstructed. On the other hand, modeling the dependencies between alerts helps to filter out isolated alerts as false positives. Moreover, many existing works rely on users' prior knowledge or training data to achieve good performance, which are seldom available in practical scenarios.

In this paper, we propose CAR, an unsupervised, data-driven *c*ollaborative *a*lerts *r*anking framework that enables one to identify trustworthy alerts and reconstruct abnormal scenarios simultaneously by exploiting both temporal and content correlations between alerts. The proposed method addresses the challenges in three steps. To model the temporal dependency between alerts, we design a tree-based model that compactly represents the alert sequence as well as preserves the long-term temporal dependencies. A set of hierarchical Bayesian priors is used to model the dependencies in a probabilistic way. We further discover a set of alert patterns that correlate individual alerts with incidences of abnormal scenarios. To model the content correlation between alerts, we propose an embedding-based model to learn the latent vector of each entity. Based on the learned embedding vector, a proximity measurement is adopted to quantify the similarity between alerts. Finally, to identify the ground truth alerts and alert patterns corresponding to abnormal behaviors, we formulate the collaborative alerts ranking as an optimization problem by incorporating the temporal and content structures between alerts.

To summarize, in this paper we make the following contributions:

- We identify an important problem (collaborative alerts ranking) in alerts post-processing, and propose a novel alerts post-processing system that integrates the false positive reduction and abnormal scenario reconstruction into one framework.
- We exploit both temporal and content correlations between alerts.
- We develop a prefix tree-based temporal model to discover the underlying abnormal scenarios by modeling the long-term dependencies between alerts.
- We build an embedding-based content model to measure the similarity between heterogeneous categorical alerts.
- We successfully apply our methodology to a real enterprise security system and demonstrate its effectiveness.

2 Problem Formulation

In this section, we introduce some notations and define the problem. Different to traditional anomaly detection methods that focus on numerical data, our goal is to build an unsupervised and data-driven alerts post processing system for heterogeneous categorical alert data, where there is less existing work and more challenging.

Heterogeneous Categorical Alert. A heterogeneous categorical alert $a = (v_1, \ldots, v_m, t)$, detected by an anomaly detection system/sensor, is a suspicious event record that contains m different categorical attributes, and the i^{th} attribute value v_i denotes an entity from the type V_i. Only the time entity t is with continuous value, but it can be chunked into segments of different granularities, such as day and hour, which then can be viewed as categorical entity. In the enterprise security system, an alert is a record involving entities of types such as the user, time, source/destination process and files. In the following, we will call it alert for short.

Alert Pattern. An alert pattern is a subsequence of alerts that may represent the multiple steps/phases of an abnormal behavior. An alert pattern of length L is constructed by L alerts, denoted as $u_{1:L} = \{a_1, \ldots, a_L\}$, ordered by their occurring times, $T(a_1) < \cdots < T(a_L)$.

In this paper, we focus on identifying the alerts and alert patterns that are most likely to indicate the abnormal behaviors.

Problem Statement. Given a set of heterogeneous categorical alerts generated by the same anomaly detection system (with one or multiple detectors deployed) until time T, $\{a_1, \ldots, a_T\}$, the maximum length of an alert pattern L_{max}, and an integer number K, the problem is how to efficiently and effectively find the top K ranked alerts that are most likely to be true positives and the corresponding alert patterns with length less than or equal to L_{max}.

3 Methodology

Intuitively, our method is based on the key observation that the isolated alerts may seldom be true positives, while the alerts whose occurrence can be favorably supported by others, both temporally and contextually, are more likely to be important and true positive alerts. Thus, our goal is to provide a framework for automatic exploration of both temporal and content-based dependency between alerts and identification of the trustworthy alerts and their corresponding abnormal patterns.

We start by exploiting the temporal structure in alert sequences. A tree-based model is built to recover the temporal dependencies between these alerts and simultaneously extract a set of alert patterns. In the meantime, we embed the categorical entities into a latent space to model the content similarities between the alerts. Based on our assumption that isolated alerts may seldom be true positives, we compute ranking scores of the alerts by maximizing consensus among the temporal and content structures, and simultaneously reconstruct the abnormal scenarios from the trustworthy alerts. Detailed description of each step is presented in the following paragraphs.

3.1 Temporal Dependency Modeling We start by symbolizing the raw alert sequence under an appropriate granularity. Specifically, we will consider only a set of important entities[1], such as the source and destination information to represent each alert, and enumerate all possible alerts in the symbol set $\Sigma = \{s_i\}$.

In the literature, random walk [7], Markov model [21], and n-gram model [2] are popular techniques to characterize temporal dependencies using distributions of future symbols conditioned on a finite history of symbols. However, they only capture finite, or short-term dependencies between alerts while a real system/user's abnormal behavior may include a number of separate steps residing on a long temporal span. To measure the long-term dependency, we model each alert, s_i, using a predictive distribution that is conditional on all previous alerts, $u_{1:i-1} = \{s_1 \ldots, s_{i-1}\}$. Thus, the joint distribution of each pattern can be estimated as:

$$p(u_{1:L}) = \prod_{i=1}^{L} p(s_i | u_{1:i-1}) \tag{3.1}$$

This is a non-Markovian model since the number of preceding alerts varies. If the prediction of next alert only relates to the values of n preceding alerts, the problem in Equation 3.1 can be approximated by an n^{th} order Markov model, i.e. $P(s_i|u_{1:i-1}) = P(s_i|u_{i-n:i-1})$.

Denote the predictive distribution conditioned on preceding alerts $u_{1:i-1}$ as $G_{[u_{1:i-1}]}$, it can be represented as a probability vector with latent variables, *i.e.*, $G_{[u_{1:i-1}]}(s) = p(s_i = s|u)$, $\forall s \in \Sigma$. To obtain the joint distribution in Equation 3.1, we need to estimate all distributions with respect to the prefixes of u, *i.e.*, $\{G_{[u']}\}_{u' \in \{u_{1:i} | 1 \leqslant i \leqslant L\}}$. Estimating the probability vectors independently relies on adequate training observations that represent the underlying distribution. However, the abnormal scenarios can be rare in reality, which results in few observations of alert pattern. This is insufficient for estimating the probability vector. Thus, we use a **prefix tree** to represent alert sequence [19] that hierarchically ties together the predictive probabilities and build a **hierarchical Bayesian model** to address the problem of insufficient training data, which uses observations that occur in very long patterns to recursively inform the estimation of the predictive probabilities for related shorter patterns and vice versa [18].

3.1.1 Model Learning In the prefix tree, each node represents an alert pattern and its probability vector. By recursively adding the prefix alert to alert pattern and marginalizing out the non-brunching interior nodes, the prefix tree can be directly constructed from an input sequence in linear time and space [19]. A marginalized prefix tree built from an example sequence $\mathbf{s} = ABCBC$ is shown in Fig. 2. The resulting prefix tree retains precisely the nodes (patterns) of interest, and eliminates all non-branching nodes.

Then we place a set of hierarchical Pitman-Yor processes to approximate the probability vectors in prefix tree [18]. Specifically, each predictive distribution is modeled as a Pitman-Yor process as follows:

$$G_{[u]} | d_u, c_u, G_{[\pi(u)]} \sim PY(d_u, c_u, G_{[\pi(u)]}) \tag{3.2}$$

where d_u and c_u represents the discount and strength parameters of the Pitman-Yor process. $G_{[\pi(u)]}$ denotes

[1] Please note that other entity information will still be considered in the content dependency modeling part (see Section 3.2)

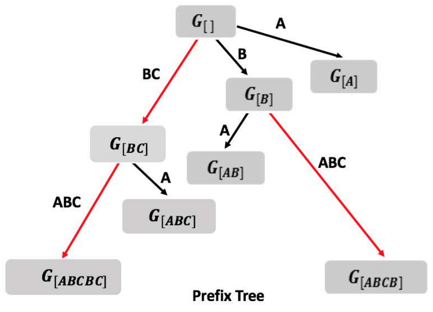


Figure 2: An example to build marginalized prefix tree on sequence $ABCBC$. The marginalized edges are marked in red

the prior distribution of $G_{[u]}$. The prior distribution is also a Pitman-Yor process with respect to the suffix of u, $\pi(u)$. The suffix consists of all but the earliest word in u. This choice of the prior structure reflects our belief that among all preceding alerts, those appearing closer to the current alert are more important in capturing the dependencies with the current alert.

To learn the parameters in this model, Gibbs sampling can be used.

3.1.2 Alerts Patterns Searching We are also interested in finding alert patterns corresponding to the abnormal scenarios by their temporal dependencies. In practice, it can be time consuming to search for the possible alert patterns in a sequence, in particular considering long-term dependencies. To solve the problem, we propose to use the breadth-first search algorithm to travel over the prefix tree in order to find alert patterns, and use the joint distribution in Equation 3.1 to measure the dependency within each alert pattern. Using the conditional probability predicted by the hierarchical Bayesian model, the temporal dependency of each alert pattern can be further expressed as:

$$p(u_{1:L}) = \prod_{i=1}^{L} G_{[u_{1:i-1}]}(s_i) \tag{3.3}$$

The stronger temporal dependency a pattern has, the more likely that alerts inside this pattern collectively point to the system or user's abnormal behaviors. Since the length of the pattern can grow very large as the height of prefix tree increases, we use a parameter L_{max} to control the maximal length of patterns. Meanwhile, the minimal length of patterns is set at 3 since real abnormal scenarios typically have more than two steps to be accomplished. Therefore, we obtain a set of alert patterns and their degree of dependencies from the temporal model, denoted as $U = \{(u, p_u) | 3 \leqslant |u| \leqslant L_{max}\}$

3.2 Content Dependency Modeling In addition to temporal information, an alert also contains rich content information. It consists of heterogeneous categorical entities — such as host, system, agent, source, and destination. In this section, we explore the content dependency of these low-level entities within each alert.

The content similarity among a series of alerts may provide hints of grouping them into the same high-level anomalies. For example, consider an attack in a computer system deleting a directory that contains two files: "*/home/document/password.txt*" and "*/home/document/info.txt*". This attack will generate two alerts which share the same source (*i.e.*, the process "*rm*") and destination (*i.e.*, the folder "*/home/document/*"). And yet another deletion over "*/home/user/info.txt*" will not be covered. Apparently, compared to the third alert, the first two are quite close in terms of the content.

How to measure the content similarity between alerts via their entities? Every alert consists of a set of entities. Their co-occurrence in one alert is one of the most commonly considered connections [1, 17]. The co-occurrence may exist within the same type of entities, e.g., a process forks another process in a computer system. It can also happen between different types of entities, e.g., a process reads a file. However, entities — as the categorical attributes of an alert — do not have any intrinsic ordering among themselves, making most popular numerical learning algorithms hard to directly leverage their co-occurrences [22].

Recent advances in distributed text representation learning have afforded effective text embedding methods that capture syntactic and semantic word relationships. Mikolov *et al.* developed an unsupervised approach of learning word embeddings by exploiting word co-occurrences in a corpus [13]. Inspired by their breakthrough, we embed all entities into a common latent space where the co-occurrences between entities are preserved. Then, the closeness between entities can be easily calculated in the space by popular metrics such as Euclidean distances and cosine similarity.

In concrete, given an alert $a = (v_1, v_2, \ldots, v_m, t)$, we model the entity co-occurrences in a by the conditional probability $p(v_i|v_j;\theta)$ for each pair of categorical entities in a, using the following Softmax function:

$$\prod_{v_i,v_j \in a} p(v_i|v_j;\theta) = \prod_{v_i,v_j \in a} \frac{\exp(w_{v_i v_j} \cdot z_{v_i} \cdot z_{v_j})}{\sum_{v'_i \in V} \exp(w_{v_i v_j} \cdot z_{v'_i} \cdot z_{v_j})} \tag{3.4}$$

where z_{v_i} and z_{v_i} are the embedding vectors for v_i and v_j respectively, and V is the set of all available entities. $w_{v_i v_j}$ is the weight for pairwise interaction between v_i's and v_j's entity types, and it is non-negative constrained.

The parameters θ are z_{v_i} and $w_{v_i v_j}$ for each $v_i \in V$.

Given all the observed alerts $\{a_1, \dots, a_T\}$, we would like to set the parameters such that Equation 3.4 is maximized. This optimization model has been historically known very expensive to solve, due to the denominator of Equation 3.4 summing over all entities of V. Thus, we follow the existing idea of negative sampling to address this challenge. In general, to avoid dealing with too many entities in V, we only update a sample of them. We surely should keep all observed co-occurred entities in our data, and we need to artificially sample a few "noisy entity pairs" — they are not supposed to co-occur so their content similarities should be low.

In the end, after taking the logarithm and negative sampling, we aim to minimize the following objective:

$$\begin{aligned}(3.5)\quad \min_{\mathbf{v},\mathbf{w}} & - \textstyle\sum_{(v_i,v_j)\in D} \log(w_{v_i v_j} \sigma(z_{v_i} \cdot z_{v_j})) \\ & - \textstyle\sum_{(v_i',v_j')\in D'} \log(w_{v_i' v_j'} \sigma(-z_{v_i'} \cdot z_{v_j'}))\end{aligned}$$

where σ is the sigmoid function, and D is the collection of entity co-occurrences observed in our data. D' is the set of negative samples constructed by certain sampling scheme. Concretely, for each co-occurrence $(v_i, v_j) \in D$, we sample k noises $(v_{i_1}', v_j), (v_{i_2}', v_j), ..., (v_{i_k}', v_j)$ where v_i' is drawn according to a noise distribution. Without a rule of thumb on guiding negative sampling, we empirically test several and find the best one when sampling v_i in a probability inversely proportional to the frequency of co-occurrence with v_j. Then, the standard mini-batch gradient descent algorithm is used to solve the Equation 3.5.

Next, given the learned embedding vector of each entity, we measure the pairwise alert similarity via the weighted combination of cosine similarities between their entities. Suppose we have two alerts $a_i = (v_{i_1}, v_{i_2}, \dots, v_{i_m})$ and $a_j = (v_{j_1}, v_{j_2}, \dots, v_{j_m})$. Their content similarity, denoted by S_{ij}, can be calculated as:

$$S_{ij} = \sum_{v_{i_k}\in a_i, v_{j_k}\in a_j} w_{v_{i_k}, v_{j_k}} \langle z_{v_{i_k}}, z_{v_{j_k}} \rangle$$

where $z_{v_{i_k}}$ and $z_{v_{j_k}}$ are embedding vectors of entities v_{i_k} and v_{j_k}, respectively, and $w_{v_{i_k} v_{j_k}}$ is the learned weight between the two entity types of v_{i_k} and v_{j_k}.

3.3 Collaborative Alerts Ranking In this section, we propose a ranking algorithm that simultaneously computes the confidence scores for both individual alert and extracted alert patterns, by maximizing their consensus in terms of both temporal structures and content dependencies.

Suppose we are given a sequence of alerts a_i, $i = 1, \dots, T$, whose confidence scores are denoted by τ_i's. The pairwise similarities between alerts, as reflected by their content information, is denoted by a similarity matrix $S \in \mathbb{R}^{T\times T}$. On the other hand, we also extract L alert patterns, u_l, $l = 1, \dots, L$ based on the temporal model, whose confidence scores are denoted μ_l's. Note that for each alert pattern u_l, we have a number of alerts associated with it, whose strength of temporal dependency is given as p_l provided by the temporal model. The relation between T alerts and L alert patterns are represented by a zero/one coincidence matrix $F \in \mathbb{R}^{T\times L}$. Each element $F_{il} = 1$ indicates that alert a_i is linked to alert pattern u_l.

Our goal is to simultaneously compute the alerts' confidence τ_i's and alert-patterns' confidence μ_l's, given the alerts' similarity S, confidence matrix F, and the temporal dependency strength within patterns, p_l's. We have the following requirements to impose in the ranking process that maximizes the consensus between temporal and content dependencies: 1) Those alert patterns with higher degrees of temporal dependency strength, p_l, are more likely to be true positives; namely those alert patterns composed of temporally correlated alerts are more relevant patterns; 2) The confidence score of each alert pattern should be close to the (average) confidence scores of the individual alerts associated with it; 3) Alerts with similar content information tend to have similar confidence scores. Given these considerations, we propose the following optimization problem

$$\begin{aligned} & \min_{\tau,\mu} - \textstyle\sum_l p_l \mu_l + \frac{\lambda_1}{2} \sum_{i,l} F_{il} (\mu_l - \tau_i)^2 \\ & \qquad + \tfrac{\lambda_2}{2} \textstyle\sum_{i,j} S_{ij} (\tau_i - \tau_j)^2 \\ (3.6) \quad & \quad s.t. \ \textstyle\sum_i \tau_i \leqslant K, \ \mu_l \geqslant 0, \ 0 \leqslant \tau_i \leqslant 1. \end{aligned}$$

Here, the first term in the objective function is used to correlate μ_l's to p_l's (requirement-1). The third term is used to enforcement the smoothness of τ_i's based on the content-level similarity S (requirement-3). These two terms independently impose constraints on the alert-patterns' confidence and alerts' confidence. They are connected with each other by the second term in the objective, which states that the alert-pattern's confidence μ_l should be bounded close to those individual alerts' confidence τ_i's associated with the pattern. By doing this, each alert and alert-pattern will be given a confidence score based on a global ranking that reaches the consensus as specified by the temporal structures and content similarities. The confidence scores are non-negative, and K is a pre-defined integer that roughly controls the number of alerts with non-zero scores.

We further write the optimization problem in Equation 3.6 in a matrix form. Define diagonal matrices $D_{F_c} \in \mathbb{R}^{L\times L}$ and $D_{F_r} \in \mathbb{R}^{T\times T}$ whose diagonal entries are $D_{F_c}(l,l) = \lambda_1 \sum_i F_{il}$, $D_{F_r}(i,i) = \lambda_1 \sum_l F_{il}$. Let L_S be the Laplacian matrix of S. Let $\mathbf{x} =$

$[\tau_1, \ldots, \tau_T, \mu_1, \ldots, \mu_L]^T \in \mathbb{R}^{(T+L)\times 1}$, then the optimization problem in Equation 3.6 can be simply written as

$$\begin{aligned} &\min_{\mathbf{x}} \mathbf{q}^T\mathbf{x} + \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} \\ &s.t.\ \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \ \ \mathbf{x} \geqslant \mathbf{0} \end{aligned} \tag{3.7}$$

where $\mathbf{q} = [\mathbf{0}, p_1, \ldots, p_L] \in \mathbb{R}^{1\times(T+L)}$, $\mathbf{Q} = \left[\begin{smallmatrix} \lambda_1 D_{F_r} + \lambda_2 L_S & -\lambda_1 F \\ -\lambda_1 F & \lambda_1 D_{F_c} \end{smallmatrix}\right] \in \mathbb{R}^{(T+L)\times(T+L)}$, $\mathbf{A} = \left[\begin{smallmatrix} \mathbf{I}_{T\times T} & \mathbf{0}_{L\times L} \\ \mathbf{1}_{1\times T} & \mathbf{0}_{1\times L} \end{smallmatrix}\right] \in \mathbb{R}^{(T+1)\times(T+L)}$, $\mathbf{b} = [1, \ldots, 1, K]^T \in \mathbb{R}^{(T+1)\times 1}$. This is a standard quadratic programming problem with global optimal solution.

4 Experiments

4.1 Experiment Setup We apply CAR to a real-world enterprise intrusion detection system. The alert data was generated by 10 intrusion detectors that monitored both host-level and network-level events on 173 machines. In total, there are $3,322$ alerts collected within one month. Among them, there are only 41 true positive alerts corresponding to three different types of attacks.

Each alert is a computer system event recording an interaction between a pair of system entities. The types of system entities involve *processes*, *files*, and *Internet sockets* (INETSockets). Three types of interactions are considered in this experiment including: (1) *a process accesses a file;* (2) *a process forks another process;* (3) *a process forks an INETSocket.* Each alert is also associated with a set of descriptive, categorical attributes and the time stamp.

We implement CAR in Java and run it on a PC with a 2.5GHz CPU and 8GB RAM. By default, we set the *maximum alert pattern length* $L_{max} = 7$, and pre-defined integer $K = 50$.

Attack Description. Our attack testbed is built by professional Russian hackers. There are 3 different types of attacks with 5, 6 and 7 attack steps, respectively. For each type of attacks, the hackers tried 3 times in three different days, which resulted in a total of 9 intrusion attacks in the data. The three types of attacks are listed in the following:

- *Emulating Enterprise Environment (EEE)*: This attack consists of seven steps. The hacker first utilizes the IRC vulnerability to create telnet processes and reverse connect to attacker host. The telnet process is used to create malware binary and open reverse connection. The malware process downloads malware binary (trojan.exe). Then the trojan.exe is created and used to connect back to attacker host. DLL is injected by running process notepad.exe and creates connection back to the hacker. The hacker uses mimikatz and kiwi to perform memory operation inside meterpreter context. Finally, malwares PwDump7.exe and wce.exe are copied and run on the target host.

- *Diversifying Attack Vectors (DAV)*: There are six steps in this attack. The hacker first writes malicous php file by http connection, then downloads the malware process (trojan.exe), and connects back to attacker host. The process notepad.exe is run to perform DLL injection. Attacker further uses mimikatz and kiwi to perform memory operation inside meterpreter context. Finally, it copies and runs PwDump7.exe and wce.exe on target host.

- *Domain Controller Penetration (DCP)*: This attack includes five steps to penetrate domain controller. First, hacker sends email message with a malicious word document, and the document writes malware python32.exe, which opens up connection to the attacker host. The hacker then runs notepad.exe and performs reflective DLL injection to gain privilege. He/she further transfers password enumerator and runs process gsecdump-v2b5.exe to get all user credentials. Finally, SQL server address is probed to connect and dump database into the attacker host.

Baseline Models. We compare CAR with the following state-of-the-art methods:

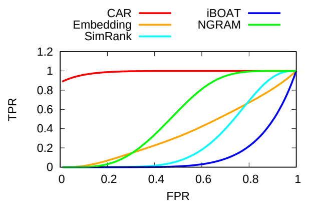
- *NGRAM* [2]: This method has been widely used for the identification of attacks and malicious software. It builds the profiles of normal alerts from the training data, and labels those alerts in the testing data that do not appear in the normal profiles as abnormal ones. In our experiments, we use the first 40% normal alerts as the training set for *NGRAM*.

- iBOAT [4]: This association rule based method has shown its effectiveness in temporal structure discovery [4]. It defines the true positive alerts as those whose corresponding temporal patterns have low confidence score.

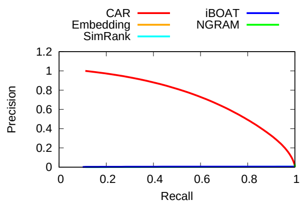
- Embedding: This is our proposed method (see Section 3.2) to learn the pairwise content-level similarities between alerts. We use values in the dominant eigenvectors of similarity matrices to determine the anomaly degree of each alert.

- SimRank [7]: *SimRank* is a widely used graph-based model to measure the similarities between categorical data. It aggregates entities into a graph and measures the pairwise similarities by applying random walk algorithm. Based on the similarity matrix, we use the same way as in Embedding to measure the anomaly degree of each alert.

Evaluation Metrics. Since most methods listed above produce abnormal scores instead of binary labels, metrics for binary labels such as accuracy are not suitable for measuring the performance. We adopt ROC curves (Receiver Operating Characteristic curves) and PRC (Precision Recall curves) for evaluation. Both curves reflect the quality of predicted scores according to their true labels at different threshold levels. To get a quantitative measurements, the AUC (area under curve) of both ROC and PRC are computed.



(a) ROC



(b) PRC

Figure 3: ROC and PRC curves of different alerts ranking methods

4.2 Results for Detecting True Positive Alerts

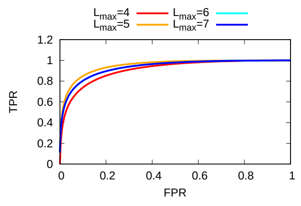
The ROC and PRC curves presented in Fig. 3. demonstrate the effectiveness of proposed method in detecting true positive alerts. As shown in the ROC curve, CAR is able to detect most of the true positive alerts when the false positive rate is controlled at a low level. The advantage of the proposed method is further quantified by the AUC values summarized in Table 1. In the PRC curve, due to the effect of 2 (out of 41) true positives ranked relatively low by the proposed method, the precision will diminish to near 0 when all true positives are required to be detected. In addition, precision rates of the baseline methods keep in low levels because these models can only filter out up to 50% false positive alerts in their best cases. It can also be observed that: embedding the categorical entities into latent space better captures the similarities between alerts, as demonstrated by the improved ROC and AUC of *Embedding* method over *SimRank* method; the *NGRAM* model achieves better accuracy compared to the other baseline models by using labeled training data; the *iBOAT* method does not perform well because it is not suited for capturing the long-term dependency.

In practice, it is unrealistic to provide accurate estimations of the length of alert patterns, neither the number of true positive alerts. Thus, we evaluate the effect of these parameters on the model performance. We vary the maximum pattern length L_{max} from 4 to 7 and set the pre-defined integer K at 1, 10, 50, and 100. The ROC curves are plotted in Fig. 4

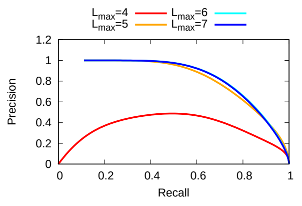
Table 1: AUC of different alerts ranking methods

Method	CAR	Embedding	SimRank	NGRAM	iBOAT
ROC	**0.998**	0.353	0.258	0.440	0.140
PRC	**0.728**	0.003	0.003	0.440	0.140

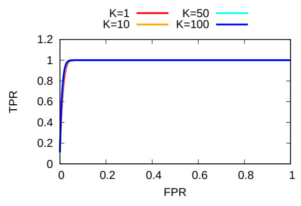
and the AUC values are summarized in Table 2. We observe that the pre-defined integer K doesn't affect the CAR performance, because it only controls the sparsity of alert scores but not the ranking among top alerts. Meanwhile, incorporating the longer patterns will benefit the alerts ranking result as demonstrated by the improvements of AUC when L_{max} increases. But the effect is quite negligible especially when $L_{max}5$. This is because the number of long alert patterns corresponding to attack scenarios is quite small.



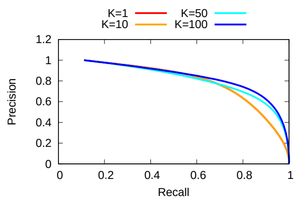
(a) ROC w.r.t. L



(b) PRC w.r.t. L



(c) ROC w.r.t. K



(d) PRC w.r.t. K

Figure 4: ROC and PRC curves w.r.t. the maximum pattern length L_{max} and the pre-defined integer K

Table 3: High-ranked alert pattern related to attack scenarios

CAR rank	Alert pattern	*Temporal* rank
1	EEE-5,EEE-5, EEE-7	1
2	EEE-1, EEE-2, EEE-3	2
3	EEE-4, EEE-5, EEE-7	3
4	DCP-2, DCP-3, DCP-4DCP-5	7
5	DAV-2, DAV-3, DAV-4, DAV-6	14
6	EEE-1, EEE-2, EEE-3, EEE-4	41

4.3 Attack Scenario Related Patterns

To evaluate the CAR method in terms of generating meaningful alert patterns, we compare it with the *Temporal* model

Table 2: AUC w.r.t. different parameter settings

Parameter	$K = 50$				L_{max}=7			
Value	L_{max}=4	L_{max}=5	L_{max}=6	L_{max}=7	K=1	K=10	K=50	K=100
ROC	0.995	**0.999**	0.998	0.998	0.996	0.998	0.998	**0.999**
PRC	0.401	0.829	**0.843**	0.840	0.760	0.760	0.802	**0.821**

presented in Section 3.1. Table 3 lists the top six alert patterns generated by CAR and compares their ranks in the *Temporal* model. These six patterns are all highly associated with the three types of attacks. We use the attack name plus the corresponding step number to represent the alert in each pattern. For instance, the alert *EEE-5* in the first pattern represents the fifth step of *Emulating Enterprise Environment* attack. Note that there are two different alerts associated with the step 5 of *EEE* attack. Generally, Table 3 shows that by exploiting the content correlation between alerts, CAR greatly boosts the rank of the attack-related patterns in *Temporal* model. More specifically, four patterns are detected for the *EEE* attack. The step 6 (the hacker looks up domain controller admin credential cached on memory) is not discovered, however, due to the lack of memory monitoring detector in our intrusion detection system. A longer pattern that captures the whole *EEE* attack scenario is also detected by CAR, with the highest rank in patterns with lengths larger than 4, but with a relatively low rank of 61 in all patterns. That is because longer patterns tend to have lower correlation scores. For the *DAV* attack, CAR successfully reconstructed four steps, with only step 1 and 6 missed because no corresponding alerts were generated by the detectors. For DCP attack, CAR also recovered four out of five steps, where the first step was actually missed by the raw detectors.

5 Related Work

5.1 Alert Post-processing Data mining techniques have been widely used in alerts post-processing [20, 8, 14]. Based on the level of prior knowledge needed for alerts post-processing, the models can be classified into supervised learning models, which rely on the labeled training alerts, and unsupervised learning models, which aim to detect unknown anomalies.

Alerts classification models proposed in [16, 10] use classifiers to distinguish true alerts with false positives. These methods can not capture alerts that are not observed in the training phase. However, in practice it is usually unrealistic to obtain plenty of training data for anomaly detection tasks.

In order to compensate for the lack of prior knowledge, alerts correlation models are developed to identify the underlying anomalies. for example, clustering techniques can group similar events together, which have proven to be highly effective in reducing large number of alerts produced for a single malicious activity [15]. Valdes and Hyvarinen proposed a probabilistic model that measures the overall similarity between alerts as a weighted combination of their similarities on the respective attributes [20]. However, the specification of weights relies on experts' knowledge. Holmann and Sick proposed an online intrusion alerts aggregation system [6], which measures the similarity between alerts using a finite mixture distribution. They characterized the continuous and categorical attributes using multinormal and multinomial distributions, respectively, and aggregated alerts into groups by solving the finite mixture distribution using EM algorithm. However, the effectiveness of this method depends on the distribution assumptions. A hierarchical clustering technique was proposed by Julisch in [8] for grouping the alerts that share common root causes. But the number of clusters need to be predetermined. In addition, the main weakness of clustering models is they do not exploit the temporal dependency, such as the causality, between alerts.

5.2 Alert Pattern Discovery Alert pattern discovery aims to reconstruct multi-step abnormal scenarios (e.g., a cyber attack) by looking at the causality between alerts, which can be used to provide a high-level view of the actual attack. Association rule mining is used to investigate the multi-step alerts reconstruction in [11]. Frequent sequence patterns are mined over alert bursts and abnormal scenario models are constructed based on pattern matching and correlativity calculation. However, the risk of missing relevant alerts and the effects of alert latency are not considered in this work. Machine learning methods such as n-gram analysis [5], Hidden Markov Model (HMM) [21], and Bayesian network (BN) [14] are commonly used to reconstruct complex scenarios by training the models under known abnormal scenarios or true positive alerts. However, the effectiveness of these methods relies on the prior knowledge to build the training data and the efficiency relies on the data preprocessing to remove the false positives.

Different from the existing methods, CAR exhibits desirable properties that an alerts post-processing system requires: it is (1) unsupervised: no training data is required and no class labels need to be provided; (2) data-driven: no prior knowledge is needed for preprocessing data or defining abnormal patterns; and (3) unified framework: CAR discovers the top-k alerts and the corresponding alert patterns simultaneously, by modeling both temporal and content dependencies between alerts.

6 Conclusion

In this paper, we addressed an important and challenging problem of alerts post-processing on massive heterogeneous categorical alert data. We proposed CAR, an unsupervised data-driven collaborative alerts ranking algorithm for detecting the trustworthy alerts and their associated abnormal patterns simultaneously. When tested on three enterprise cyber attack scenarios, CAR demonstrated the superiority in terms of various skill and robustness metrics, including $55-85\%$ AUC improvement as well as interpretability of detected alert patterns. An interesting direction for further exploration would be incorporating raw event data with the alert data in abnormal scenario reconstruction.

References

[1] S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. *red*, 30(2):3, 2008.

[2] M. Caselli, E. Zambon, and F. Kargl. Sequence-aware intrusion detection in industrial control systems. In *Proceedings of the Workshop on Cyber-Physical System Security*, pages 13–24, 2015.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[4] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang. iboat: Isolation-based online anomalous trajectory detection. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):806–818, 2013.

[5] O. Dain and R. K. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *Proceedings of the 2001 ACM workshop on Data Mining for Security Applications*, volume 13, 2001.

[6] A. Hofmann and B. Sick. Online intrusion alert aggregation with generative data stream modeling. *IEEE Transactions on Dependable and Secure Computing*, 8(2):282–294, 2011.

[7] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543. ACM, 2002.

[8] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 366–375. ACM, 2002.

[9] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *SDM*, pages 25–36. SIAM, 2003.

[10] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security (TiSSEC)*, 3(4):227–261, 2000.

[11] W. Li, L. Zhi-tang, L. Dong, and L. Jie. Attack scenario construction with a new sequential mining technique. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing,(SNPD2007)*, volume 1, pages 872–877. IEEE, 2007.

[12] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal outlier detection in vehicle traffic data. In *2009 IEEE 25th International Conference on Data Engineering*, pages 1319–1322. IEEE, 2009.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[14] X. Qin and W. Lee. Statistical causality analysis of infosec alert data. In *International Workshop on Recent Advances in Intrusion Detection*, pages 73–93. Springer, 2003.

[15] R. Sadoddin and A. Ghorbani. Alert correlation survey: framework and techniques. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, page 37. ACM, 2006.

[16] M. Shimamura and K. Kono. Using attack information to reduce false positives in network ids. In *11th IEEE Symposium on Computers and Communications (ISCC'06)*, pages 386–393. IEEE, 2006.

[17] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[18] Y. W. Teh. A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 985–992. Association for Computational Linguistics, 2006.

[19] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[20] A. Valdes and K. Skinner. Probabilistic alert correlation. In *International Workshop on Recent Advances in Intrusion Detection*, pages 54–68. Springer, 2001.

[21] X. Zan, F. Gao, J. Han, and Y. Sun. A hidden markov model based framework for tracking and predicting of attack intention. In *2009 International Conference on Multimedia Information Networking and Security*, volume 2, pages 498–501. IEEE, 2009.

[22] K. Zhang, Q. Wang, Z. Chen, I. Marsic, V. Kumar, G. Jiang, and J. Zhang. From categorical to numerical: Multiple transitive distance learning and embedding. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 46–54, 2015.