ConLBS: An Attack Investigation Approach by Contrastive Learning with Behavior Sequence

Jiawei Li , Ru Zhang * , Jianyi Liu

Posted Date: 1 November 2023

doi: 10.20944/preprints202311.0019.v1

Keywords: attack investigation; contrastive learning; behavior sequence; audit logs

Article

ConLBS: An Attack Investigation Approach by Contrastive Learning with Behavior Sequence

Jiawei Li, Ru Zhang * and Jianyi Liu

School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; lijw960502@bupt.edu.cn (Jiawei Li), liujy@bupt.edu.cn (Jianyi Liu)

* Correspondence: zhangru@bupt.edu.cn

Abstract: Attack investigation is an important research field in forensics analysis. Many existing supervised attack investigation methods rely on well-labeled data for effective training. While the unsupervised approach based on BERT can mitigate the issues, the high degree of similarity between certain real-world attack and normal behaviors makes it challenging to accurately identify disguised attacks. This paper proposes ConLBS, an attack investigation approach that combines the contrastive learning framework and multi-layer Transformer network to realize the classification of behavior sequences. Specifically, ConLBS constructs behavior sequences describing behavior patterns from audit logs, and a novel lemmatization strategy is proposed to map the semantics to the attack pattern layer. Four different augmentation strategies are explored to enhance the differentiation between attack and normal behavior sequences. Moreover, ConLBS can perform unsupervised representation learning on unlabeled sequences, and can be trained either supervised or unsupervised depending on the availability of labeled data. The performance of ConLBS is evaluated in two public datasets. The results show that ConLBS can effectively identify at-tack behavior sequences in the cases of unlabeled data or less labeled data to realize attack investigation, and achieve the superior effectiveness compared to existing methods and models.

Keywords: attack investigation; contrastive learning; behavior sequence; audit logs

1. Introduction

Enterprises face threats from covert and persistent multi-step attacks, such as Advanced Persistent Threats (APT). To counter such attacks, attack investigation approaches have been extensively researched for identifying and tracing attack behaviors within information systems, which is an important research field of forensic analysis [1–4]. These methods conduct comprehensive causality analysis of a large volume of audit logs collected from ubiquitous system monitoring to identify attack patterns that imply the tactics and objectives of attackers [5–8]. However, traditional methods rely heavily on feature engineering and require extensive manual work [9–12]. In contrast, deep learning (DL) techniques have the capacity to learn irregular patterns from massive amounts of data that may elude human observation, thereby facilitating the automation of data analysis processes.

Previous researches have introduced DL-based methods to advance attack investigation [5,13,14], yielding remarkable results. ATLAS [5] and AIRTAG [14] are state-of-the-art DL-based attack investigation approaches. However, these efforts still suffer certain limitations. ATLAS is a supervised learning method that requires labeled data for training. Unlike general domain DL tasks with publicly available datasets, the realm of attack investigation lacks well-labeled datasets. The reason is that the audit logs contain detailed confidential information from within enterprises, and precisely labeling extensive audit logs necessitates expertise in both log and network security [15]. In response to these limitations, AIRTAG leverages unlabeled log text data for BERT [16] model pre-training and employs a one-class support vector machine (OC-SVM) as a downstream classifier for unsupervised attack investigation. The essence of this unsupervised downstream task is to discover attack behaviors through similarity. However, the data representations learned by the BERT model are to some extent collapsing [17], meaning that almost all log text data are mapped to a small space

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

and therefore produce high similarity. Furthermore, attackers frequently disguise their activities as normal or share processes with legitimate users, causing certain attack behaviors to closely resemble normal user behaviors. These two issues hinder AIRTAG from effectively identifying some attack behaviors in the downstream attack investigation task.

To address the above-mentioned limitations, this paper employs the contrastive learning (CL) framework and sequence representation techniques for capturing irregular behavior patterns present in audit logs. This model can perform representation learning on a large amount of unlabeled data and capture token-level and sequence-level features based on the training objective tasks. Furthermore, The CL framework encourages two augmented sequences from the same behavior to be closer while keeping sequences from different behaviors far apart [18]. Thus, it can improve the accuracy of unsupervised classifier in identifying disguised attack behavior, and it can realize supervised fine-tuning by using pre-trained models and embedded representations to learn both attack and normal behavior sequences with a small number of labeled samples.

This paper proposes ConLBS, an attack investigation approach by contrastive learning with behavior sequence. Behavior sequences are introduced to describe the behavior patterns of high-level behaviors, which contain contextual information about system events and represents the execution flow of various behaviors at the system level. ConLBS combines the contrastive learning framework with a multi-layer Transformer network to acquire embedded representations of unlabeled behavior sequences, and then it trains a classifier for identifying attack behavior sequences. The overall workflow of ConLBS is depicted in Figure 1. In the initial phase, ConLBS creates platform-independent provenance graphs from audit logs and optimizes these graphs to reduce their complexity before proceeding to construct behavior sequences. Then, ConLBS employs Depth-First-Search (DFS) to gather context information about system events and generate behavior sequences. Additionally, a novel lemmatization strategy is introduced to extract the semantics of behavior sequences. Second, building upon the SimCLR framework [19], ConLBS has devised a contrastive learning model that facilitates the acquisition of embedded representations for unlabeled behavior sequences at both the entity-level and sequence-level. Four sequence augmentation strategies are proposed for contrastive learning. Finally, ConLBS proves versatile in its application, as it can be utilized for both unsupervised single-class task training and fine-tuning for supervised single-sentence classification tasks, depending on the availability of labeled data. The performance of ConLBS in identifying attack events is evaluated with 13 attack scenarios in two public datasets. The results show that ConLBS can effectively identify attack behavior sequences in the cases of unlabeled data or less labeled data to realize attack investigation. And compared with existing methods and models, our method achieves superior results.

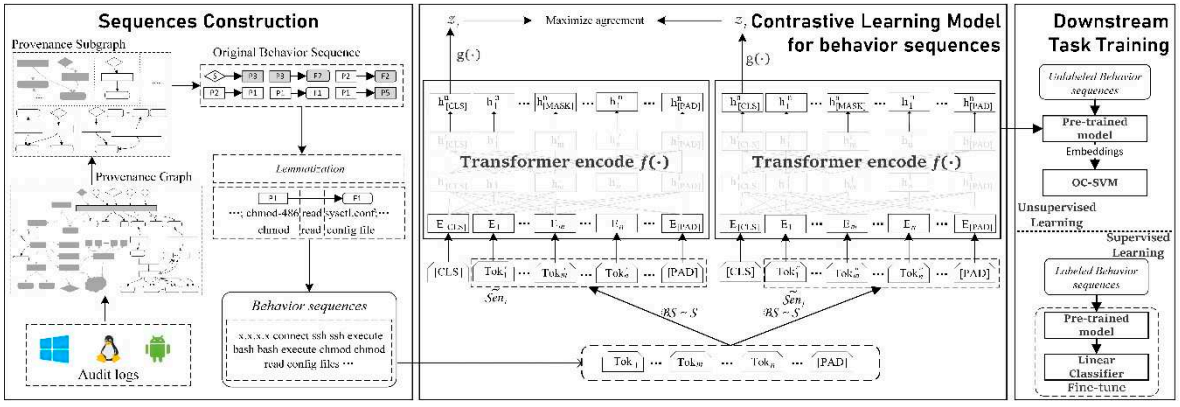

Figure 1. The overall of ConLBS workflow.

 doi:10.20944/preprints202311.0019.v1

2. Related Work

2.1. Attack Investigation

Audit logs are collected by system monitoring tools from different operating systems. An audit log encapsulates a specific system event or system call that includes system entities, relationships, timestamps, and other essential system-related information. The concept of constructing provenance graphs from OS-level audit logs was proposed by King et al. [20]. Some investigations in the area of attack analysis utilize rule-based or Indicator of Compromises (IOCs) matching methods to identify possible threat behaviors. Nevertheless, the precision and comprehensiveness of the rule database and IOCs are crucial factors that impact the effectiveness of these techniques [2,10]. Holmes [2] maps low-level audit logs to tactics, techniques, and procedures (TTPs) and advanced persistent threat (APT) stages through rule-based matching within the knowledge base. Other techniques propose investigation strategies based on statistical analysis, leveraging the comparatively lower frequency of threat events in contrast to normal events to determine the authenticity of the alerts [21]. However, such methods may mistakenly categorize low-frequency normal events as high-threat occurrences. OmegaLog [6] combines application event logs and system logs to create a Universal Provenance Graph (UPG) that portrays multi-layer semantic data. In contrast, WATSON [3] infers log semantics from contextual indications and consolidates event semantics to depict behaviors. This technique greatly decreases the effort required for investigating attacks. However, aforementioned traditional methods rely heavily on feature engineering and require extensive manual work.

Deep learning-based approaches enable the creation of attack investigation models by identifying unique features of normal or malicious behaviors [5,13,14]. ATLAS [5] applies Long Short-Term Memory (LSTM) networks for supervised sequence learning. AIRTAG [14] parses log files, utilizing BERT to train a pre-trained model, and subsequently train a downstream classifier. However, these methods are constrained by the availability of high-quality labeled data and model performance, making them less effective in addressing certain specific scenarios in real-world environments. These scenarios may include situations where the number of attack behaviors is significantly lower than that of normal behaviors, leading to sample imbalance, or cases in which attackers' disguises result in high similarity between attack sequences and normal sequences.

2.2. Contrastive Learning Framework

Recently, the use of contrastive learning in natural language processing (NLP) tasks has increased significantly [22–25]. Some of these methods draw inspiration from the SimCLR architecture, such as DeCLUTR [25] and CLEAR [18]. DeCLUTR takes a holistic training approach by amalgamating both contrastive and masked language model objectives. However, their primary focus lies in utilizing spans for contrastive learning, which may potentially result in fragmented semantic comprehension. CLEAR closely aligns with DeCLUTR in terms of architecture and objectives. Both approaches place a central emphasis on pre-training language models, albeit requiring substantial corpora and resource investments.

The SimCLR architecture consists of four components: (1) The data augmentation strategies ($t \sim T$) is used to independently generate different input samples; (2) A base encoder network $f(\cdot)$; (3) A projection head $g(\cdot)$; (4) A contrastive loss function that maximizes the agreement. Figure 2. illustrates the fundamental framework for contrastive learning of data representations. Depending on the data characteristics, data augmentation strategies can be explored to enhance downstream tasks. An appropriate encoding network can be chosen for $f(\cdot)$, such as GNN or BERT, based on the specific task requirements.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

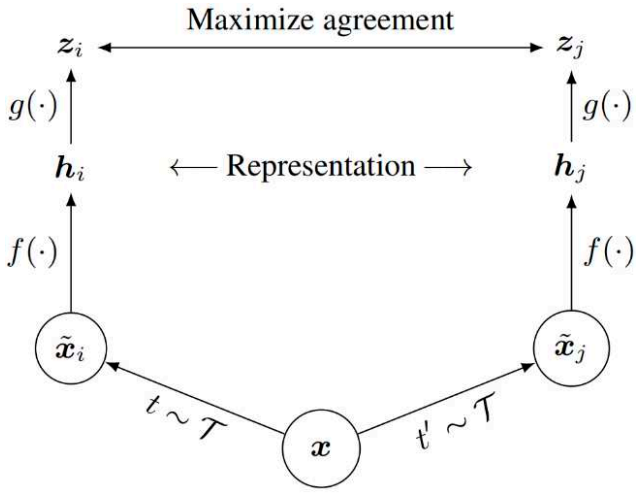

Figure 2. The SimCLR architecture [19].

3. Methodology

3.1. Provenance graphs construction and optimization

Provenance graphs construction. ConLBS extracts the system event as a quadruple $event =< sub, oper, obj, Time >$, where $oper$ denotes the operation action from a subject sub to an object obj, and $Time$ represents the timestamp. For example, a log recording the reading of a code file could be represented as $< code.exe\_43200, read, \%Path\% \backslash main.py, 2023/7/22\ 9{:}31{:}32 >$. Then, ConLBS performs causal correlation on the extracted system events to construct platform-independent provenance graphs. These graphs signify the behavior processes and information flows in the OS-level. The nodes stand for subjects and objects, while the directed edges signify subject operations on objects. ConLBS can gather comprehensive contextual information of system events from the provenance graphs, resulting in a more accurate portrayal of behavioral patterns. As shown in Figure 3., step A demonstrates the process of constructing provenance graphs from audit logs.

Provenance graphs optimization. Audit logs record coarse-grained system operations and a lot of redundant information, leading to large and complex provenance graphs. ConLBS eliminates erroneous dependencies and decreases graph complexity while retaining crucial behavioral data for attack investigation.

First, ConLBS splits provenance graphs into subgraphs that describe different high-level behaviors. An intuition is that system events belonging to the same behavior occur at shorter intervals and have the similar patten. The formula is designed to model this intuition:

$$SIM(event_i, event_j) = \theta * (1 - \frac{T_j - T_i}{T_{end} - T_{start}}) + \mu * \frac{sim\_tok(e_i, e_j)}{max\_len} \quad (1)$$

θ and μ are the weight coefficient. In this formula, $sim\_tok(e_i, e_j)$ represents the similarity between entities e_i and e_j in two events. The formula is as fellow:

$$\frac{sim\_tok(e_i,e_j)}{max\_len} = \begin{cases} 0 & if \quad e_{i.type} \neq e_{j.type} \\ same\_name(e_i, e_j) & else\ if \quad e_{type} = process \\ \frac{same\_bit + same\_port}{33} & else\ if \quad e_{type} = IP \\ \frac{same\_tok(e_i,e_j)}{max(len(e_i),len(e_j))} & else\ if \quad e_{type} = file\ or\ url \end{cases} \quad (2)$$

where type denotes the types of the entities in system events. $same\_name(e_i, e_j)$ is set 1 if the process name and PID both are same, otherwise the value is 0. $same\_bit$ counts the number of the same initial bits of IP address. Each directory name of file or url is treated as a token. $same\_tok(e_i, e_j)$ represents the number of the same tokens. We group system events based on whether the $SIM(event_i, event_j)$ exceeds specified threshold, which is set to 0.7. According to the above formula, the entity is divided into several partitions.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

Second, the redundant and behavior-unrelated system events are identified and removed. Among the audit logs, only one or a few logs are directly related to the behavior, while other logs record the system calls triggered by the behaviors. These behavior-unrelated system events appear repeatedly in different behaviors, and even if removed do not affect the flow of information and evidence related to the attack. Therefore, the above clustered system events are merged and renamed with semantic descriptions.

Third, ConLBS merges multiple directed edges with the same operation between a subject and an object. The timestamps are modified to a time range for determining the sequence of system events.

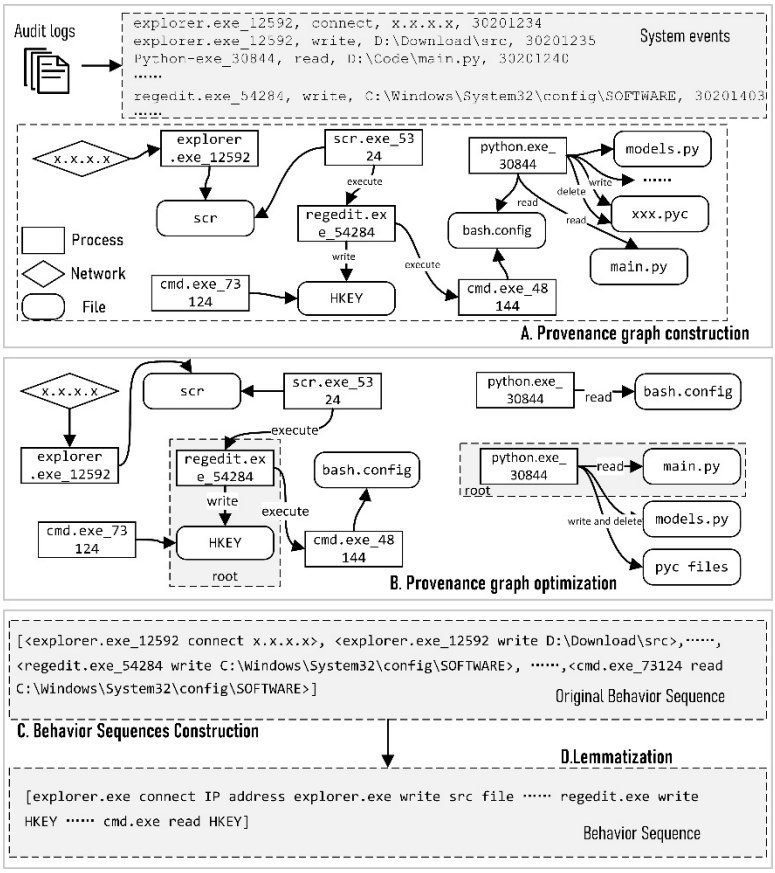

Figure 3. The process of constructing behavior sequences from audit logs.

3.2. Behavior Sequences

ConLBS extracts behavior sequences from the optimized graphs (step C in Figure 3), can describe the behavior patterns of high-level behaviors at the system level. Subsequently, the original semantic of behavior sequences is extracted by using lemmatization (step D in Figure 3). Compared with ATLAS, ConLBS does not rely on labeled attack entities in the process of constructing sequences, and the lemmatization strategy proposed by ConLBS is more suitable for describing behavior semantics.

Behavior sequence construction. The system events are taken as the root, such as < regedit. exe_54284 write C:\Windows\System32\config\SOFTWARE (HKEY) > , and DFS with specific termination conditions are used to traverse forward and backward to obtain the context information. Specifically, during backward traversal of the graph, the constraint is enforced to ensure that the timestamp of each subsequent edge must be monotonically increasing compared to all preceding edges. In contrast, during the forward traversal of the graph, another constraint is enforced, requiring that the timestamp of each preceding edge must maintain a monotonically decreasing order in relation to all other edges.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

Lemmatization. ConLBS employs lemmatization to eliminate noise, such as hostnames in file paths, and to extract the original semantics of the entities within the sequences. Previous efforts have also considered noise removal and semantic extraction, many of them have resulted in the loss of some semantics [2]. This study utilizes dedicated mapping rules tailored to various types of nodes to ensure a more comprehensive semantic representation. Table 1 illustrates a partial representation of the semantic mapping rules for the three types of nodes. For process entities, semantic description is derived from the process name, which serves as the primary source of semantic information for these nodes. For network entities, IP addresses are categorized as either 'internal' or 'external'. Additionally, websites are referred to as 'URLs'. For file entities, specific rules are applied based on the file types. Firstly, the content of the file description is used to extract semantic information. For example, a picture file (Desktop\moon.jpg) is mapped to 'picture file'. Secondly, semantic information can be extracted from the file path. For example, files located at C:\Windows\system32 are represented as 'system file'. Thirdly, for files that do not meet the aforementioned mapping rules, the file type or suffix is utilized to convey semantics.

Table 1. The partial rules of the semantic extraction.

Type	Node Name	Semantic
process	name_PID	name
network	IP-Port, website	IP address, url
file	.jpg, .png, .py, .java	picture file, code file
	\system32\, \Program files\	system file, app file
	*.html, *.lst	html file, lst file

3.3. Behavior sequence augmentation

In this paper four different augmentation strategies are explored based on common situations in attack investigation to enhance the differentiation between attack and normal behaviors, shown in Figure 4.

(a) *Sequence truncation* randomly remove events from the head and tail of the behavior sequences and preserve the continuous sequence in the middle. The maximum length of the removed event is set to $max\_len = 0.2 * k$, where k is the total length of the sequence. The truncation enables the model to learn the intermediate process of the behaviors.

(b) *Event deletion* randomly selects events in the behavior sequence and replaces them with a special token [DEL]. Percentage of events deleted was 20%. This strategy simulates scenarios where some system events were not recorded by the monitor tools or were lost.

(c) *Noise addition* inserts random events into the behavior sequences. The inserted position is random. The addition of noise simulates scenarios in which the behavior sequence may include system events that do not belong to that particular behavior. Events of 5% length are randomly added at four selected locations, ensuring a total length of around 20%.

(d) *Substitution* is a strategy used to enhance the robustness of the model. It involves randomly selecting certain events and replacing them with other events that share same entity. The number of replaced events does not exceed 20%.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

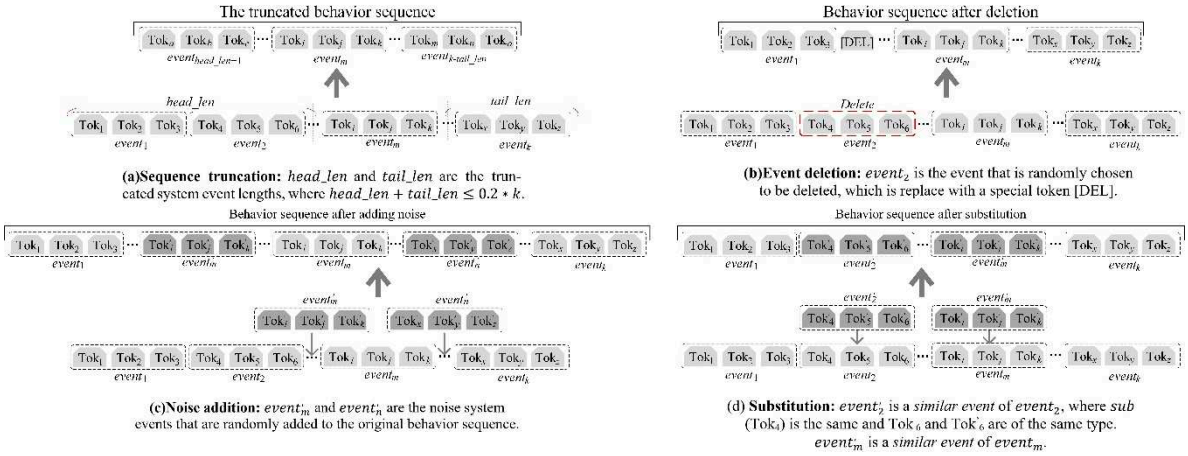

Figure 4. Four different basic behavior sequence augmentation strategies. The system events are smallest unit of action.

3.4. Behavior sequence representation

The four main components of our CL framework are shown in the Figure 1. Data augmentation strategies (BS ~ S) are used to generate two related augmented behavior sequences $\mathcal{S}\tilde{e}n_i$ and $\mathcal{S}\tilde{e}n_j$ from the initial behavior sequence.

Multilayer Transformer encoder. We utilize the multilayer Transformer to learn the representation of the input behavior sequences $\mathcal{S}\tilde{e}n_i$ and $\mathcal{S}\tilde{e}n_j$. The pre-training task is the same as BERT MLM, we randomly mask 15% tokens of the input behaviors, and among the selected tokens, 80% probability is replaced by $[MASK]$, 10% probability is randomly replaced by other tokens, and 10% probability is left unchanged. The loss function for the masked tokens is defined as:

$$\mathcal{L}_{MLM} = -\sum_{i=1}^{M} log\left(p\left(\widetilde{tok}_i = tok_i|\theta,\theta_1\right)\right), tok_i \in V] \quad (3)$$

where M is the number of masked entities, θ is the parameters of the Transformer Encoder, θ_1 is the parameter of the output layer connected to the Encoder in the Masked Entity task. Probability function p depends on the parameters θ and θ_1, $\widetilde{tok}_i$ represents a token masked at the $i-th$ position in the tokenized behavior sequence.

Projection head. A small neural network projection head $g(\cdot)$ that maps representations to the space where contrastive loss is applied. A MLP is used with one hidden layer to obtain $z_i = g(h_i^n) = \mathrm{W}^{(2)}\sigma\mathrm{W}^{(1)}h_i^n$, where σ is a ReLU non-linearity. Previous work has proved it beneficial to define the contrastive loss on z_i rather than h_i^n.

The Loss for Training. The contrastive learning loss has been tremendously used in previous work [17,19]. Following these works, we use the contrastive learning loss function for a contrastive prediction task, that is, trying to predict positive augmentation pair $\mathcal{S}\tilde{e}n_i$ and $\mathcal{S}\tilde{e}n_j$ in the augmented set $\{\mathcal{S}\tilde{e}n\}$ (the sample size is 2*N*). The two variants from the same behavior sequence form the positive pair, while the other 2(*N*−1) augmented samples in the set are treated as negative examples. The loss function for a positive pair is defined as:

$$l(i,j) = -log\frac{exp(sim(z_i,z_j)/\mathcal{T})}{\sum_{k=1}^{2N}\mathbb{I}_{[k\neq i]}exp(sim(z_i,z_j)/\mathcal{T})} \quad (4)$$

Where $\mathcal{T}$ is a temperature parameter, $sim(z_i,z_j)$ denotes the cosine similarity of two vector z_i and z_j, and $\mathbb{I}_{[k\neq i]}$ is an indicator function to judge whether $k \neq i$. Finally, we average all 2N in-batch classification losses to obtain the final contrastive loss:

$$\mathcal{L}_{ConL} = \frac{1}{2N}\sum_{i=1}^{2N}\sum_{j=1}^{2N} b(i,j)\ l(i,j) \quad (5)$$

When i and j is a positive pair, $b(i,j)$ returns 1, otherwise 0.

The overall loss function is obtained by combining the loss function of multilayer Transformer encoder (token-level) and the loss function of contrastive learning (sequence-level):

$$\mathcal{L}_{total} = \mathcal{L}_{MLM} + \mathcal{L}_{ConL} \quad (6)$$

 doi:10.20944/preprints202311.0019.v1

3.5. Sequence classification training

Supervised learning. In real enterprise environments, Intrusion Detection Systems (IDS) and security analysts label logs related to discovered attacks. We can utilize these labeled data to fine-tune the model to learn both attack and normal behavior patterns. Since the behavior sequence representation phase has already enabled the model to learn features of behavior sequences, only a small amount of data is needed for fine-tuning. This paper abstracts behavior sequence classification as a single-sentence binary classification task and employs linear classifier MLP for downstream task training. The experiments demonstrate that using 500 labeled samples can achieve comparable results with ATLAS training on the entire dataset.

Unsupervised learning. Unsupervised methods can effectively address challenges arising from data imbalances during training for downstream tasks. This paper uses OC-SVM for training the downstream task, which has been proven effective in previous work [14]. Unlabeled datasets that do not contain attacks are employed for training to learn normal behavior patterns. During testing, attack behavior sequences are identified by detecting outliers, which are sequences positioned outside the classifier's boundary.

4. Experiment

4.1. Datasets and setups

Datasets. The performance of ConLBS is evaluated using two publicly available datasets, including ATLAS dataset [5] and DAPRA CADETS dataset [26]. Both datasets contain multiple simulated attack scenarios. Throughout the attack behaviors, normal behaviors such as SSH login may also occur on the hosts. The size of these two datasets is comparable to real-world data.

Setups. For the Model configuration, like the previous method [16], our Transformer is set to 12 layers, 12 heads, and 768 hidden layers. The minibatches contains 256 behavior sequences of maximum length 512 tokens. We adopt Adam optimizer and set the learning rate to 5e-7 and we use 0.1 for dropout on all layers and in attention. The temperature $\mathcal{T}$ of the loss is set to 0.1. A MLP with one hidden layer are used to obtain $z_i = g(h_i^n)$. After train ing is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation h_i^n for categorizing behavioral sequences.

4.2. Attack Investigation Results

When evaluating the performance of ConLBS, we employ labeled data from the datasets for fine-tuning, simulating the scenario in which logs are labeled by security analysts in real enterprise environments. Table 2 reports the results of ConLBS at predicting attack events in each attack scenario. As seen, ConLBS correctly predicts both attack and normal events with an average F1-score of 99.786% and 99.823% across both datasets. It can be seen from the results that the quantity of FP and FN is very small compared with that of TP and TN, so we can obtain high Precision and Recall values. By comparing FP and FN, our method incorrectly predicts normal events as attacks more frequently. This outcome is acceptable in real attack investigation, because the risk of underreporting attacks outweighs that of falsely reporting them.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

Table 2. Attack investigation results on two datasets.

Attack Scenarios	Attack Investigation Results						
	TP	TN	FP	FN	Precision	Recall	F1-score
ATLAS. S-1	4,536	78,856	28	13	99.387%	99.714%	99.550%
ATLAS. S-2	13,584	331,051	47	10	99.655%	99.926%	99.791%
ATLAS. S-3	4,975	109,285	22	23	99.560%	99.540%	99.550%
ATLAS. S-4	13,199	88,576	21	4	99.841%	99.970%	99.905%
ATLAS. M-1	6,331	171,131	13	9	99.795%	99.858%	99.827%
ATLAS. M-2	28,914	180,326	51	17	99.824%	99.941%	99.883%
ATLAS. M-3	24,728	140,347	94	7	99.621%	99.972%	99.796%
ATLAS. M-4	5,945	137,167	24	22	99.598%	99.631%	99.615%
ATLAS. M-5	23,526	452,354	86	37	99.636%	99.843%	99.739%
ATLAS. M-6	6,372	201,569	17	22	99.734%	99.656%	99.695%
ATLAS. Avg.	13,211	189,066	40	16	99.696%	99.876%	99.786%
CADETS. case-1	87,658	436,957	218	76	99.752%	99.913%	99.833%
CADETS. case-2	53,631	472,913	175	49	99.675%	99.909%	99.792%
CADETS. case-3	34,097	209,681	58	47	99.830%	99.862%	99.846%
CADETS. Avg.	58,462	373,184	150	57	99.744%	99.902%	99.823%

Figure 5 show the ROC curve of ConLBS on two datasets. The ROC curve demonstrates that our classification model achieves excellent results in both datasets, which shows that ConLBS can effectively identify attack events and realize attack investigation. In fact, attack investigation results shows that there is a large difference between the attack behavior sequences and the normal behavior sequences. Attack behaviors typically involve intricate steps and numerous operations, often leading to longer behavior sequences that encompass more entities. In contrast, normal user behavior mostly performs simple and repetitive actions, which results in a large number of shorter, similar sequences.

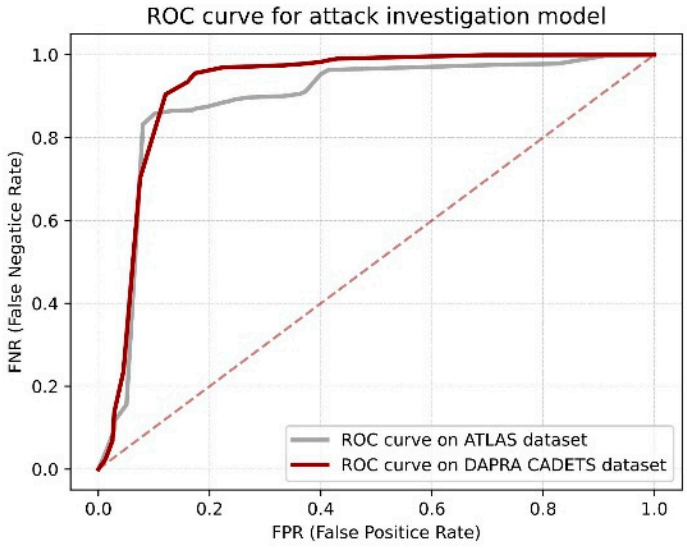

Figure 5. ROC curve of ConLBS on two datasets.

The results in Table 3 illustrate the effect of different lemmatization strategies and sequence representation models on the classification results. The model's performance is weak when using raw unprocessed semantics. And the results reveal that ConLBS's lemmatization strategy outperforms ATLAS's lemmatization strategy. The experimental results show that appropriate semantic information can improve the classification effect of the model. Using $\text{BERT}_{\text{Re-train}}$, a pre-trained sequence representation model obtained by using behavior sequences in our contrastive

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

learning model, achieves better results (F1-score +0.606%) compared to directly using the public $BERT_{Base}$ model. This is because the generic model lacks a significant number of unknown words in the behavioral sequences.

Table 3. Performance comparison of ConLBS using different semantic granularity and pre-trained models.***RS*** indicates the use of raw unprocessed semantics, and ***Lem*** indicate the use of semantics obtained using lemmatization techniques.

Method	Precision	Recall	F1-score
RS **+BERT**$_{Base}$	87.782%	84.333%	86.023%
Lem $_{ATLAS}$**+BERT**$_{Base}$	97.102%	92.184%	94.579%
Lem $_{ConLBS}$**+BERT**$_{Base}$	99.532%	98.831%	99.180%
RS **+BERT**$_{Re\text{-}train}$	93.850%	89.700%	91.728%
Lem $_{ATLAS}$**+BERT**$_{Re\text{-}train}$	99.132%	99.365%	99.248%
Lem $_{ConLBS}$**+BERT**$_{Re\text{-}train}$	**99.696%**	**99.876%**	**99.786%**

4.3. Comparison Analysis

This paper compares ConLBS with to state-of-the-art supervised and unsupervised attack investigation methods. Figure 6. illustrates the number of FN and FP for ConLBS and AIRTAG in various attack scenarios. ConLBS exhibits a lower average number of FN compared to AIRTAG, while its average number of FP is slightly higher than that of AIRTAG. These results indicate that CL model of ConLBS effectively increases the separation between attack and normal sequences. Figure 7. shows the performance of ATLAS and ConLBS (Fine-tune) trained with different numbers of labeled samples. When using 500 labeled samples, ConLBS achieves comparable results with ATLAS and ConLBS trained with full (30721) labeled samples. This result signifies that ConLBS can efficiently conduct attack investigations even when there is a scarcity of attack samples.

Figure 6. The number of False Negatives (FN) and False Positives (FP) of the AIRTAG and ConLBS.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 1 November 2023 doi:10.20944/preprints202311.0019.v1

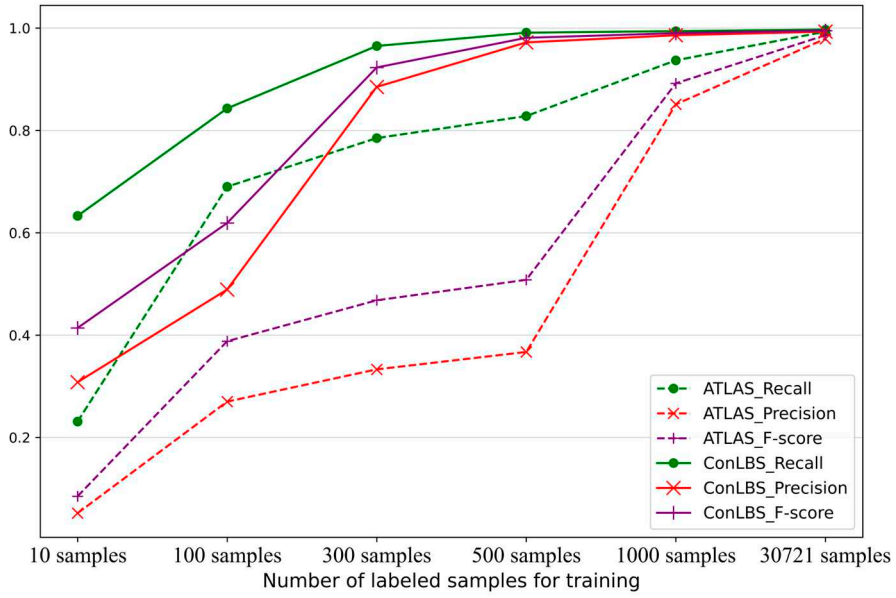

Figure 7. Performance of ATLAS and ConLBS (Fine-tune) trained with different numbers of labeled samples.

This paper also compares ConLBS with several typical deep learning models, as presented in Table 4. In comparison to CNN [28] and LSTM [29], the behavior sequences are sampled to achieve a balance between positive and negative samples. Word2vec [30] is employed to sequences and convert them into fixed-dimensional feature vectors. The results show that the performance of CNN is much lower than other methods, because the convolution kernel and window size limit the effective learning of long sequences. LSTM solves this problem, but is limited by word2vec embeddings. BERT [16] and RoBERTa [31] have demonstrated good results, but encountering attacks that masquerade as normal behavior is challenging. Certain segments of these attack behavior sequences are similar to normal behaviors.

Table 4. Comparison of ConLBS with deep learning models.

Base models/method	Recall	Precision	F1-score
Word2vec+CNN [27]	87.425%	89.379%	88.391%
Word2vec+LSTM[28]	95.854%	96.412%	96.132%
BERT [16]	98.460%	98.891%	98.675%
RoBERTa [29]	99.601%	99.829%	99.715%
ConLBS	**99.902%**	99.744%	99.823%

4.4. Runtime Performance of ConLBS

The time consumption of ConLBS is measured on two publicly available datasets. The size of these two datasets is comparable to real-world data. Table 5 reports the runtime performance of attack investigation methods. During the data preprocessing phase, the average processing speed of constructing dependency graphs from the datasets is 358MB/min. The total time cost of reading log data, constructing graphs, and extracting behavior sequences by ConLBS is 23 minutes and 48 seconds. The training process is offline, and once completed, the model does not need to re-learn previously learned behavior sequences. The training time consumption of ConLBS exceeds that of ATLAS due to ConLBS having a larger number of learned samples. Ultimately, the model's average time to identify a sequence as an attack is 2.53 s.

 doi:10.20944/preprints202311.0019.v1

Table 5. Runtime performance of attack investigation approaches.

Method	Logs size（/min）	Graph/Sequence construction	Train Time	Investigation Time (Avg.)
POIROT[1]	114.5MB	1:54:35	--	7.72s
ATLAS	169MB	0:30:23	0:28:26	5.0s
ConLBS	**358MB**	**0:23:48**	0:36:35	**2.53s**

5. Conclusion

Existing supervised attack investigation approaches require labeled and balanced data for training. While unsupervised methods can mitigate the issues mentioned above, the high degree of similarity between certain real-world attack behaviors and normal behaviors in the sequences makes it challenging for current unsupervised methods based on BERT to accurately identify disguised attacks. Thus, this paper introduces ConLBS, which does not rely on labeled data to learn embedded representation of behavior sequences, and can be trained either supervised or unsupervised depending on the availability of labeled data. This paper introduces behavior sequences to describe high-level behavior patterns and explores several sequence augmentation strategies for enhancing contrastive learning. The results show that ConLBS can effectively identify attack behavior sequences in the cases of unlabeled data or less labeled data to realize attack investigation.

In future work, we plan to explore new representations of behavior patterns, such as using a topological approach to represent the execution flow of behavior at the system level. In addition to this, exploring data-enhancement strategies that can facilitate downstream tasks and improving contrastive learning models will also be part of the future work.

Acknowledgments: This work is supported in part by the National Natural Science Foundation of China under Grant U21B2020 and Grant U1936216.

References

1. Milajerdi S M, Eshete B, Gjomemo R, et al. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunt-ing[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. London, UK, 2019: 1795-1812.
2. Milajerdi S M, Gjomemo R, Eshete B, et al. Holmes: real-time apt detection through correlation of suspicious information flows[C]//Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP). SAN FRANCISCO, USA,2019: 1137-1152.
3. Zeng J, Chua Z L, Chen Y, et al. Watson: Abstracting behaviors from audit logs via aggregation of contextual seman-tics[C]//Proceedings of the 28th Annual Network and Distributed System Security Symposium, NDSS. 2021.
4. Gao P, Shao F, Liu X, et al. Enabling efficient cyber threat hunting with cyber threat intelligence[C]//2021 IEEE 37th International Conference on Data Engineering (ICDE). IEEE, 2021: 193-204.
5. Alsaheel et al., “ATLAS: A Sequence-based Learning Approach for Attack Investigation”, in Proc. of 30th USENIX Security Sym-posium, [Online], 2021, pp. 3005-3022.
6. W. U. Hassan, M. A. Noureddine, P. Datta, A. Bates, “OmegaLog: High-Fidelity Attack Investigation via Transparent Multi-layer Log Analysis”, in Proceedings of Network and Distributed System Se-curity Symposium 2020, [Online], 2020.
7. P. Gao, X. Xiao, Z. Li, F. Xu, S. R. Kulkarni and P. Mittal, “AIQL: Enabling Efficient Attack Investigation from System Monitoring Data,” in Proceedings of 2018 USENIX Annual Technical Confer-ence (USENIX ATC 18), Boston, USA, 2018, pp. 113-126.
8. K. Yonghwi, et al. "MCI: Modeling-based Causality Inference in Audit Logging for Attack Investigation," in Proceedings of Net-work and Distributed System Security Symposium, vol. 2, pp. 4, 2018.
9. Zhao J, Yan Q, Liu X, et al. Cyber Threat Intelligence Modeling Based on Heterogeneous Graph Convolutional Network[C]//23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020). 2020: 241-256.
10. M. N. Hossain, S. Sheikhi, and R. Sekar. "Combating dependence explosion in forensic analysis using alternative tag propagation semantics," in 2020 IEEE Symposium on Security and Privacy (SP), pp. 1139-1155, 2020.

11. Zhu T, Wang J, Ruan L, et al. General, Efficient, and Real-time Data Compaction Strategy for APT Forensic Analysis[J]. IEEE Transactions on Information Forensics and Security, 2021.
12. R. Yang et al. RATScope: Recording and Reconstructing Missing RAT Semantic Behaviors for Forensic Analysis on Windows[J]. IEEE Trans. Dependable and Secure Computer. 2020: 1–1.
13. Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In ACM SIGSAC Conference on Computer and Commu-nications Security, 2017.
14. Ding H, Zhai J, Nan Y, et al. {AIRTAG}: Towards Automated Attack Investigation by Unsupervised Learning with Log Texts[C]//32nd USENIX Security Symposium (USENIX Security 23). 2023: 373-390.
15. Liu, Fucheng, et al. "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise." Pro-ceedings of the 2019 ACM SIGSAC conference on computer and communications security. 2019.
16. J. Devlin, M. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of NAACL-HLT, 2019, pp.4171-4186.
17. Yan Y, Li R, Wang S, et al. Consert: A contrastive framework for self-supervised sentence representation transfer[J]. arXiv preprint arXiv:2105.11741, 2021.
18. Wu Z, Wang S, Gu J, et al. Clear: Contrastive learning for sentence representation[J]. arXiv preprint arXiv:2012.15466, 2020.
19. Chen T, Kornblith S, Norouzi M, et al. A simple framework for contrastive learning of visual representations[C]//International con-ference on machine learning. PMLR, 2020: 1597-1607.
20. King, Samuel T., and Peter M. Chen. "Backtracking intrusions." in Proc. ACM Symp. Oper. Syst. Princ, pp. 223-236. 2003.
21. W. U. Hassan, et al., "Nodoze: Combatting threat alert fatigue with automated provenance triage." in Proceedings of Network and Dis-tributed System Security Symposium 2019, USA, 2019.
22. Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method by mu-tual information maximization. In Proceedings of the 2020 Confer-ence on Empirical Methods in Natural Language Processing (EMNLP), pages 1601–1610.
23. Hongchao Fang and Pengtao Xie. 2020. Cert: Contrastive self-supervised learning for language understanding. arXiv preprint arXiv:2005.12766.
24. Fredrik Carlsson, Magnus Sahlgren, Evangelia Gogoulou, Amaru Cuba Gyllensten, and Erik Ylipa¨a¨ Hellqvist. 2021. Semantic re-tuning with contrastive tension. In International Conference on Learning Representations.
25. John M Giorgi, Osvald Nitski, Gary D Bader, and Bo Wang. 2020. Declutr: Deep contrastive learning for unsupervised textual repre-sentations. arXiv preprint arXiv:2006.03659.
26. J. Torrey, "Transparent Computing Engagement 3 Data Release," 2020, [Online]. Available: https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md.
27. Wang Q, Hassan W U, Li D, et al. You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis[C]//NDSS. 2020.
28. Y. Zhang, and B. C. Wallace. "A Sensitivity Analysis of (and Prac-titioners' Guide to) Convolutional Neural Networks for Sentence Classification," in Proc. Int. Jt. Conf. Nat. Lang. Process., vol. 1, pp. 253-263, 2017.
29. S. Hochreiter, and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
30. M. Tomas, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Dis-tributed representations of words and phrases and their composi-tionality." in Adv. Neural inf. Process. Syst., -vol. 2, pp. 3111-3119, 2013.
31. Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," in Proc. Int. Conf. Learn. Represent., 2020.