T-Trace: Constructing the APTs Provenance Graphs Through Multiple Syslogs Correlation

Teng Li, *Member, IEEE*, Ximeng Liu, *Senior Member, IEEE*, Wei Qiao, Xiongjie Zhu, Yulong Shen, *Member, IEEE*, and Jianfeng Ma, *Member, IEEE*

***Abstract*—Advanced Persistent Threats (APTs) employ sophisticated and covert tactics to infiltrate target systems, leading to increased vulnerability and an elevated risk of exposure. Consequently, it is essential for us to proactively create an extensive and clearly outlined attack chain for APTs in order to effectively combat these threats. Unlike traditional malware or application threats, APTs can sidestep cyber security efforts and cause severe damage to organizations or even state security. Nonetheless, earlier methods struggle to accurately track APTs and may face a dependency explosion issue, as identifying the intricate and complex unknown malicious activities within APTs proves to be challenging. In this paper, we propose and build an approach, T-trace, which constructs the events provenance graphs by analyzing the correlations among logs. The approach precisely finds the log communities with tensor decomposition and calculates significance scores to extract the events. The APTs can be inferred by discovering the event communities and constructing the provenance graph with log correlation. In the experiment, we used DARPA data sets and launched four current practical APTs. Compared with current approaches, the results show that T-trace can efficiently reduce time cost by 90% and achieve a 92% accuracy rate in constructing the provenance graph, which can be practically applied in APTs provenance.**

***Index Terms*—APTs, forensic system, log analysis, provenance, tensor decomposition.**

I. Introduction

As THE name *advanced* suggests, Advanced Persistent Threats (APTs) apply continuous, clandestine, and sophisticated hacking techniques to gain access to a system and remain inside for a prolonged time, with potentially destructive consequences. Therefore, tracing the attack roadmap and constructing a provenance graph are critical to apprehend the Tactics, Techniques, and Procedures (TTPs) of APTs, which can further proactively prevent the threats. However, unlike traditional malware or application threats, APTs can sidestep cyber security efforts and cause severe damage to organizations or even state security, in that: first, they use more sophisticated and complex methods to penetrate the networks. In layman's terms, they are not hit-and-run attacks and attempt to remain in the system for a more extended period. Through the execution launched by hackers, APTs not only aim to attack the specific part of a network but the entire network instead. These characteristics bring new challenges while constructing the attack provenance graph.

Apart from the above challenges, the common Intrusion Detection Systems (IDS) [1] fail to detect such attacks because they can evade detection technologies [2], [3]. To address the issues, the existing research works rely on log files and the previous execution steps to retrieve the entire attack operations [4]. Although some works such as [5], [6], [7], [8] have illustrated great potentials of log provenance in forensic analysis, several limitations are still suffered:

1) Insufficient Pattern Learning: Current approaches [9], [10], [11], [12] take a set of representative training data as input and apply a sequence process technique (e.g., recurrent neural network) to predict the upcoming event based on the past events. Deeplog [10] and Tiresias [11] model normal user behaviors and flag deviations as anomalies. Automated APT classifier [12] studies the existing behaviors of the real APTs. Since these approaches identify malware and APT techniques based on the existing patterns learned from the training data, they fail to identify 41% malware that is not listed in the training data. Apparently, an unknown APT technique cannot be detected if it is zero-day [13]. Problems with training data can lead to false positives, false negatives, or both [14].

2) Lack of Semantic Correlations: Existing approaches [10] [4] only focus on analyzing log entries without considering the relationships among events. Hence, the activities cannot be obtained. NoDoze [15] and PrioTracker [16] trace the anomalous logs and information flows to pinpoint the APTs. However, these methods only consider the causal correlations among multiple logs without considering the time-lined event chain or the temporal correlations of a single log without considering inner relationships among multiple logs. Such a scheme reduces the forensic performance as the existing causal temporal correlations that are quite complex. Regarding CamQuery [7], it provides inline, realtime provenance analysis, and focuses more on the analysis

Manuscript received 2 December 2021; revised 22 April 2023; accepted 26 April 2023. Date of publication 8 May 2023; date of current version 16 May 2024. This work was supported in part by the National Natural Science Foundation of China under Grants 62272370 and U21A20464, in part by Young Elite Scientists Sponsorship Program by CAST under Grant 2022QNRC001, in part by Shaanxi University of Science and Technology Association Youth Talent Promotion Project under Grant 20210120, in part by the China 111 Project under Grant B16037, in part by the Fundamental Research Funds for the Central Universities under Grant QTZX23071. *(Corresponding author: Ximeng Liu.)*

Teng Li, Xiongjie Zhu, and Jianfeng Ma are with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: litengxidian@gmail.com; aqaqzxsw123@gmail.com; jfma@mail.xidian.edu.cn).

Ximeng Liu is with the College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China (e-mail: snbnix@gmail.com).

Wei Qiao is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China (e-mail: dicer.10@outlook.com).

Yulong Shen is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China (e-mail: ylshen@mail.xidian.edu.cn).

Digital Object Identifier 10.1109/TDSC.2023.3273918

of quick response to timely systems. It focuses on the interaction log between user-level applications and kernel objects captured by the OS reference monitor. CamQuery does not pay attention to lightweight application logs, so it is not detailed enough in the construction of the attack chain. Catching Falling Dominoes focuses [17] more on cloud security analysis, which explains the root cause of security incidents in terms of management operations in clouds. It may result in incomplete attack chain extraction during the pruning process. WATSON [18] only focuses on audit logs in the attack detection process, and focuses more on clustering through semantically similar behaviors, thereby reducing the workload. T-trace is based on multi-source logs, and its focus is to build a complete attack chain, so as to fully reproduce the attack process.

3) Dependency Explosion: Some researches construct provenance graphs to identify APTs [19], [20], [21]. However, their schemes might be inaccurate because each event is dependent on its preceding events, which can cause the dependency explosion problem. Although the approaches SLEUTH [22] and MORSE [14] can be applied to solve such an issue by leveraging tags derived from audit logs to encode the assessment trustworthiness and data sensitivity. Unfortunately, extraction tags from the raw audit logs need the expert knowledge because the derivation is based on the human policies and rules. Although Unicorn [23], Provenance Baseline [24], and Rule-based Detection [25], do not need expert knowledge but our focus is different. UNICORN presents a graph sketching algorithm to summarize long-running system executions. Essentially, it makes use of statistical properties of graphs to represent semantic information. ProvenanceBaseline focus on categorical data. Rule-based Detection detects anomalous entities using an association rule-based scoring and ranking, and dedicated techniques visualizing potential anomalies within the rankings. These schemes also give efficient results, but T-trace is a provenance scheme based on multi-source logs through feature association, and pays more attention to the comprehensiveness and integrity of the provenance attack graph. Apart from the tag-based solution, OmegaLog [4] relies on a universal provenance graph with application logs and UISCOPE [26] attributes the events to high-level UI elements. Nonetheless, these methods do not apply to general systems containing only system logs, program logs, and network logs without other special treatment.

To address the proposed issues, we propose T-trace, a novel concise and training-free forensic system to construct the provenance graphs for the APTs investigation. Basically, T-trace tracks the ”criminal clues” through event logs. T-trace identifies the past activities performed by the attacker and traces the relationships among events to construct provenance graphs. First, T-trace leverages tensor decomposition to mine the implicit relationship among a large number of original logs and obtains log communities from the corresponding log templates. Then, T-trace compares logs in pairwise and calculates significance scores to gather the relevant logs together to identify the events. Next, the APTs provenance graph can be constructed by discovering the event communities with log correlation. Finally, T-trace prunes the irrelevant relationships in the graph to solve the dependency explosion problem and finds out the attack chains of APTs. The data we choose are network flow records, firewall logs and system audit logs, which can be easily accessed on the user systems.

We evaluate T-trace with four launched practical APT attacks, and open-source data sets are also applied to examine the efficiency and accuracy of our approach. The results show that the accuracy of event identification can be 38% higher than prior work, and T-trace efficiently reduce time cost by 90% and achieve a 92% accuracy rate in constructing the provenance graph. To our best knowledge, T-trace is the first to construct APTs provenance graphs from raw logs with causal and temporal correlations and solve the dependency explosion without knowledge-based learning method.

In summary, this paper presents the following contributions:

1) We first propose the saliency scoring algorithm with tensor decomposition to identify the events from logs, which mitigates the semantic gap between low-level logs and understanding high-level events. We only use a small amount of expert knowledge in event feature extraction. The entire attack extraction and attack chain construction after that can be performed automatically by the computer.
2) T-trace can perform attack provenance analysis by correlating multiple syslogs to uncover unknown attack chains in APTs. This method circumvents the reliance on training datasets, marking a significant breakthrough compared to traditional learning-based approaches in tracing APTs.
3) T-trace significantly reduces the dependency explosion problem in constructing the concise provenance graph, and yields compact scenario graphs all without missing any significant attacker activity for further proactive threats prevention. 4. T-trace achieves a 38% higher accuracy rate than previous work in identifying specific events, while also significantly reducing time expenditure by 90%. Furthermore, it attains a 92% accuracy rate in generating concise dependency graphs for alert events.

This paper is structured as follows: in Sections II and III, we give the motivation scenario and related work. In Sections IV and V, we introduce T-trace and elaborate on it in event identification and attack provenance. We show the evaluation performance in Section VI. Finally, we conclude the paper and discuss future work.

II. Motivation Scenario

In this section, we present a running example going through the paper and discuss the main challenges in our approach. This example demonstrates an attack carried out by FireEye Mandiant Red Team [13], which is also a part of our research data source.

The example includes the outdoor (Initial Reconnaissance and Initial Compromise) and indoor (Establish Foothold, Escalate Privileges, Internal Reconnaissance, Move Laterally, Maintain Presence and Complete Mission) attack activities.

Details are illustrated in Fig. 1. Through a phishing link, the attacker penetrates into the system of an organization and steals data from the local network. Although an expert with

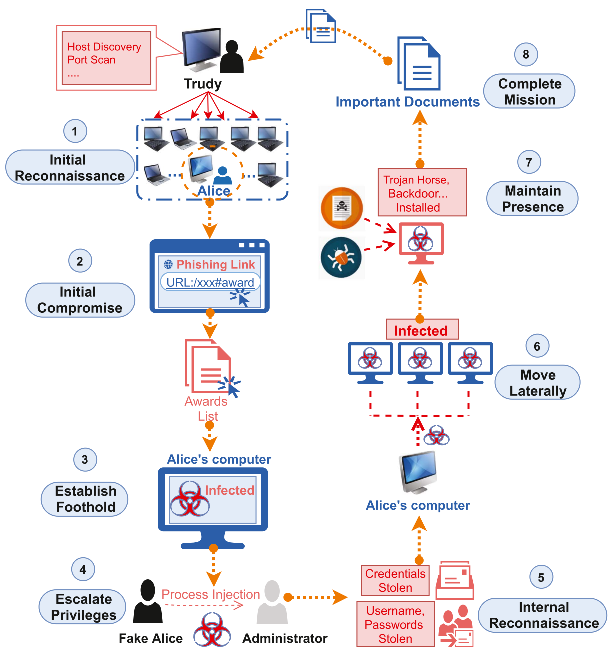


Fig. 1. Running Example.

intense security awareness protects the network from attacks such as DDoS, replay attacks and wormhole attacks, the stealthy adversary Trudy may exploit breaches of vulnerable hosts from the ones deployed by enterprises.

Afterward, Trudy accesses the network and remains inside without being noticed. Suppose that when Alice is using the office computer to work, she receives a message claiming that she can view the list of winners of the company activities this month. Alice then downloads these documents without noticing the malware embedded by Trudy. Hence, Trudy compromises Alice's host without launching any attacks immediately, but enhancing his authority via credential dumping instead. Later on, Trudy illegally accesses the company computer system and exploits the Server Message Block (SMB) protocol vulnerability. He embeds Trojans in the system and leaves backdoors in these systems. The malicious activities (e.g., data interception) can be launched through these backdoors.

This example illustrates three key challenges described below:

Needles in haystack: In this scenario, the crime clues we can get are huge log data. The data volume is too large to be able to return useful information. The first challenge, of course, is sifting through the huge volumes of log data that come into the system and iteratively identifying events and correlations.

Compromise under normal operation: APTs can effectively bypass existing IDS, and syslogs often fail to issue warnings about adversaries' actions, particularly before the actual attack occurs. Most logs in the system do not contain entries that say *Help! Help! I am being attacked!* Instead, the logs show *successful login* from a user. Nothing is happening on the surface of your computer, but hidden malicious attacks are actually occurring.

Attack provenance: The relationships among the events are complex in APTs. Determining the Tactics, Techniques and Procedures (TTPs) used by adversaries and constructing the provenance graph are challenging for the reason that there is often no roadmap that can be used to hunt. Many adversaries start to penetrate the system before the network is paralyzed. Thus, only detecting the final attacks within the system is not enough and constructing the whole provenance graph is another challenge in our work.

To overcome these challenges, we need to refer to the multiple source logs in the system. In the running example, Trudy used a phishing link to invade Alice's host. However, a single kind of log will not contain all the attack clues. We need multiple logs to build a complete attack intrusion path. And in the later attack stage, it often involves the conversion of multiple attack forms and attack targets. In order to find the crime clues completely, we can extract the relationship between multiple log entries by tensor decomposition, then the weight distribution and community discovery function use this relationship to extract the high cohesion between the events transformed by the log entries, mine implicit relationships among events,and extract the event set of the log that may be missed by the detected system. Finally, reconstructing the event chain by time sequence or use other ways to construct the whole attach provenance graph.

Despite blending seamlessly into benign background activity, two factors stand out regarding the attack. First, the adversary's meta-actions still belong to the event printed by the logs in the system. Second, these events have more relations among themselves and comprise the whole provenance graph. In the next section, we describe the T-trace approach based on these two key observations.

TABLE I
APPROACHES COMPARISON

Approaches	[26]	[22]	[4]	[11]	[15]	T-trace
Raw Logs Input						✓
Event Identification	✓	✓		✓		✓
Use Train Dataset	✓		✓	✓		
Event-level Provenance	✓	✓		✓	✓	✓
Solve DE Problem	✓		✓		✓	✓
Universal Attack Cycle			✓			✓

III. RELATED WORK

To our best knowledge, T-trace is the first to construct the APTs' provenance graph from raw logs with causal and temporal correlations and solve the dependency explosion problem without the knowledge-based learning method. A comparison of our approach with prior work is outlined in Table I.

APTs detection: Existing work [5], [6], [7] have emphasized the importance of the APTs detection. NoDoze [15] relies on anomalous events and PrioTracker [16] depends on anomalous information flows, stealthy attackers can leverage novel malware or unknown approaches to evade the detection. OmegaLog [4] constructs a Universal Provenance Graph (UPG) with application logs, causally relating events across applications that cannot be used in the normal detection system. SLEUTH [22] and MORSE [14] leverage tags, which encodes an assessment of

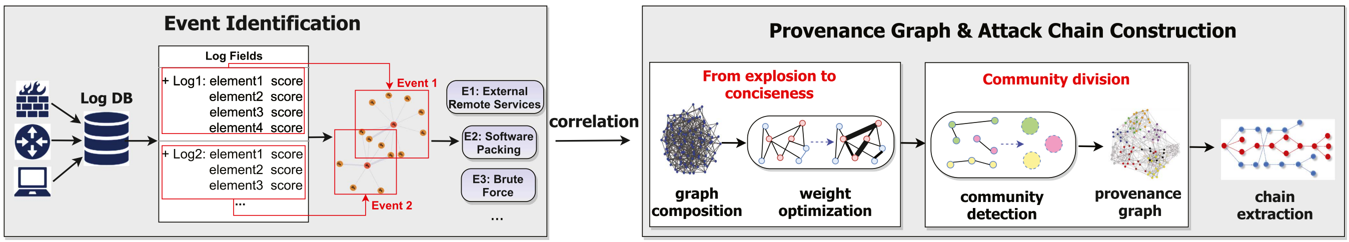


Fig. 2. System Model of T-trace.

trustworthiness and sensitivity of data, to detect the attacks, but in the APTs scenario we cannot derive the tags from the raw audit logs.

Provenance with logs: HOLMES [27] and UISCOPE [26] employ homogeneous logs to detect and track APTs campaigns. UISCOPE [26] leverages Event Tracing for Windows (ETW) logs that can only trace the events within the system. Log2vec [28] detects insider threats with system audit logs. These approaches focus on the internal threats in the system but ignore the attempt before breaching the system. Many approaches [10], [11], [12] use representative training data with sequence processing techniques, e.g. deep learning and recurrent neural networks, to learn from past events and do the attack provenance. However, we need to trace the attack without mainly depending on the expert knowledge due to the unknown APTs.

Attack graph analysis: The attack graph enumerates the possible attack paths from the attacker's point of view which can help the defender to intuitively understand the tactics of the adversaries. With the development of offensive and defensive technologies and practical needs, state attack graphs [29], attribute attack graphs [30], and Bayesian attack graphs [31] have been gradually developed. However, the attack graph is not suitable for modeling and analyzing concurrent and collaborative attack processes, and the state explosion problem is prone to occur during the generation of the attack graph, which makes the scale of the attack graph too large.

IV. Approach Overview

T-trace consists of two main steps: *Event Identification* is to abstract events from multiple logs and *Provenance Graph & Attack Chain Construction* can trace the APTs by constructing the provenance graph and attack chain, which is shown in Fig. 2. To gain the framework of T-trace, we introduce each component in this section.

In the event identification, there are three main steps: *Log Preprocessing*, *Log Tensor Decomposition* and *Event Abstraction*. First, the syslogs, firewall logs and network flow data are collected through system audit tools and packet capture tools. T-trace extracts the log fields with regular expressions and forms the structured data clusters to pre-process these data. The feature extraction part relies on manual efforts and expert knowledge and we indeed need to combine the human efforts in this field to make our system more concise. According to our experience and data experiments, what we really combat in APTs are the big attack organizations. For different and complex systems, T-trace can set scalable characteristics. The feature extraction part relies on manual efforts and expert knowledge. We need to set appropriate log characteristics and characteristic relationships, such as process number, port number, process parent-child relationship, and time interval threshold setting. Afterward, we give an enhanced-Louvain community discovery algorithm that can automatically complete the division and extraction of attacking communities. Because of the higher complexity of APT attacks, we indeed need to combine some human knowledge to make sure our system is intelligent. However, the workload of this part is relatively small. The entire attack extraction and attack chain construction after that can be performed automatically by the computer. Moreover, after the feature is established, the subsequent feature expansion and supplementation require little manual participation, and the manual feature setting greatly increases the effectiveness of attack provenance. Next, T-trace builds a rank-three tensor model by extracting the required message templates, time, and processes from the original logs. The event sub-tensor can be obtained by tensor factorization. Then, to abstract the events from logs, a saliency scoring algorithm is used. Utilizing a saliency scoring algorithm, T-trace performs pairwise comparisons of log elements and calculates a saliency score for each pair. It then constructs an event graph by connecting log elements with scores surpassing a predetermined threshold. This event graph effectively illustrates the behavior of an event.

For the provenance graph & attack chain construction phase, T-trace takes three steps to retrieve the intricate relations among the events. In order to find out the relationship between events, we propose an *Enhanced Louvain Algorithm* for discovering community and perform weight optimization operations on feature vectors, which can effectively solve the state explosion problem. This method connects separate event entries, generates the attack chains, and makes a reasonable analysis of the event sequence. Although it is based on a heuristic algorithm, the log data format has a certain generality. Log information such as timestamp, process id, IP address, port number, etc., have the same form in different systems. We have set up an extensible feature construction method, so for different attack scenarios, T-trace still has efficient traceability. The approach consists of the following three steps:

- *Stage 1:* From explosion to conciseness. There are multiple relationships among events in terms of the various features. For example, the event occurs in the same period, and the process between the events has a parent-child relationship.

TABLE II
MEANING OF THE SYMBOLS USED IN T-TRACE

Tensor	
$\boldsymbol{X}$	A rank-3 log tensor.
$\boldsymbol{A}$	Process Message group Matrix. $a_{tq} \in \boldsymbol{A}$, a_{tq} denotes the proportion of specific message templates to process message groups.
$\boldsymbol{B}$	Event tensor matrix. $b_{pqe} \in \boldsymbol{B}$, b_{pqe} denotes the proportion of process message groups in the event.
$\boldsymbol{C}$	Time Matrix. It represents the time relationship between events.
Event Identification Parameters	
T	Message templates. $t \in T$, Each t represents a single message template.
W	Time windows. $w \in W$, Each w represents a time stamp.
Q	Process message groups. $q \in Q$, Q represents a collection of concurrent message templates in a process.
P	Processes. $p \in P$, P represents the set of processes associated with the events.
E	Events. $E = (p_i, q_j)\|p_i \in P, q_j \in Q$. Events are tuples of related processes and process message groups.
Provenance Graph Parameters	
V	A set of nodes representing event entries.
E	The edge set formed by the relationship between event entries.
D	The dimension of feature relationship.
G	a three-dimensional matrix M of $V \times V \times D$.
A, B	Node community.
P	Community partition.
P', P_τ	Refined partition.
P^*	The optimal community division of graph G
C, C_τ	Community in the partition.

Thus, the event calls the same file object. Vectorizing these features and associating each dimension in the vector to form an unweighted undirected relationship graph can initially show the relationship among events. However, in the composition process, because of various relationships between events, serious dependency explosion problems will occur. Therefore, we propose to use weight optimization to reconstruct the relationship graph.

- *Stage 2:* Community division. After the weights are optimized, we perform community detection on the weighted graph to classify events. During the detection process, the existing community discovery methods have the problems of badly connected communities and long detection time. We propose an *Enhanced Louvain Algorithm* for discovering community and we can get the provenance graph in this end.
- *Stage 3:* Chain extraction. After the community division is completed, we extract the attack-related communities. Then the approach converts the undirected graph into a directed network and generates the attack chains based on the event sequences, the parent-child relationships between the event processes, and the order of the event call objects.

TABLE III
LOG FIELDS

Field	Description
TimeStamp	The start time of event
Pid	Process ID
P-pid	Parent process ID
Pname	Process name
H-ip	Host IP address
D-ip	Destination IP address
H-port	Host port
D-port	Destination port
Path	absolute path of GET/POST
Objname	Object name
Ctype	Internet media type
Status	Http Status

V. SYSTEM MODEL

As we have studied hundreds of documents and notes related to APT [32], we have found that the events in the threats have two-dimension relationships: *causal* and *temporal* correlations. These relations in the logs are unfolded in current approaches. In this section, we elaborate on how to leverage the data clues left in the system to reflect the original actions launched by the attackers.

A. Overview of the Symbol Meaning

To make it easier for readers to read the formulas, in Table II, we give explanations for the symbols that appear more frequently.

B. Event Identification

In this section, we introduce how T-trace abstracts the events from logs. At present, machine learning is widely used to extract events from the raw logs, but it suffers the drawbacks of high learning time cost and requirements for professional knowledge. Using tensor decomposition to extract events can handle various raw logs, and it does not need to import training sets or related professional knowledge. However, tensor decomposition may not extract the events accurately due to its inherent high-frequency templates matching mechanism. Thus, we combine the saliency scoring algorithm with tensor decomposition to further find the relations among logs and extract the corresponding events.

1) Preprocessing: We collect the system logs, and networks (HTTP, DNS, UDP/TCP) logs through system audit tools and packet capture tools. As we know, these logs generally have information such as $Timestamp$, $Process\_name$ and $Execution\_path$. Next, the logs are put into the parsing functions, which can parse the logs by the regular expressions. Finally, the logs will be parsed into structured entities as Table III. Although feature extraction requires a certain amount of human effort, in the process of multiple attack detection, feature extraction is one-time, so the algorithm overhead is also low.

2) Case Study: To better understand the meaning and advantages of tensor decomposition, we present a PC backdoor

TABLE IV
PC BACKDOOR EXECUTION EVENT

Process	Message Template
remote control	restart_pc restart_APP delete_app screenShot kill_process
computer timer	upddateAPP DownAPP RequesTimer

execution event from APT-Q-63 for the case study. Such a backdoor has two main functions, namely the remote control function, and the timer function. To achieve remote control, the attackers collect user information and upload it to a designated server, from which they download files and execute them. During the remote control process, the attackers perform operations such as restarting the PC and implementing screenshots. In the timer function, the attackers add other actions such as RequesTimer to perform periodic remote control judgments. Each function is often completed within a process, which we refer to as process message groups. The process itself contains many atomic commands, which we refer to as message templates. Therefore, as indicated in Table IV, this backdoor execution event is composed of process message groups like remote control and computer timer, and each process message group is made up of message templates like restart pc. We discovered from the raw data that computer timers and computer restarts occur regularly. Additionally, there were a lot of process message groups that were triggered by other events. They consequently constituted a challenge in locating the network event. To extract target events under a specified timeline, tensor decomposition on complex log data is required.

3) Tensor Decomposition: To extract events from complex logs and capture their causal-temporal patterns, we apply tensor decomposition to them. Extracting causal-temporal patterns is an advantage of our method over traditional NTF [33], which can only model co-occurring log templates. The same process may occur in different timelines. However, only some processes are triggered by the target event. We regard each line of the original logs as a three-dimensional vector, which contains a message template, process and time window. The relationship between the event and original logs can be found by tensor decomposition which is shown in Fig. 3. Besides, the relationship between events and processes can be further found.

The tensor X is established using the message template $t \in T$ and time window $w \in W$ obtained from above. Meanwhile, we define $e \in E$ to represent events. The purpose of decomposition is to get the relationship between the message templates and events, and the relationship between the event and time. To achieve this purpose, we use the original message templates and time stamps which have no apparent relationships.

Definition 1 (Message Template): We define the fixed format of each line in the original logs as its message template. Each

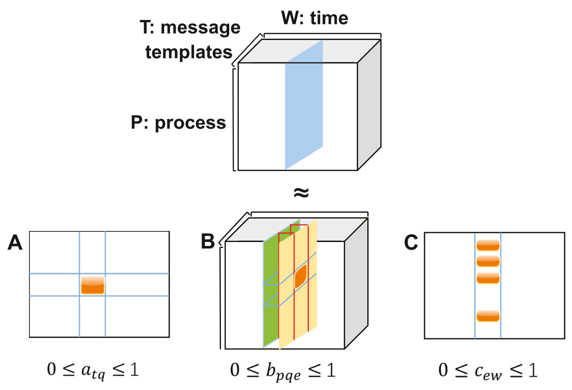


Fig. 3. Tensor Decomposition. The target tensor X is factorized into a Process Message Group matrix A, an Event rank-3 tensor B, and a Time matrix C.

message template represents the smallest atomic behavior in the log, such as *restart_pc* in Table IV is a message template.

Definition 2 (Process Message group): $q \in Q$ represents a process message group which means a collection of concurrent message templates in a process. From the definition, the Process message group is considered to be a causal extension of the message template. The same message template will appear in different processes. For example, *restart_pc* in Table IV will appear not only in the remote control template group but also in daily processes such as software updates. so it is necessary to extract the relationships between the process and logs.

Definition 3 (Event): Events are composed of various process message groups that occur on different processes. $E = \{(p_i, q_j)|p_i \in P, q_j \in Q\}$, where the symbol Q is the set of Process message groups obtained from the decomposition of X, and the symbol P is the set of processes associated with the events. The log generated by the process is large, and the target event is composed of some of the logs. For example, *PC backdoor execution* event corresponds to multiple process groups including *remote control* and *computer timer*, which in turn corresponds to multiple message templates.

The essential idea to establish the third-order tensor based on the original logs is to treat the log data of each time as layered superposition. According to this idea, T-trace establishes a rank-three tensor $X = T \times P \times W$, where T, P, and W represent the three dimensions of the tensor X. The value of X represents the number of message templates t_i that appear on a process p_i in a time window w_i, $x \geq 0$. In this case, we assume a matrix $A = T \times Q$, a small rank-three tensor $B = Q \times P \times E$, and a matrix $C = E \times W$. Matrix A is called the Process Message group Matrix and is used to capture the correspondence between Process message groups and Message Templates. Matrix B is called the event tensor matrix and is used to represent the proportion of the process message group with the same type in all process message groups under the target event. Matrix C is called Time Matrix and is used to represent all events that occurred at a specific time and represents the time relationship between events. X is a third-order tensor of $T \times P \times W$, we expand the above formula

$$a_t = \{a_{t_1q}, a_{t_2q}, a_{t_3q,...,}a_{tq}\}$$

$$b_t = \{b_{p_1qe}, b_{p_2qe}, b_{p_3qe,\ldots,b_{pqe}}\} \tag{1}$$

$$\left(\sum_{q=1}^{Q} a_t \otimes b_{qe}\right)_{tp} = \sum_{q=1}^{Q} a_{tq} \times b_{qpe} \tag{2}$$

The formula of a_t and b_t are shown in (1), which can derive the following (2). Besides, w_k is defined in (3) which can deduce the (4)

$$w_k = \{w_{j_1k}, w_{j_2k}, \ldots, w_{Jk}\} \tag{3}$$

$$\left[\sum_{q=1}^{Q} a_t \otimes b_{qe}\right] \otimes c_e = x_{tpw} \tag{4}$$

Our purpose is to satisfy $X{=}A\times B\times C$, that is

$$x_{tpw} = \sum_{q=1}^{Q}\left(\sum_{e=1}^{E} a_{tq} \times b_{qpe}\right) \times c_{ew} \tag{5}$$

Definition 4 (Process Message group Matrix A): If a specific type of message template belongs to a process message group, then $a_{tq} \geq 0$. Its value represents the proportion of this type of message template in all message templates under this single process. If a process message group p_i consists of several message templates, the number of message templates is n, and the message templates t_i appear j times, then $a_{tq} = \frac{j}{n}$.

Definition 5 (Event Tensor Matrix B): If a message group X occurred on a process belongs to event Y, then $b_{qpe} \geq 0$. Its value represents the proportion of the process message group with the same type as X in all process message groups under event Y. If n process message groups are composed of several "process message group", and there are j times of (p_i, q_j) occurrence, then $b_{qpe} = \frac{j}{n}$.

Definition 6 (Time Matrix C): It represents all events that occurred at a specific time and represents the time relationship between events.

4) Sub-Tensor Updating: After tensor decomposition modeling, the key is how to find the most suitable A, B and C, which can satisfy $X \cong A \times B \times C$. To intuitively evaluate the consistency between the known value and the ideal value of A, B and C, we define an evaluation formula. The closer the evaluation result is to 0, the higher the degree of agreement with the ideal result can achieve.

Evaluation formula: The calculation time may be complicated because of the repeated usage of the evaluation formula. To simplify the operation, we add an auxiliary function Y based on log simplification operation to update the sub tensors A, B and C, respectively. Tensor decomposition extracts events through iterative functions, and the iterative functions at each step need to call the evaluation formula. This is the key factor in reusing the evaluation formula

$$J = \frac{1}{2}\sum_{t=1}^{T}\sum_{p=1}^{P}\sum_{w=1}^{W}\left(x_{tpw} - \sum_{e=1}^{E}\left(\sum_{q=1}^{Q} a_{tq} \times b_{qpe}\right) \times c_{ew}\right)^2 \tag{6}$$

Auxiliary function Y

$$Y{=}\sum_{(t,p,w)\in X}\Bigg(x_{tpw}{\times}f{\times}log\frac{1}{\sum_{e=1}^{E}(\sum_{q=1}^{Q} a_{tq} \times b_{qpe}) \times c_{ew} \times f} - x_{tpw} + \sum_{e=1}^{E}\left(\sum_{q=1}^{Q} a_{tq} \times b_{qpe}\right){\times}c_{ew}{+}x_{tpw}{\times}log x_{tpw}\Bigg) \tag{7}$$

where f is

$$f = \frac{a_{tq}b_{qpe}c_{ew}}{\sum_{e'=1}^{E}\sum_{q'=1}^{Q} a_{tq'}b_{q'pe'}w_{e'w}} \tag{8}$$

Then the partial derivatives of A, B, C of auxiliary function Y are calculated respectively to obtain the renewal formula

$$f_{tqpew} = a_{tq}b_{qpe}c_{ew} \tag{9}$$

$$g_{tpw} = \sum^{Q,E'} a_{tq'}b_{q'pe'}c_{e'w} \tag{10}$$

$$a_{tq} := \sum^{P,W,E}\frac{\breve{f}_{tqpew}}{\breve{g}_{tpw}}x_{tpw}\frac{1}{\sum^{P,W,E} b_{qpe}c_{ew}} \tag{11}$$

$$b_{qpe} := \sum^{T,W}\frac{\breve{f}_{tqpew}}{\breve{g}_{tpw}}x_{tpw}\frac{1}{\sum^{T,W} a_{tq}c_{ew}} \tag{12}$$

$$c_{ew} := \sum^{T,P,Q}\frac{\breve{f}_{tqpew}}{\breve{g}_{tpw}}x_{tpw}\frac{1}{\sum^{T,P,Q} a_{tq}b_{qpe}} \tag{13}$$

When the value of the evaluation function J is lower than the preset threshold, the decomposed sub-tensor can be derived. According to the process message group matrix, the relationship between a process message group and message template is established. According to the event tensor, the corresponding relationships between the event and process, and relationships between the event and process message group are established. According to the time matrix, the relationship between event and time is established. At this time, the extracted events can be exported in chronological order, and each event contains its constituent logs. The composition log is composed of simplified processing according to the process message group, process, and message template within the process message group. Finally, the extracted network or system events should be mapped to several message templates, such as the PC backdoor execution event in Case Study.

5) Event Abstraction: According to our observations and previous work [34], the logs in an event are often closely related. Thus, we can further divide events by calculating the correlations between the logs in the single process event. Tensor factorization first finds out whether there are common elements among logs and use the frequency of repeated elements as the relations between logs. To qualify the correlations better, we use a significance score to measure the correlation strength between logs To distinguish useful relations from useless ones. We calculated the significance score of a term by comparing the frequency with which the term appears in the foreground set and the background set. The foreground set is also called

```
Algorithm 1: Saliency Scoring Algorithm.
  Input: Elements sets in log fields $E$, event threshold $T_e$,
  relation threshold $T_r$.
  Output: Event $G$.
 1: Given a set of elements $E = e_1, \ldots, e_n (1 < i \leq n)$.
 2: for $e_i \in E$ do
 3:     Figure out the significance score of $e_i$ as $s_i$.
 4:     for $e_j \in E (j \neq i)$ do
 5:         Figure out the significance score between $e_i$ and
            $e_j$ as $s_{ij}$.
 6:         if $s_{ij} < T_r$ then
 7:             Continue.
 8:         else
 9:             Add edge $(e_i, e_j)$ into $G$.
10:         end if
11:     end for
12:     if $s_i < T_e$ then
13:         Continue.
14:     end if
15:     Add point $e_i$ into $G$.
16: end for
17: return $G$.
```

the search result set, which represents the collection of all logs related to this event, and the background set is also called the set of search results in the index set by the user, which is related to the user, or a collection of logs that the user cares about. If the two frequencies are very different, the term is considered important. For example, if a term "port" only exists in 5 documents in a 10 million document index and yet is found in 4 of the 100 documents that make up a user's search results, we may consider the term is of great importance, because 5/10,000,000 vs 4/100 is a big swing in frequency. The expression of the significance score is as follows:

$$Score = \begin{cases} (p_{fore} - p_{back}) \frac{p_{fore}}{p_{back}} & p_{fore} - p_{back} > 0 \\ 0 & \text{else} \end{cases}$$
$$\text{where} \quad p_{fore} = \frac{n_{t_1}}{n_d}, \quad p_{back} = \frac{n_{t_2}}{n_d} \tag{14}$$

Among them, p_{fore} is the probability that the element appears in the foreground set (search result set). It is calculated by the appearance times of term in logs n_{t_1} and the numbers of logs n_d. p_{back} is the probability of occurrence in the background set (the set of search results in the index set by the user). It is figured out by dividing the appearance times of a term in the user's index n_{t_2} and n_d. The higher the score, the more important the term. Based on this, we designed an algorithm to calculate the significant correlations between logs.

Algorithm 1 describes the process of abstracting the events from logs with a saliency scoring algorithm. For the score of each element, we remove the one with a low score, which means that the element has little influence. For the score of the connection between elements, we find the set of elements that are most closely connected, map it to the log, and sort out the events corresponding to the log. The determination of the thresholds can affect the accuracy of the results and is shown in the evaluation part. By repeating the operation, we can get the event groups in the end.

TABLE V
MULTIPLE RELATIONSHIPS OF EVENTS

Dimension	Description
d1	(u.TimeStamp - v.TimeStamp) <t
d2	u.Pid = v.Pid
d3	u.P-pid = v.P-pid
d4	u.Pname = v.Pname
d5	u.H-ip = v.H-ip
d6	u.D-ip = v.D-ip
d7	u.H-port = v.H-port
d8	u.D-port = v.D-port
d9	u.Type = v.Type
d10	u.Status = v.Status
d11	u.Objname = v.Objname
d12	u.Path = v.Pname
d13	u.Path = v.Objname

C. Provenance Graph and Attack Chain Construction

In this section, we elaborate our approach on how to analyze the intricate relationships among events and construct the provenance graph and attack chain.

1) Graph Composition: After extracting fields from the events, we can establish the relational connection as shown in Table V.

- *d1:* Events within the same period are often correlated, so we set the connection of relationships in this dimension through time range.
- *d2–d4:* During the attack, there is often a relationship between events with the same process information, so we establish connections with the same id and process name.
- *d5–d10:* Through the analysis of the network packets during the attack, we found that there is a strong relationship between events with the same IP, port, transmission type, transmission status and other information.
- *d11–d13:* Name and path reflect the access of process and network path to the file object. If a networking event u downloads a malicious program and another process v executes the program, u and v should belong to the same attack chain. Or if process A created a malicious object obj, and process B accessed obj, then there should also be an attack connection between process A and B.

For different types of attacks, we can extract other features as shown in the above example to establish more correlations. Next, we leverage the formatted data and connect the nodes through the feature relationship of log entries to form an undirected graph. In the graph $G(V, E, D)$, V is a set of nodes representing event entries, E is the edge set formed by the relationship between event entries, and D is the dimension of feature relationship. Finally, G forms a three-dimensional matrix M of $V \times V \times D$. Then $M_{i,j,k} = 1$ indicates that there is a k-dimensional relationship between node i and node j, otherwise $M_{i,j,k} = 0$. The edge e in E can be expressed as $\{(i, j, d, d_2, d_3 \cdots dn) \mid i, j \in v, d_k \in D\}$.

2) Weight Optimization: Due to the constructed intricate relationship between the event entries, there will be a very obvious dependency explosion problem among the events. To solve this problem, we proposed a method to assign the weights for the features by using Logistic Regression to optimize the weights. The weight vector obtained through Logistic Regression can reduce the weight of irrelevant relationships between events, so that we can better divide the community.

We divide the event entries into two communities, A and B, where A represents the attack-related community and B represents the attack-unrelated community. To better distinguish attacking and other communities, we need to get $|e_A| \gg |e_{AB}|$ and $|e_B| \gg |e_{AB}|$, where e_A and e_B represent the edges within communities A and B, and e_{AB} is the edge between the two communities. So, we use Logistic Regression to find a weight vector $\vec{\alpha}$, where the weight relationship can satisfy the following inequality conditions

$$\sum_{e \in e_A} \vec{\alpha} \cdot e_i \gg \sum_{e \in e_{AB}} \vec{\alpha} \cdot e_i$$
$$\sum_{e \in e_B} \vec{\alpha} \cdot e_i \gg \sum_{e \in e_{AB}} \vec{\alpha} \cdot e_i \tag{15}$$

We assume that there are m training edges, $E = (x_i, y_i)$, $i \in [1, m]$. Here, x_i represents the i-th training vector e_i, if $e_i \in e_{AB}$ then $y_i = 1$, otherwise $y_i = 0$. In order to prevent negative weights after weighting edges with $\vec{\alpha}$, we will map the weight range to [0,1] through S function

$$w = S\left(\sum_{i=1}^{k} \alpha_i \cdot e_i\right) = \frac{1}{1 + e^{\sum_{i=1}^{k} \alpha_i \cdot e_i}} \tag{16}$$

Then our constructor g will be as follows:

$$h_{\vec{\alpha}}(x_i) = g\left(\alpha^T x_i\right) = \frac{1}{1 + e^{-\vec{\alpha} x_i}} \tag{17}$$

We use $h_{\vec{\alpha}}(x_i)$ to denote the probability of $(e_i \in e_{AB}, y_i = 1)$, then the other is $1 - h_{\vec{\alpha}}(x_i)$. Finally we use the log-likelihood method to minimize the cost function to get the weight vector we need

$$\begin{aligned} cost = & -\frac{1}{m}\left[\sum_{i=1}^{m} y_i \log h_{\vec{\alpha}}(x_i)\right. \\ & \left. + \sum_{i=1}^{m} (1 - y_i) \log\left(1 - h_{\vec{\alpha}}(x_i)\right)\right] \end{aligned} \tag{18}$$

Next, we can bring the weight vector into the feature dimension vector, weight each feature in the connected graph, and reconstruct the graph to solve the impact of the dependency explosion on community detection.

3) Community Division: After the weight setting of the previous stage, we can classify the nodes through community detection methods to get provenance graph. Then we can find out the attack-related communities from the provenance graph.

We define A_{ij} as the weight of the edge between node i and j, $k_i = \sum_j A_{ij}$ represents the sum of the weights of all edges connected to node i. Here, c_i is the community of node i, and $m = \frac{1}{2} \sum_{ij} A_{ij}$ is the sum of all edges.

Algorithm 2: Local Movement.

```
Input: Graph $G$.
Output: Partition $P$.
1: $P \leftarrow$ INITIAL$\{G\}$.
2: $Q \leftarrow$ QUEUE$\{V(G)\}$.
3: while $Q$ is not empty do
4:     $v \leftarrow Q$.pop().
5:     $best\_commnuity \leftarrow$SelectBestCommunity().
6:     $N \leftarrow \{u|(u,v) \in E(G), u \notin$
       $(best\_community \cup Q)\}$.
7:     $Q$.push($N$).
8: end while
9: return $P$.
```

Then the definition of modularity is as follows:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m}\right] \delta(c_i, c_j)$$
$$\delta(u, v) = \begin{cases} 1 & \text{when } u == v \\ 0 & \text{else} \end{cases} \tag{19}$$

Louvain algorithm divides communities based on maximizing modularity, the local movement of which has low efficiency. Besides, there are badly connected communities in the community division. To solve the problems, we propose an *Enhanced Louvain Algorithm*. To be more specific, the algorithm consists of the following three steps.

Step 1. Local movement: In local movement, it can be concluded that after node i moves from A to another community B, there are four cases of nodes that may have community movement in the following steps.

1) The neighbor node of node i is not in community B.
2) The neighbor node of node i is in community B.
3) Nodes that are not connected to node i in community A.
4) Nodes that are not connected to node i in community B.

We know that when node i moves, it has the most impact on the neighbor nodes of i, so the neighbor nodes of i are added to the queue first, and the nodes in community A connected to the neighbor nodes of i will also be considered when the neighbor nodes dequeue. Therefore, it is more important to consider the first of the above four situations when the node moves. Through experiments, it is found that we can only consider the first case of movement during the local movement. This can greatly reduce the computing time and has little impact on the partition effect.

In Algorithm 2, we initialize a partition P, in which each node is initialized as a singleton community. Then we set an empty queue Q. The nodes are added to the queue in random order. We take the node from the head of the queue and determine whether the node should be moved to another community through the quality function. If the node is moved, all neighbor nodes that do not belong to the new community and have not joined the queue will be added to the queue. Then we will continue to remove the node from the queue for the move operation until the queue is empty.

Step 2. Partitions refining: Because the node movement only considers the movement of a single node, it may produce the

Node	Event	Node	Event
0	Paste text	8	Move document
1	Move document	9	Create document
2	Rename document	10	Download document
3	Open document	11	Open document
4	Download document	12	Open browser
5	Open phishing email	13	Open code editor
6	Rename document	14	Create user account
7	Open document	15	Delete user account

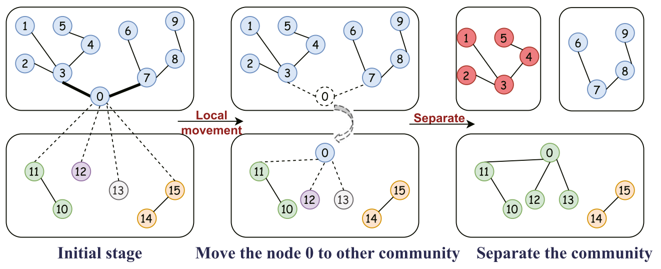


Fig. 4. Refining process: Nodes 1–5 are attack events, and other nodes are non-attack events. The nodes with the same color belong to the same community.

badly connected community, or even disconnect. Therefore, before the nodes aggregation, we should do a further analysis of the partition. Then we can perform node aggregation based on the refined partition to reduce the occurrence of the badly connected community.

As is shown in the initial stage of the Fig. 4, the thin edges indicate the weak relationship, and the thick edges indicate the strong relationship. In order to explain community refinement more intuitively, we give a simple example. In the upper table of Fig. 4, nodes(1-5) describe a simple phishing email attack event, the user opened the phishing email and downloaded the document, and nodes(6-9) are a normal document operation event of the user. The nodes(10-15) are other users' activities in other communities. The nodes(1-5) are connected to the nodes (6-9) through two edges, 3-0 and 7-0. All these nodes became a community. We assume that 3-0 and 7-0 are nodes 3 and 7 and node 0 belong to the same time threshold. If the ip and process id of node 0 have a stronger correlation with the colored node below, node 0 will move to the community below. If we only consider the movement of a single node, when node 0 moves to other communities, and no more nodes can move, it will aggregate the nodes. When the disconnected community has become an aggregation node in the graph, it will be no longer possible to split, so that the nodes(1-5) and the nodes(7-9) will form a disconnected community.

To solve this problem, we need to refine the community. After the local movement divides the community, we refine each community and then aggregate the nodes based on the refined community to avoid aggregating the badly connected communities into new nodes. In the refining process, the greedy movement is too restrictive to achieve the optimal division, but the selection of the community through randomness allows a wider exploration of the partition space. Thus, we allow nodes to randomly merge with any community that increases the degree of modularity. During the movement, the greater the increase in

Algorithm 3: Partitions Refining.

```
Input: Graph $G$ and Partition $P$.
Output: Refined Partition $P'$.
 1: $P' \leftarrow$ INITIAL$\{G\}$.
 2: for $C \in P$ do
 3:     $R \leftarrow \{v|v \in C, E(v, C-v) \geq \gamma||v|| * (||C|| - ||v||)\}$.
 4:     for $v \in R$ do
 5:         if $v$ has not been merged then
 6:             $T \leftarrow \{C'|C' \in C, C' \subseteq P', E(C', C-C') \geq \gamma||C'|| * (||C|| - ||C'||)\}$.
 7:             for $C' \in T$ do
 8:                 if $\Delta Q(v$ moves to $C') > 0$ then
 9:                     $v \rightarrow exp(\frac{1}{\theta}\Delta Q)$.
10:                 else
11:                     $Be\ a\ singleton\ community.$
12:                 end if
13:             end for
14:         end if
15:     end for
16: end for
17: return $P'$.
```

modularity, the greater the possibility of node movement. The random selection degree of the community is affected by the parameter $\theta > 0$.

In Algorithm 3, we initialize a refined partition P', in which each node is initialized as a singleton community. We define the recursive expansion size of the collection as $\|S\| = \sum_{s\in S}\|s\|$. For example, the recursive expansion of set $S = \{\{a,b,c\},\{d,e\},\{f\}\}$ can be expressed by the following Equation

$$\|S\| = (\|a\| + \|b\| + \|c\|) + (\|d\| + \|e\|) + (\|f\|) \tag{20}$$

We define a relational expression as follows to describe the connectivity of the nodes and communities

$$E(S, C-S) \geq \gamma\|S\| \cdot \|C-S\| \tag{21}$$

The nodes and communities that satisfy (21) are well-connected nodes and well-connected communities. We first select a set of well-connected nodes. For the nodes that are not merged in the set of well-connected nodes, we select well-connected communities to move them in. During the move-in process, nodes are moved randomly according to the increase in modularity. Finally, we can obtain a refined partition so that we can aggregate nodes based on the refined partition. The time complexity of Partitions Refining does not exceed $O(V+E)$.

Step 3. Nodes aggregation: After partitions refining, we need to aggregate the community into a new node. The edge weight between nodes in the community is converted to the self-loop weight of the new nodes. The edge weight between the communities is converted to the edge weight between the new nodes. Then, the program will re-execute the local movement.

4) Chain Extraction: Based on the attack-related community obtained, T-trace constructs the attack chain of the attack-related

Algorithm 4: Chain Extraction Algorithm.

```
Input: Attack-related community $C$ from Partition $P$.
Output: $Attack\_chain$.
1: for $event\_i, event\_j \in C$ do
2:     if $event\_i.id$ is the parent of $event\_j.id$ then
3:         directed_edge.add $(event\_i, event\_j)$.
4:     end if
5:     if $event_i.obj = event_j.obj$ && time$(event_i.obj)$ < time$(event_j.obj)$ then
6:         directed_edge.add $(event\_i, event\_j)$.
7:     end if
8: end for
9: $Attack\_chain \leftarrow$ SORT{$directed\_edge$} in timestamp.
```

community. Leveraging the chronological order, the correlation between the process IDs, and the order in which the process calls the target file, T-trace extracts the attack chain after community discovery and the whole process is shown in Algorithm 4.

D. Theoretical Analysis of Provenance Graph Construction

The core of the performance optimization of our solution lies in the community division, and we specially provide the following proofs for the guarantee of correctness in the provenance graph construction.

Definition 7: Define $P_0, \ldots, P_\tau$ as the partition of graph $G(V,E)$. There is a partition iteration sequence $t = 0, \ldots \tau - 1$ such that $P_{t+1} = P_t(v_t \to C_t)$, where $v_t \in V, C_t \in P_t$. Define $\mathrm{H}(P)$ as the modularity of the partition, $\Delta\mathrm{H}(v \to C)$ refers to the change of modularity after node v moves to community C. Define that when $\mathrm{H}(P_{t+1}) \geq \mathrm{H}(P_t)$, the sequence is called a non-decreasing moving sequence.

Theorem 1: For graph $G(V,E)$, P^* is the optimal community division of graph G. There is a non-decreasing sequence $P_0, \ldots P_\tau$, where $P_0 = \{\{v\} | v \in V\}$, $P_\tau = P^*$, and $\tau = n - |P^*|$. which can make it reach the best partition within n times. The optimal partition can be achieved in $n - |P^*|$ steps.

Proof: Let C^* be a community in the optimal partition P^*, where $\nu_0 \in C^*$ and $C_0 = \{v_0\}$. Let $P_0 = \{\{v\} | \nu \in V\}$ be the singleton partition. For $t = 1, \ldots, |C^*| - 1$, let $v_t \in C^* - C_{t-1}, C_t = \{v_0, \ldots, v_t\} \in P_t$, and $P_t = P_{t-1}(v_t \to C_{t-1})$. By contradiction, it can be proved that there is always a non-decreasing sequence $P_0, \ldots P_{|c^*|-1}$.

Suppose that there is a node v_t for t that does not satisfy $\Delta\mathrm{H}(v_t \to C_{t-1}) \geq 0$. Let $S = C^* - C_{t-1}, R = C_{t-1}$, for all nodes $\nu \in S$ satisfies

$$E(v,R) - \gamma\|v\| \cdot \|R\| < 0 \tag{22}$$

That is

$$E(S,R) = \sum_{v \in s} E\left(v \cdot R\right) < \gamma\|S\| \cdot \|R\| \tag{23}$$

However, according to formula $E(S, C - S) \geq \gamma\|S\| \cdot \|C - S\|$, for all $S \subseteq C^*$, $R = C^* - S$

$$E(S,R) \geq \gamma\|S\| \cdot \|R\| \tag{24}$$

Therefore, there is always a non-decreasing sequence $P_0, \ldots P_{|c^*|-1}$, making $C_t = C^*$. The above conclusion can be applied to each community $C^* \in P^*$. For each community movement $C^* \in P^*$, it can be completed within $|C^*| - 1$ step, so the total movements τ is in $\sum_{C^* \in P^*} |C^*| - 1 = n - |P^*|$.

Theorem 2: For graph $G(V,E)$, let $P_{t+1} = Enhanced\_Louvain(G, P_t)$. Then exists τ such that when $t \geq \tau, P_t = P_\tau$.

Proof: In the community division, if $P_t \neq P_{t+1}$, when $t' \leq t$, $P_{t+1} \neq P_{t'}$. Suppose there is no τ, such that when $t \geq \tau$, $P_t = P_\tau$. This means that for any τ, there is a $t > \tau$ such that for all $t' < t$, $P_t \neq P_{t'}$. That is, the sequence $P_0, P_1, \ldots$ is infinite. However, the partitions of graph G must be finite, so τ must exist, so that when $t \geq \tau$, there is $P_t = P_\tau$. So we can prove that the algorithm is asymptotically stable.

Theorem 3: For graph $G(V,E)$, let $P_{t+1} = Enhanced\_Louvain(G, P_t)$. When $P_t = P_{t+1}$, P_t is the optimal partition of nodes.

Proof: After the algorithm iteration is stable, $P_t = P_{t+1}$. The following proof is given by contradiction. Assume that when $P_t = P_{t+1}$, P_t is not the optimal partition of nodes. Then there is a node $v \in C \in P_t$ and a community $D \in P_t$, such that $\Delta\mathrm{H}(v \to D) > 0$. In this way, in the $Local\ Movement$ step, node v will be moved to community D, namely $P_t \neq P_{t+1}$. This is contrary to the iterative stability of the algorithm. Therefore, when $P_t = P_{t+1}$, P_t is the optimal partition of nodes.

VI. Performance Evaluation

In this section, we evaluate each component and the overall performance of T-trace to show its effectiveness in identifying the events, by constructing provenance graph and attack chain from large system log data. We compare T-trace with current APTs provenance approaches on open source data sets to examine the efficiency and accuracy of our approach. Our evaluation addresses the following research questions [15][23].

Q1. How about the performance of event recognition? (Section VI-B)

Q2. How to determine the threshold in event abstraction? (Section VI-C)

Q3. Compared with other existing community discovery methods, what is the performance of T-trace's community division? (Section VI-D)

Q4. Based on the attack community obtained after community division, what is the performance of T-trace to construct a complete attack chain? (Section VI-E)

Q5. What is the runtime overhead of T-trace to perform provenance graph construction? (Section VI-F)

A. Datasets and Set up

We evaluate T-trace with four launched practical APT attacks, and open source data sets are also applied to examine the efficiency and accuracy of our approach. In the experiment, the attack simulation environment is shown below.

- *Target:* We use the windows 7 operating system on the machine Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz 2.80 GHz.

TABLE VI
THE APTS IN EXPERIMENT

Attack Name	Tactics	CVE
Microsoft Office Memory Corruption Vulnerability	Email attachment	2017-11882
LNK Remote Code Execution Vulnerability	Email attachment	2017-8464
AdobeFlash Player Vulnerability	Controlled Website	2018-15982
Windows VBScript Engine Remote Code Execution Vulnerability	Email with file link	2018-8174

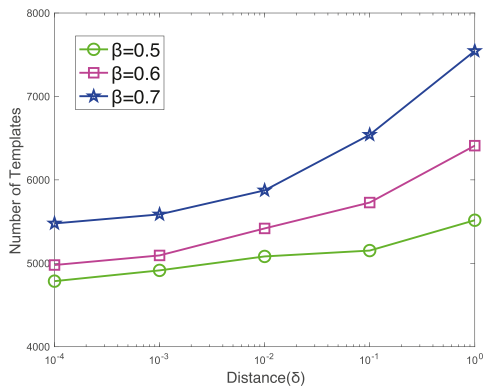


Fig. 5. Comparison of the number of extracted templates with different values of δ and β. δ is the distance threshold used in DBSCAN, and β is the proportion of template words among all words.

- *Attacker:* We use the Kali Linux 64-bit operating system on the machine Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz 2.80 GHz.

According to existing attack reports, we emulated four practical APT attacks. We show the attack name, compromise tactics, and types of CVE exploited in Table VI. To better restore the real-user scenario, our simulation environment also includes the benign activities, such as web browsing, and chatting.

We used the data set released by the DARPA Transparent Computing program [35], which was generated during the engagement between the red team and the blue team in May 2019. The attackers used whatever means available to them to test as much variation in capabilities as possible. The attackers set up different servers under a variety of systems, and also simulated a large number of normal user events during the attack.

We also use the CSE-CIC-IDS2018 Log Set [36] to examine the efficiency and accuracy of T-trace. The dataset contains benign events and the most up-to-date common attacks, which resembles the true real-world data. It also includes the results of the network traffic analysis with labeled flows based on the time stamp, source and destination IPs/ports, protocols and attacks.

B. *Performance of Event Identification*

We tested the LTF(Log Tensor Flow) model from both template extraction and event extraction. This shows that T-trace has high performance in the event identification process.

As for the template extraction ability, we use 500000 consecutive original network and system logs in Fig. 5. The test index is the effectiveness and accuracy of template extraction. Due to the complexity of the original network and system logs, it is difficult to judge the accuracy or effectiveness of the template. Based on this situation, we propose several indicators to replace the effectiveness and accuracy equivalently. First, we calculate the deleted words without numbers in the extracted templates. Through the analysis of 500 lines of original network and system logs, we observe that the words without numbers are often template words, such as host and path names. The words without numbers are about 1800 when β equals to 0.7 and δ equals to 0.0001 initially, and only about 1200 words without numbers are left when β equals to 0.7 and δ equals to 0.0001, and the extracted template is improved by about 600, which means that increasing δ will make the extraction better. However, we also find many non-template words, which also do not contain numbers. Therefore, the efficiency and accuracy of template extraction can be judged by calculating the deleted words without numbers in the extracted template, but the non-template words without numbers have a greater impact on the experimental results. Therefore, we calculate the number of extracted templates. The more templates are extracted, the more words are used as template words in the original logs. The fewer templates are extracted, the more words are deleted as parameters in the original logs. There are many advantages to using this index. One is that we can judge whether the template extraction is overfitted or not completely fitted by the number of extracted templates. The other is that we can adjust the parameters β and δ (β is the proportion of template words in all words, generally between 0.6 and 0.8. δ is a parameter in DBSCAN that affects the number of clusters). As shown in Fig. 5, The bigger β and δ are, the larger the number and the more templates are extracted, but too many or too few templates are not good. If there are too many templates, the speed of tensor decomposition will be affected, and the efficiency will decrease obviously. If the number of templates is too small, it will inevitably affect the accuracy of subsequent event extraction, and even cannot distinguish adjacent events. As is shown in Fig. 5, $\beta = 0.6$ and 0 $\delta = 0.01$, and the model has the best template extraction ability at this time. We also take these two parameters as the values of our subsequent experiments. Of course, for different original logs, the optimal values of these two parameters will be different, and the actual optimal values need to be tested separately.

In the test of event extraction ability, we use 500000 consecutive original network and system logs to extract about 6000 templates in Fig. 6. Our main goal is to model log data and network events and extract meaningful events for network operations. The test index is the effectiveness and accuracy of event extraction. Because of the complexity of the original log, it is difficult to distinguish whether the extracted network events are accurate or meaningful. Based on this situation, we use a new index to represent the ability of the LTF model to express data and the effectiveness of event extraction. We calculate the proportion of the extracted event templates in the total templates. Because the LTF tensor decomposition model can adjust the time window to obtain different number of network events (the larger the event window is, the fewer events are extracted). Therefore, only controlling the size of the event time window to see the number of events cannot judge the event extraction ability of the LTF model. At the same time, from the theoretical

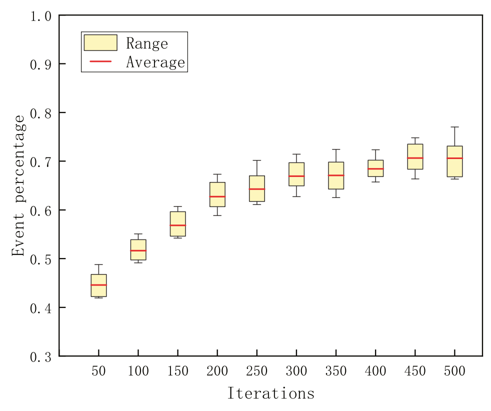


Fig. 6. Comparison of the proportion of event templates to total templates with different iterations. Higher proportion implies greater effectiveness and accuracy of event extraction.

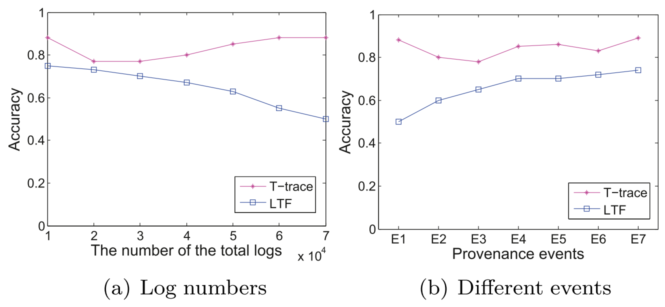


Fig. 7. Accuracy Comparison with Logs and Events.

analysis, if the extracted network events are 100% correct, then the template contained in these events must account for 100% of the total template. Thus, it is effective and convenient to use this index to test the event extraction ability of the LTF tensor decomposition model. It can be seen from Fig. 6 that the ability of event extraction increases with the number of iterations for different original network and system logs. Because the different initial sub-tensors will also affect the efficiency of tensor decomposition, so when the initial sub-tensors are generated by random numbers, the seed of random numbers must be kept unchanged in the subsequent parameter adjustment process to prevent different initial sub-tensors from affecting the results. As can be seen in Fig. 6, when the number of iterations is 100, the ideal result can be achieved. Although the results of 200 and 500 iterations are higher than those of 100 iterations, considering the overfitting factor and the increase of time operation cost, it is generally considered that 100 iterations are the ideal result, and the LTF tensor decomposition model also has a good event extraction ability.

We evaluate T-trace and LTF in different scales of log volume with different events in them. As the experiment shows in Fig. 7, the accuracy of the T-trace can consistently maintain a high level of accuracy as the total number of logs increase. In contrast, the accuracy of LTF tends to decrease as the logs increases. The accuracy of T-trace can be 38% higher than the LTF when the line number was about 7000 which is shown in Fig. 7(a). We also

TABLE VII
DESCRIPTION OF PROVENANCE EVENTS

No.	Event name	Attack stage	Logs number
E1	External Remote Services	Stage 1	5
E2	Powershell	Stage 3	10
E3	Modify Existing Service	Stage 6	35
E4	Account Discovery	Stage 5	52
E5	Spearphishing Link	Stage 2	67
E6	Stored Data Manipulation	Stage 8	90
E7	Credential Dumping	Stage 4	96

select 7 events that belong to different attack stages discussed in our running example to compare the results in identifying the events, and Table VII shows the details of events. Fig. 7(b) shows that the accuracy of LTF is positively correlated with the number of logs. The accuracy of LTF is low when the number of logs is small.

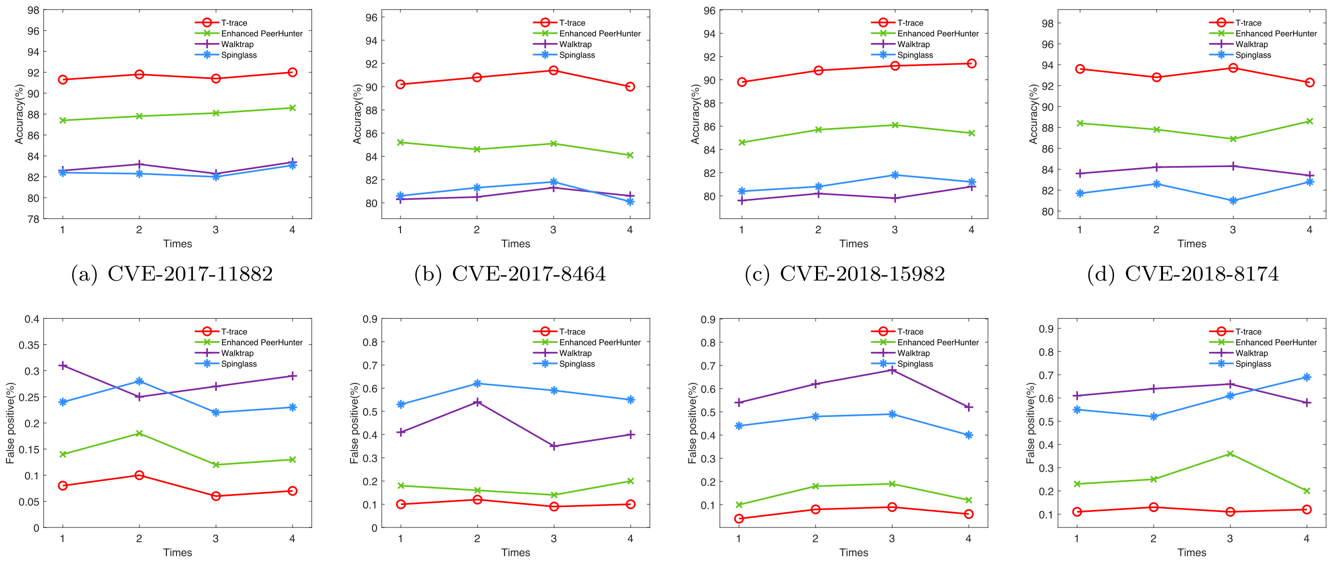
While the same number increases, the accuracy of the T-trace is relatively stable. The reason is that LTF is based on a recommendation algorithm, in which the more the target quantity, the easier it is to find the relationship. However, the number of logs composing events is not stable in practical systems, which leads to the fluctuating event extraction accuracy of LTF. Overall, T-trace reaches an accuracy of about 85% on average, which is 10-20% higher than the LTF.

C. Threshold Determination in Event Abstraction

To prevent redundant logs from interfering with the experiment, in the first step, we need to preprocess the logs. For redundant logs (logs with a high repetition rate), if there is no time relationship between the redundant logs and the others in the same event source, we consider that these logs are not necessary for analysis. Through experiments, we have accurately determined a threshold that can effectively filter logs.

We compare T-trace to the log tensor factorization (LTF) approach [34], which uses tensor decomposition obtained events. In our experiment, we use log collector tools such as Event Tracing for Windows (ETW), Procmon, Wireshark, and system Firewall to collect logs. The collected logs contain fields such as timestamp, process name, operation, operation path, result, details, etc. To prevent redundant logs from interfering with the experiment, in the first step, we need to preprocess the logs. For redundant logs (logs with a high repetition rate), if there is no time relationship between the redundant logs and the others in the same event source, we consider that these logs are not necessary for analysis. In the experiment, we set a threshold to filter out logs with excessive frequency. We define the frequency of the Process Message group appearing in the time windows as the threshold for filtering logs. Specifically, we chose 0.02 as the threshold in our experiments.

The event threshold T_e and relation threshold T_r mentioned in which Algorithm 1 can influence the result of event abstraction. We evaluate the accuracy of event identification using different thresholds on 70,000 log entries. We denote the accuracy of event abstraction by dividing the number of events correctly identified by the total number of events provided in the T-trace. The result shows that when the threshold is too low (<0.0084), T-trace is



(a) CVE-2017-11882 (b) CVE-2017-8464 (c) CVE-2018-15982 (d) CVE-2018-8174

(e) CVE-2017-11882 (f) CVE-2017-8464 (g) CVE-2018-15982 (h) CVE-2018-8174

Fig. 8. Accuracy and False Positive of Provenance Graph.

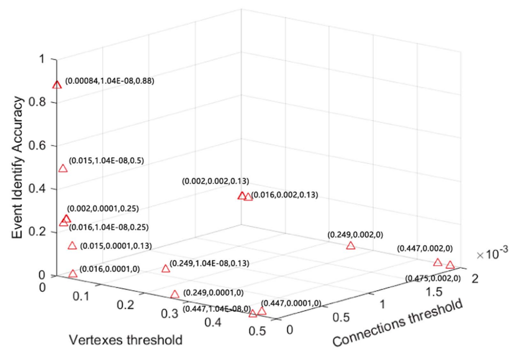


Fig. 9. Accuracy with Different Thresholds .

easy to include more redundant events so that the granularity of event extraction becomes smaller. However, when the vertex threshold is set too high, T-trace may inadvertently overlook crucial events, leading to a coarser granularity in event extraction. According to the results shown in Fig. 9, as the $vertex$ is 0.002 and $connection$ is 1.04×10^{-8}, the event identifying accuracy of T-trace can reach the highest rate of 88%.

D. Performance of Provenance Graph

Compared with other provenance graph construction methods, we have fully proved the effectiveness of T-trace attack provenance under the DARPA dataset and a variety of simulated attacks.

To construct a more complete and accurate attack chain, we need community detection with high accuracy and a low false-positive rate. We define the accuracy AC and false-positive rate FP to measure the optimization of our community discovery results.

TABLE VIII
Experimental Results of the DARPA Datasets

Experiment	Precision	Recall	Accuracy	F1-score
DARPA FiveDirections	0.99	1.0	0.99	0.99
DARPA ClearScope	0.98	1.0	0.98	0.99
DARPA THEIA	1.0	1.0	1.0	1.0

In (25), we denote the number of true positives, false positives, true negatives, and all events as tp, fp, tn, E_{all}, respectively

$$AC = \frac{tp + tn}{E_{all}}, \quad FP = \frac{fp}{fp + tn} \tag{25}$$

To avoid the "accuracy" paradox, we also use the F1-score to measure system performance, which is the harmonic mean of precision and recall

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recal}} \tag{26}$$

In order to demonstrate our detection performance, we adopted the data set released by DARPA. The experiment simulates a company's key security services, such as Web server, SSH server, email server, and SMB server. The red team performs APT attacks by using, for example, Firefox backdoor, Nginx backdoor, and phishing emails. A more detailed description of the attack is available online.

During the engagement, the red team used different attack methods to launch APT attacks, and these attacks accounted for less than 0.001% of the audit data [27]. As shown in Table VIII, despite the large number of benign events in the attack process, T-trace still has a high-efficiency detection performance.

In the simulation experiment, we launch the four APT attacks to evaluate the performance of our approach. As is shown in Fig. 8, we compare the accuracy and false-positive rates of T-trace, $Enhanced\ PeerHunter$ [37], $Walktrap$ [38] and $Spin\text{-}glass$

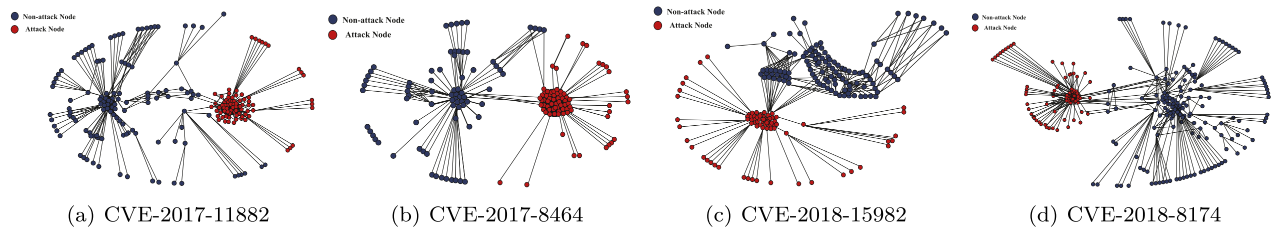


Fig. 10. Community Division of Attacks.

[39] in the detection of multiple attacks. Because T-trace has optimized the badly connected communities in detection, our approach can greatly reduce the incorrect connections between attack-related events and attack-unrelated events. Better events division effectively improves the accuracy of the detection system and reduces the false-positive. The accuracy of our provenance graph reaches 92%, and the false-positive is as low as 0.09%. Compared with the Louvain algorithm without refining partitions, the accuracy of our approach is improved by 5.5% as shown in Fig. 8(a)–(d), and the false-positive rate is reduced by 50% as shown in Fig. 8(e)–(h).

E. Performance of Attack Chain Extraction

Based on the simulated attack, we compared the HERCULE and POIROT schemes, and constructed the attack chain. In the performance comparison, T-trace has better performance in terms of accuracy rate, false positive rate, F1-score and time efficiency.

In Fig. 10, T-trace provides visual outputs of the community division for the attacks. The red community is the attack-related community, and the others are the attack-unrelated communities. Based on the attack-related community, the attack chain can be constructed. Next, we show the attack chain of CVE-2017-11882, which is a Microsoft Office memory corruption vulnerability. An attacker can use this vulnerability to create malicious Microsoft Word documents and launch an attack with it. T-trace can find out the following threats steps with the attack chain extraction.

In the first step, the attacker crafts a simple message in the phishing email urging the targeted recipient to download a file from Gmail. In the second step, once the host clicks and edits the downloaded Word document, it will cause the stack overflow of the EQNEDT32 module. Then it makes a reverse TCP connection to the C&C server's IP address 192.168.1.5. In the third step, the attacker obtains the reverse shell on the victim's system and then browses different folders and files. After the attacker finds the target file, he initiates the FTP client in the command line to upload the target file to the IP address 192.168.1.5.

In attack detection, we compared HERCULE [40] and POIROT [41]. Compared with Hercule, we optimized the community discovery algorithm and refined the poorly connected communities generated during the community discovery process. And the POIROT develops efficient approximate graph matching algorithms to match query graphs against the data from audit logs, but Poirot's query graph requires non-trivial efforts of cyber analysts to manually construct it. Table IX shows that we extract the latent correlations of the log entries generated from the attack flow via T-trace with the average accuracy of 91.4% and the average false positive rate of 0.05%. Compared with HERCULE and POIROT, the average detection accuracy increases by 4.3% and 1.6%, and the average false positive reduces by 0.08% and 0.01%. It shows that the F1-score of T-trace is closer to 1, with better performance. Also, T-trace has higher time efficiency, and the detection time is 73.6% lower than HERCULE and 57.2% lower than POIROT. T-trace can accurately and rapidly constructs a provenance graph, which can further extract a more accurate and real attack chain. The results show that our approach can be effective in practical applications.

TABLE IX
EXPERIMENTAL RESULTS OF SIMULATED ATTACK

Scheme	Accuracy	F1-score	False Positive	Time(s)
T-trace	91.4%	0.88	0.09%	20.3
HERCULE	87%	0.72	0.17%	87.1
POIROT	89.7%	0.80	0.10%	52.1

F. Runtime Performance of T-Trace

To better reflect the time cost of T-trace under the huge amount of data, we choose the attack logs in CSE-CIC-IDS2018. With the attack logs, we compare the time costs of T-trace with `Enhanced PeerHunter` [37]. `Enhanced PeerHunter` is a P2P network flow detection component. It uses mutual contacts to cluster bots into communities. Then, it uses network-flow-level community behavior analysis to detect potential botnets. By comparison, we confirm that T-trace has better time efficiency.

In the experiment, we find that the detection algorithm consumes a lot of time in community detection. In the community detection phase, we set pruning judgments to reduce the local movement of nodes, and thus get better experimental results. As described in our theory, the experimental results in Fig. 11 prove that T-trace optimizes the local movement by reducing the movement judgment, which can effectively improve the calculation efficiency. T-trace can effectively improve computational efficiency and reduce the time by up to 90%. The more iterations the algorithm processes, the more time optimization T-trace can gain.

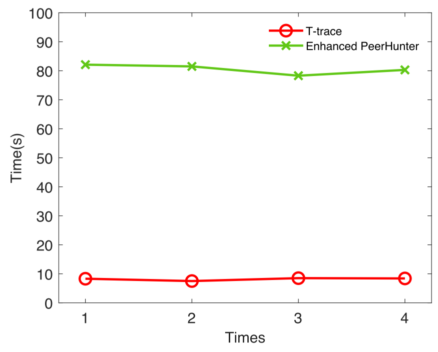


Fig. 11. Time Efficiency Comparison.

VII. Conclusion and Future Work

In this paper, our goal is to identify ongoing APTs campaigns, disclose disparate steps over a long time and provide a high-level visual graph of the attack scenario to an analyst, based on audit logs and network flows from the enterprise. Our approach transfers the low-level logs to structured events, which mitigates the semantic gaps. Moreover, we construct the provenance graph and prune away unrelated connections to the attack campaign, which can solve the dependency explosion problem. We evaluate the efficiency and accuracy of our approach with four launched practical APTs attacks and open-source data sets. The results show that T-trace improves accuracy rate by 38% when compared to prior work in identifying the concrete events, and it efficiently reduces time cost by 90% and achieves a 92% accuracy rate in generating concise dependency graphs for the alert events.

For future work, we will test our approach in more platforms and systems and will give further attack prevention instructions according to our provenance results. Moreover, the experiments have shown some promising results in identifying the network events from raw logs, and we plan to investigate this further. We will also run more experiments with more diverse datasets.

References

[1] K. Doshi, Y. Yilmaz, and S. Uludag, "Timely detection and mitigation of stealthy DDoS attacks via IoT networks," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2164–2176, Sept./Oct. 2021.

[2] M. N. Hossain et al., "Dependence-preserving data compaction for scalable forensic analysis," in *Proc. USENIX Secur. Symp.*, 2018, pp. 1723–1740.

[3] C. Tan, Q. Wang, L. Wang, and L. Zhao, "Attack provenance tracing in cyberspace: Solutions, challenges and future directions," *IEEE Netw.*, vol. 33, no. 2, pp. 174–180, Mar./Apr. 2018.

[4] W. U. Hassan, M. A. Noureddine, P. Datta, and A. Bates, "OmegaLog: High-fidelity attack investigation via transparent multi-layer log analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–16.

[5] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *Proc. Symp. Secur. Privacy*, 2020, pp. 1172–1189.

[6] W. U. Hassan, L. Aguse, N. Aguse, A. Bates, and T. Moyer, "Towards scalable cluster auditing through grammatical inference over provenance graphs," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[7] T. Pasquier et al., "Runtime analysis of whole-system provenance," in *Proc. Comput. Commun. Secur. Conf.*, 2018, pp. 1601–1616.

[8] T. Li et al., "AClog: Attack chain construction based on log correlation," in *Proc. Glob. Commun. Conf. Conf.*, 2019, pp. 1–6.

[9] H. Studiawan, F. Sohel, and C. Payne, "Anomaly detection in operating system logs with deep learning-based sentiment analysis," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2136–2148, Sep./Oct. 2021.

[10] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. Comput. Commun. Secur. Conf.*, 2017, pp. 1285–1298.

[11] Y. Shen, E. Mariconti, P. A. Vervier, and G. Stringhini, "Tiresias: Predicting security events through deep learning," in *Proc. Comput. Commun. Secur. Conf.*, 2018, pp. 592–605.

[12] M. Barre, A. Gehani, and V. Yegneswaran, "Mining data provenance to detect advanced persistent threats," in *Proc. 11th Int. Workshop Theory Pract. Provenance*, 2019, pp. 1–11.

[13] Fireeye, "M-trends 2020: Insights into today's cyber attacks," 2020. [Online]. Available: https://content.fireeye.com/m-trends

[14] M. N. Hossain, S. Sheikhi, and R. Sekar, "Combating dependence explosion in forensic analysis using alternative tag propagation semantics," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 1139–1155.

[15] W. U. Hassan et al., "Nodoze: Combatting threat alert fatigue with automated provenance triage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.

[16] Y. Liu et al., "Towards a timely causality analysis for enterprise security," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[17] A. Tabiban, Y. Jarraya, M. Zhang, M. Pourzandi, L. Wang, and M. Debbabi, "Catching falling dominoes: Cloud management-level provenance analysis with application to openstack," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2020, pp. 1–9.

[18] J. Zeng et al., "WATSON: Abstracting behaviors from audit logs via aggregation of contextual semantics," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–18.

[19] B. Li et al., "JSgraph: Enabling reconstruction of web attacks via efficient tracking of live in-browser Javascript executions," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[20] S. Ma, J. Zhai, F. Wang, K. H. Lee, X. Zhang, and D. Xu, "MPI: Multiple perspective attack investigation with semantic aware execution partitioning," in *Proc. USENIX Secur. Symp.*, 2017, pp. 1111–1128.

[21] C. Xiong et al., "Conan: A practical real-time APT detection system with high accuracy and efficiency," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 551–565, Jan./Feb. 2022.

[22] M. N. Hossain et al., "SLEUTH: Real-time attack scenario reconstruction from COTS audit data," in *Proc. USENIX Secur. Symp.*, 2017, pp. 487–504.

[23] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," 2020, *arXiv: 2001.01525*.

[24] G. Berrada et al., "A baseline for unsupervised advanced persistent threat detection in system-level provenance," *Future Gener. Comput. Syst.*, vol. 108, pp. 401–413, 2020.

[25] S. Benabderrahmane, G. Berrada, J. Cheney, and P. Valtchev, "A rule mining-based advanced persistent threats detection system," 2021, *arXiv:2105.10053*.

[26] R. Yang, S. Ma, H. Xu, X. Zhang, and Y. Chen, "UIScope: Accurate, instrumentation-free, and visible attack investigation for GUI applications," *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020.

[27] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "Holmes: Real-time apt detection through correlation of suspicious information flows," in *Proc. Symp. Secur. Privacy*, 2019, pp. 1137–1152.

[28] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proc. Comput. Commun. Secur. Conf.*, 2019, pp. 1777–1794.

[29] B. Yiğit, G. Gür, F. Alagöz, and B. Tellenbach, "Cost-aware securing of IoT systems using attack graphs," *Ad Hoc Netw.*, vol. 86, pp. 23–35, 2019.

[30] M. Inokuchi et al., "Design procedure of knowledge base for practical attack graph generation," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2019, pp. 594–601.

[31] X. Sun, J. Dai, P. Liu, A. Singhal, and J. Yen, "Using Bayesian networks for probabilistic identification of zero-day attack paths," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 10, pp. 2506–2521, Oct. 2018.

[32] "APTnotes," 2019. [Online]. Available: https://github.com/kbandla/APTnotes

[33] A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*, Hoboken, NJ, USA: Wiley, 2009.

[34] T. Kimura et al., "Spatio-temporal factorization of log data for understanding network events," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 610–618.

[35] A. D. Keromytis, "Transparent computing engagement 5 data release," 2020. [Online]. Available: https://github.com/darpa-i2o/Transparent-Computing
[36] "CSE-CIC-IDS2018," 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html
[37] D. Zhuang and J. M. Chang, "Enhanced peerhunter: Detecting peer-to-peer botnets through network-flow level community behavior analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 6, pp. 1485–1500, Jun. 2019.
[38] F. Hu, Y. Zhu, Y. Shi, J. Cai, L. Chen, and S. Shen, "An algorithm walktrap-SPM for detecting overlapping community structure," *Int. J. Modern Phys. B*, vol. 31, no. 15, 2017, Art. no. 1750121.
[39] S. Rahiminejad, M. R. Maurya, and S. Subramaniam, "Topological and functional comparison of community detection algorithms in biological networks," *BMC Bioinf.*, vol. 20, no. 1, 2019, Art. no. 212.
[40] K. Pei et al., "HERCULE: Attack story reconstruction via community discovery on correlated log graph," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, 2016, pp. 583–595.
[41] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. Venkatakrishnan, "POIROT: Aligning attack behavior with kernel audit records for cyber threat hunting," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1795–1812.

Teng Li (Member, IEEE) received the BS degree from the school of computer science and technology, Xidian University, China, in 2013, and the PhD degree from the school of computer science and technology, Xidian University, China, in 2018. He is currently an associate professor with the school of cyber engineering, Xidian University, China. His current research interests include wireless and mobile networks, distributed systems and intelligent terminals with focus on security and privacy issues.

Ximeng Liu (Senior Member, IEEE) received the BSc degree in electronic engineering from Xidian University, Xi'an, China, in 2010, and the PhD degree in cryptography from Xidian University, China, in 2015. Now, he is a full professor with the College of Mathematics and Computer Science, Fuzhou University, China. Also, he is a research fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 100 research articles including *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Computers*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Services Computing* and *IEEE Transactions on Cloud Computing*. He awards "Minjiang Scholars" distinguished professor, "Qishan Scholars" in Fuzhou University, and ACM SIGSAC China Rising Star Award (2018). His research interests include cloud security, applied cryptography and Big Data security. He served as a leader guest editor for *Wireless Communications and Mobile Computing* and a senior member of the ACM, CCF.

Wei Qiao received the BS degree from Xidian University, Xi'an, China, in 2021. He is currently working toward the master's degree with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His research interests include attack detection, network security, cryptography and information security.

Xiongjie Zhu received the BS degree in software engineering from the Xi'an University of Technology, in 2022. He is currently working toward the master's degree in cyber security with the School of Network Engineering, Xidian University. His research interests cover intrusion detection and prevention and apt attack research in cyber security.

Yulong Shen (Member, IEEE) received the BS and MS degrees in computer science and the PhD degree in cryptography from Xidian University, Xi'an, China, in 2002, 2005, and 2008, respectively. He is currently a professor with the School of Computer Science and Technology, Xidian University, where he is also the associate director of the Shaanxi Key Laboratory of Network and System Security and a member of the State Key Laboratory of Integrated Services Networks. His research interests include wireless network security and cloud computing security. He has also served on the technical program committees of several international conferences, including ICEBE, INCoS, CIS, and SOWN.

Jianfeng Ma (Member, IEEE) received the BS degree in computer science from Shaanxi Normal University, in 1982, and the MS degree in computer science from Xidian University, in 1992, and the PhD degree in computer science from Xidian University, in 1995. Currently, he is the directer of Department of Cyber engineering and a professor with the School of Cyber Engineering, Xidian University. He has published more than 150 journal and conference papers. His research interests include information security, cryptography, and network security.