Synthetic Audit Log Generation for Embedded APT Campaign Detection

Abstract

With the increasing sophistication of Advanced Persistent Threats (APTs), the need for robust detection and mitigation methods has become more pressing than ever. This paper addresses the challenges associated with detecting APT attacks from system audit logs, exacerbated by the absence of a comprehensive benchmark dataset. In this paper, we propose a configurable synthetic audit log generation facilitated by adopting a red-team emulator for representative pattern generation. We propose a two-way approach that involves anomaly detection and attack pattern detection tailored for highly imbalanced abnormal events to detect and label attack patterns with MITRE ATT&CK Techniques. Our methodology extends to the attribution of APT campaigns by matching identified techniques with known ATP campaigns. In an extensive evaluation involving eight real-world APT campaigns and 20 synthesized attacks, our learning-based approach surpasses competing methods, providing not only enhanced attack behavior detection capabilities but also a ranked list of potential known campaigns, offering valuable support for attack investigations.

1 Introduction

The emergence of sophisticated threats such as Advanced Persistent Threats (APTs) brings greater challenges to the cybersecurity community, such as BlackEnergy [22] and SolarWinds Compromise [24]. APT cyberattacks are believed to be plotted by a group of advanced threat actors with economic or political motivations. Unlike malware or botnet attacks, these APT activities consist of multiple stages, including gaining a foothold in a victim environment, remaining undetected over a long period of time, exfiltrating sensitive data, and compromising the integrity of systems. Although malware analysis has made some progress [32, 64], effective detection and mitigation of APTs have become urgent requirements for system protection. Amid the difficulties in APT threat studies, a significant obstacle has been the absence of a benchmark dataset to evaluate their solutions, which motivates the study of synthetic audit log generation in this work.

Recent work [8, 26, 28, 47, 61, 66, 69] has shown that data provenance methods facilitate threat detection performance, along with a decrease in false alarms, and reduce the workload of security experts. The data provenance transforms the system logs into a graph representation, which effectively depicts the causal and temporal relations between the system entities. Provenance-based intrusion detection systems, such as anomaly-based [28, 29] and tag propagation-based systems [30, 31, 47], analyze causal relations and correlate data flow. Given the potentially vast size of the provenance graph, these systems have proven effective in examining individual techniques and multistage attacks. In this work, the concept of data provenance is employed for data size reduction as well as causality inference.

Recent graph learning-based systems [26, 27, 66] have integrated information on nodes and edges with embedding techniques for attack investigation. Leveraging the rich structural properties of the data provenance enables accurate recognition of abnormal events and helps reduce false positives. Provenance analysis includes both coarse-grained analysis, which categorizes events as benign or malicious, and fine-grained analysis that maps events to TTP (Tactic, Technique, and Procedure), as seen in systems like Holmes [47] and RepSheet [28]. However, current fine-grained approaches require manual participation to specify mapping rules. In addition, when an attack is detected, human experts must sift through a substantial volume of event logs to trace back the initial intrusion point, prioritize alerts, and potentially attribute the threat to an organization. Forensic analysis of a security incident remains a labor-intensive task. To automate the analysis, there is a need for a large labeled dataset that could serve as a benchmark for evaluating attack pattern detection and APT campaign discovery, and could be used as training data for machine learning solutions.

To address the challenge, this study aims to develop a generative threat model designed to generate APT attack scenarios embedded in audit logs, adhering to the structure and vocab-

ularies of MITRE ATT&CK. Using the synthesized audit log, we construct a machine learning-based methodology to discover malicious behaviors and ultimately identify potential APT threats. Specifically, the tasks of this study are: 1) to obtain attack scenarios using a red team emulator and accurately label the collected logs with TTP identifiers based on the MITRE ATT&CK framework, 2) to synthesize audit logs containing APT attack scenarios and benign behaviors to closely resemble real data logs, 3) to design a neural network detection model to discover malicious behaviors, and 4) to identify APT attacks by matching the discovered behaviors with known APT campaigns.

2 Background and Motivation

In this section, we first introduce provenance analysis and its challenges with a running example. We then analyze the problem of APT attack investigation from our perspectives.

2.1 Running Example

We refer to APT28 threat intelligence reports [10] to construct its attack scenario, illustrated in Figure 1 ⓑ as the running example in this work. In the initial stages of the attack, phishing emails containing malicious attachments are sent to the victim to download the attachment IOC_09_11.rar that the activity is designated as the technique *T1566.001 Spearphishing Attachment* of the attack lifecycle <Initial Compromise>. This malicious attachment exploits a vulnerability in WinRAR versions below 6.23 (as mentioned in CVE-2023-38831) for the attack. The RAR compressed file contains a normal PDF file, IOC_09_11.pdf, and a folder with the same name that contains a malicious batch script.

When the victim clicks on the normal PDF file, it triggers the batch script in the folder, IOC_09_11.pdf .cmd, initiating malicious activities designated as *T1204.002 User Execution: Malicious File* of the attack lifecycle <Establish Foothold>. The activities are listed with steps ③ to ⑧ of Figure 1 ⓑ. APT28 attack continues to establish a reverse shell between the attacker and the victim, designated as *T1071 Application Layer Protocol* of attack lifecycle <Establish Foothold>,(steps ⑨ to ⑰), and steals information such as local state files, designated as *T1082 System Information Discovery* and *T1005 Data from Local System* of <Internal Reconnaissance> with steps ⑱ to ⑳. Then uploads the stolen information to a legitimate web service - webhook.site, designated as *T1567 Exfiltration Over Web Service* of <Complete Mission>, with steps ⑱, ㉑ to ㉓.

2.2 Data Provenance

In a provenance graph, the nodes typically represent data entities or processes, and the edges represent the workflow or relationship among the nodes. Data provenance leverages the informative representation of a provenance graph to track the origin of an event and explain the reasons of the evolution to its present state. In alignment with previous research [8,28,46, 67,69], we adopt the data provenance approach to model audit logs. A provenance graph of the audit log can be represented as a directed acyclic graph $G = (V, E)$ that the node in V are system entities such as process, file, and network, and an edge in E represents an event between two system entities such as *Process Create* , *WriteFile* and *TCP Connect*.

The data model of the provenance graph is shown in Figure 9 of Appendix.A.2. The system entities are determined by the underlying operating system. Here, we focus on the Windows platform and consider that four entity systems consist of *process, file, network, and registry*, and 27 system operations. A provenance graph of the audit log connects the entities with their event relations, as well as the timestamp of event occurrence in this study.

2.3 APT Attack Lifecycle

The frameworks, such as Lockheed Martin's cyber kill chain [39], MITRE ATT&CK framework [56], and Mandiant's adversary life cycle [43], help defenders understand the intentions and operations of APT attacks and how collective operations of APT attacks achieve their objectives. In this study, we use Mandiant's adversary life cycle, as illustrated in Figure 1 ⓐ, as a reference for the APT attack lifecycle in stages/tactics, although other frameworks can be equally applicable. Each tactic encompasses associated techniques, and each technique may have many implementations/abilities. Additionally, we employ the lifecycle framework to streamline the generation and analysis of synthetic audit logs.

A typical APT attack consists of multiple stages, including initial compromise, establish foothold, internal reconnaissance, escalate privileges, move laterally, maintain presence, and ultimately complete mission. Note that the stages of internal reconnaissance, escalating privileges, moving laterally, and maintaining presence can occur multiple times, and not all four stages must be present in every instance. Figure 1 depicts the lifecycle stage that APT28 techniques belong to.

2.4 Challenges

Even though a provenance graph offers a way to understand causal connections, it still needs to spend non-trivial efforts to recognize malicious behavior from normal activities without delay. Analysts need to navigate through a large volume of audit log events, apply their knowledge to formulate hypotheses, repeatedly search and evaluate for signs of threats and vulnerabilities, and finally map audit data into identifiable behaviors. Instead of characterizing and seeking artifacts, a more resilient strategy for recognizing cyberattacks is to

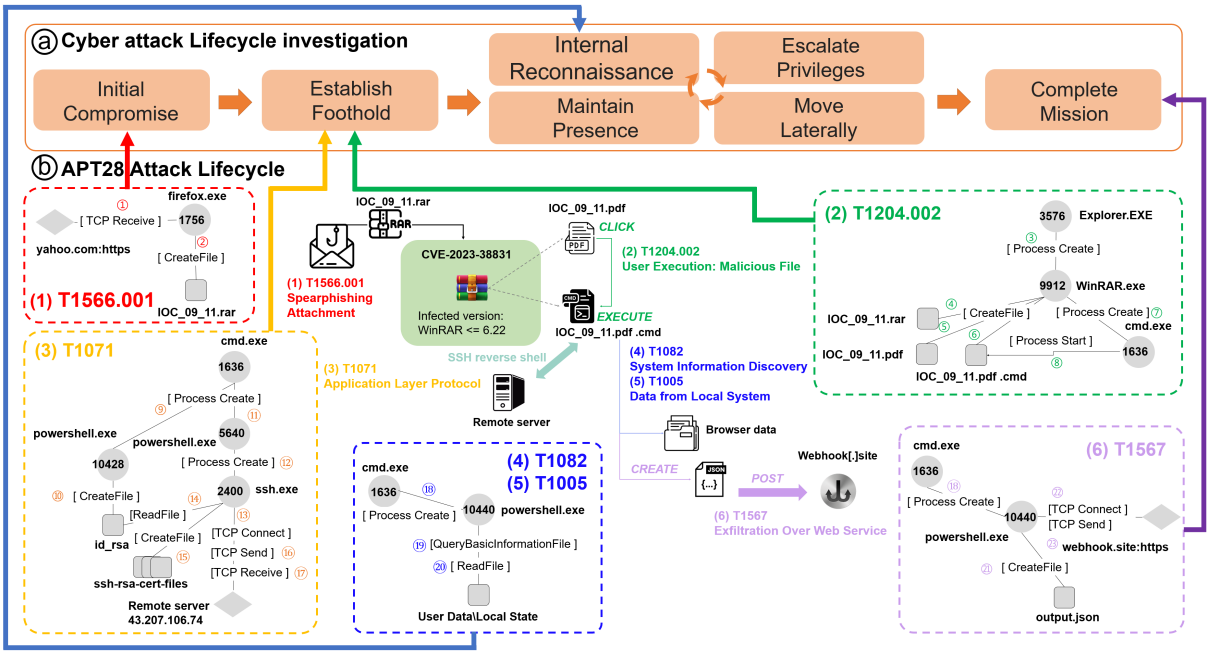

Figure 1: APT28 Attack: Lifecycle, MITRE ATT&CK techniques and events

capture the techniques adversaries employ to achieve their objectives, which is called TTP-based hunting [20, 32].

TTP-based hunting serves as a valuable complement to existing methodologies, such as the use of indicators of compromise (IOCs) [46] and statistical analysis of data for anomaly detection [28, 47]. However, the automated recognition of TTPs from audit events poses two significant challenges:

- **Manual TTP specifiation rule**. Holmes [47], RepSheet [28], and the Sigma rules search engine [5] are relevant to our study. Holmes first discovers TTPs and then infers potential attacks; RepSheet reconstructs tactical provenance graphs to depict an attack scenario; Sigma is an open signature format designed for describing log events, allowing researchers and analysts to articulate and share their detection approaches. However, these approaches that rely on human-created TTP specifications have two limitations.

  First, TTP rule specifications often do not fully correspond to the complete definition of a technique. Using Sigma as an example, it allows analysts to contribute their insights to the development of rule specifications, which may, in turn, involve a trade-off between precision and recall, leading to the creation of some inperfect rules. Another limitation is that rule development is based on known incidents, making the rules sometimes ineffective in detecting APTs or zero-day attacks.

- **APT campaign detection**. Several studies have investigated APT organization attribution using various data sources, such as malware analysis [53], cyber threat intelligence (CTI) analysis [51], and threat alert correlation [52]. Among many studies, APTer is the most relevant study for our work, but APTer attributes a campaign to a threat actor based on the correlation of alarms generated by threat alerts (such as IDS/IPS and SIEM systems) rather than low-level audit events. The 'low-and-slow' attack strategy employed by APTs introduces two difficulties in detecting APT campaign activities embedded in audit logs, namely, high false negative rates and extended dwell time. According to the FireEye report [23], the global median dwell time in 2018 was 78 days. This implies that attackers remain undetected on a network or system for more than two months, resulting in sparse observable malicious behaviors for attack correlation.

- **Lack of benchmark dataset**. With the rapid advancement of artificial intelligence, data-driven methods, such as machine learning and deep learning, are increasingly effective in addressing certain cyber threats, such as attack analysis [8, 21, 55, 67], and offer the potential to develop learning-based TTP detection methods in a timely manner. However, as far as we know, currently available audit log benchmark datasets for host-based attack detection lack detailed labels and diverse campaigns. Researchers typically rely on either self-created datasets or datasets from the Defense Advanced Research Projects Agency (DARPA), such as Transparent Computing (TC) engagement [19] and Operationally Transparent Cyber (OpTC) [18]. In addition, self-created datasets are rarely

released due to proprietary property and sensitivity concerns.

Some self-created datasets, such as ADFA-WD [16] and StreamSpot [44], are publicly available. ADFA-WD consists of Dynamic Link Library (DLL) traces recorded by Procmon on a Windows XP host, while StreamSpot describes one attack and five benign scenarios compiled into three datasets. Unfortunately, these datasets may not precisely reflect the ratio between benign and malicious events in real-world environments and could be limited in adaptability to other methods due to the limited number of events provided. Due to their limited sizes and diversity, these datasets are beneficial for studies that perform well in a controlled lab setting, but may not generalize well to real-world environments. Moreover, these public datasets lack TTP labels, which may not support in-depth analyses.

3 Problem and Approach

3.1 Approaches

To address these challenges, our approach involves detecting malicious behaviors (TTPs) from audit logs and subsequently determining the presence of an APT attack by aligning the detected behaviors with known campaigns. Additionally, recognizing the necessity for TTP-labeled APT campaign audit log datasets, we aim to generate campaign datasets by generalizing and integrating simulated campaigns into benign datasets and labeling audit log events with TTP labels. Moreover, our ultimate goal is to achieve APT campaign detection by using a learning-based TTP classification mechanism and aligning discovered TTPs with known APT campaigns.

APT campaigns generally adhere to the cyber attack lifecycle, as illustrated in Figure 1 for the APT28 campaign. To be more specific, the detection of APT campaigns involves identifying malicious behaviors corresponding to the stages of the APT lifecycle. This TTP identification, along with the interconnections between these behaviors, is crucial for effective detection of APT campaigns. In this context, audit logs that contain embedded APT campaigns become essential for the applications of machine learning methods. We employ a red team platform to carry out APT campaign simulations and capture the associated audit logs. This approach ensures that malicious behaviors are accurately labeled with MITRE ATT&CK TTP identifiers.

With a sufficient number of audit logs with embedded APT campaigns, we proceed to learn the representations of labeled malicious behaviors using a deep learning approach. We use the trained model to evaluate a tested audit log to identify suspected malicious behaviors and generate a directed graph, called the TTP graph, to represent the discovered malicious activities. In the TTP graph, the nodes correspond to the labeled TTPs, whereas the edges signify causal or temporal relationships among the nodes. The advantage of the TTP graph lies in its size reduction, facilitating long-term tracking of audit logs. Additionally, we model known APT campaigns as query graphs, aligning the TTP graph with the highest likelihood as threat actors.

3.2 Framework

Figure 2 shows the proposed framework, which consists of a generative threat model (SAGA, Synthetic APT Campaign Generation) and an APT threat detection model. The generative threat model is crafted to generate APT campaigns following the specified cyber attack lifecycle. The threat detection model is dedicated to identifying attack behaviors and attributing them to an attack group.

In the generative threat model, we start the simulation of a multi-stage APT campaign using a red-team emulator, collecting associated audit logs labeled with attack patterns (such as CALDERA Ability) for a given attack a. Next, we generalize each attack pattern into an attack pattern template. Each template t specifies a configuration for its prerequisites and the rule of replacement of the system entities. These configurations assist in the generation of diverse variations of attack patterns that satisfy specific prerequisites. When provided with a set of attack pattern templates T and background events B, the proposed threat generation model SAGA involves a multi-stage APT campaign generation. For more details, see Section 4. The generation process results in synthesized audit events E used as training data to train the threat detection model and testing data for APT attack detection methods.

In the threat detection model, audit events E are first collected and transformed into provenance graphs G for analysis. We consider the prolonged duration of multi-stage attacks, which often span several months; thus, we conduct graph reduction to facilitate the analysis of a vast amount of data provenance. Next, learning graph embeddings from the causally connected provenance graph grasp node and edge representations to identify attack behavior a. We model these discovered behaviors as a query graph G_q, called TTP graph, and find an alignment $(G_q :: G_c)$ corresponding to known campaigns G_c. The alignment process determines an alert if the likelihood exceeds a threshold, indicating the presence of an attack campaign. The details are described in Section 5.

4 Generative Threat Model

4.1 Attack Pattern Templates

To simulate real-world adversaries, we consider CALDERA [1, 9], as the red team platform to emulate real adversaries to achieve complicated attacks, due to its openness and alignment with the TTPs of MITRE ATT&CK. To deploy CALDERA, we set up a master server and Remote Access Tool (RAT) clients on a compromised host. An attack

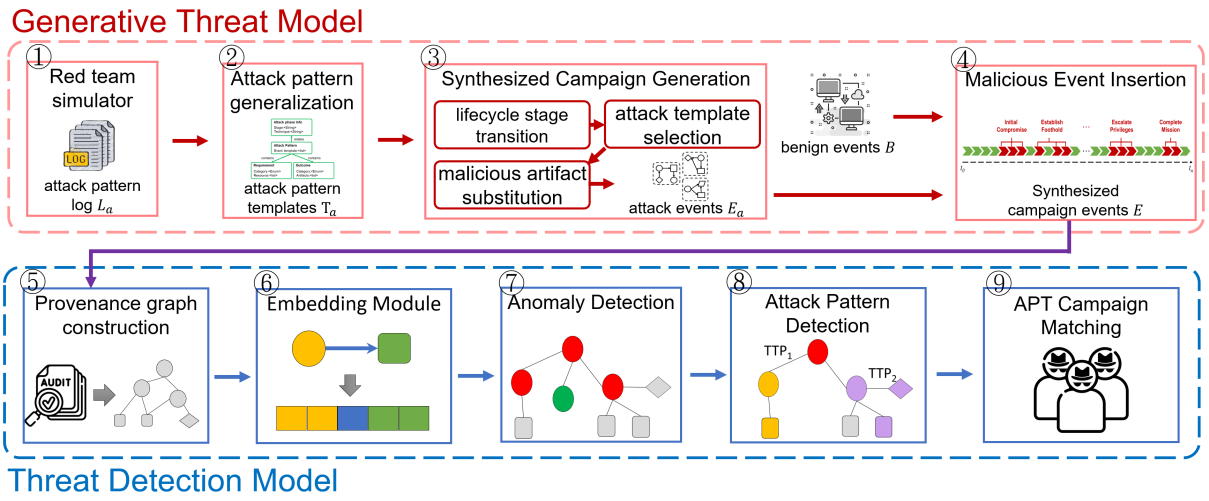

Figure 2: SAGA and APT campaign detection

pattern (called as *ability*) of CALDERA represents a distinct implementation of ATT&CK technique. We execute these abilities in CALDERA, meticulously record all details of relevant audit events using procmon [57] on the victim machine, and label the associated events.

Given that an attack pattern is just one instance of an attack behavior with specific argument values, it becomes imperative to generalize these patterns into a higher-level representation, such as a context-free grammar. This enables the generation of a broader spectrum of attack instances. We abstract each attack pattern into an attack template T, detailing its prerequisites, outcomes, and the corresponding stage and technique in the attack lifecycle, described in Figure 1 ⓐ. The meta-information guides the generation of the generative attack instances, ensuring causal correlation among the generated instances. Figure 3 describes the attack template model, complemented by an example in Figure 4.

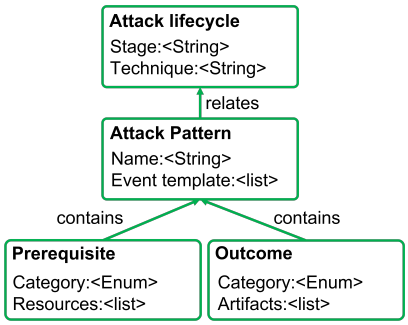

Figure 3: Attack pattern template model

- **Attack lifecycle** shows the attack *stage* and *technique* that this attack pattern belongs. For example, an attack pattern may associate with Technique *T1566.001 Spearphishing Attachment* and the stage *Initial Compromise* in the adversary lifecycle (as lines 2-3 in Figure 4).
- **Attack pattern** comprises an attack pattern name and its associated audit events as event templates. The name refers to an ability, such as *1afaec09315ab71fdfb167175e8a019a*, to implement a certain technique *T1566.001*. Event templates contain specific audit event records generated when an ability is executed. We generalize the system entities of audit events so as to generate more attack instances in synthetic audit log generation. Following the same example, Three system entities are involved in phisihing: a *chrome* connected to a *Microsoft Exchange* and downloading *job.docx* through the browser. These entities can be replaced with other terms in the same category or hypernyms (lines 11-15 and lines 29-32 in Figure 4), such as *Process.browser* for *chrome* and *File.phishing* for *job.docx*.
- **Prerequisite** specifies conditions on the system resources of the attack pattern that should be satisfied, such as *process.browser* and *file.phishing* (lines 4-8 in Figure 4) of the attack pattern template of APT28. When generating an APT attack, a campaign information table holds a list of previously executed attacks and their used system entities to avoid confusion. An attack instance is generated only if all prerequisites are satisfied; more details are given in Section 4.3.2.
- **Outcome** addresses the artifacts generated after executing this attack, such as a phishing file (line 9 in Figure 4), collected sensitive information, and system state change.

4.2 Synthesized Campaign Generation: SAGA

When provided with a set of attack templates T and a sequence of background audit events B, the process of generating a

```
"Attack Template Information":{
    "stage": "Initial Compromise",
    "technique" : "T1566.001" ,
    "prerequisite" : [
        "process.browser"
        "network.host machine",
        "network.mail_server",
        "file.phishing"],
    "outcome" :["file.phishing"],
    "events" : [
        {"srcNode" : {
            "Name" :"process_browser['Name']",
            ...
            "pid" : "process_browser['pid']",
            "Type" : "Process"},
        "dstNode" : {
            ...
            "Dstaddress" : "network_mail_server['Ip']",
            "Port": "network_mail_server['port']",
            "Type" : "Network"},
        "relation": "TCP Connect",
        "timestamp":20220415213135},
        {...},
        {"srcNode" : {
            "Name" :"process_browser['Name']",
            ...
            "pid" : "process_browser['pid']",
            "Type" : "Process"},
        "dstNode" : {
            ...
            "Name": "file_phishing['Name']",
            "Type": "File"},
        "relation": "Createfile",
        "timestamp":20220415213136}]}
```

Figure 4: APT28 Attack pattern template description

campaign as synthesized audit events E involves producing a sequence of attack events as described in Section 4.3 and inserting these malicious events into the background events described in Section 4.4.

Generating a sequence of attack instances conditioned on the adversary lifecycle involves *lifecycle stage transition*, *attack template selection*, and *artifact substitution*. The *lifecycle stage transition* process determines the next stage conditioned on the current stage and the adversary lifecycle shown in Figure 1ⓐ. The *attack pattern template selection* process decides an attack pattern depending on the current stage and a given sequence of the compromised attack patterns. The *artifact substitution* process creates the audit events of an attack instance based on the selected attack template. To generate a sequence of malicious events, a random number is generated to decide the number of stages of the attack lifecycle. For each stage, an attack template is selected based on the requirements of the given stages and compromised attacks. Next, the system entities of the template are replaced with malicious artifacts to generate diverse audit events associated with a TTP identifier. The audit events generated are malicious attack behaviors in a campaign. The process continues until the number of stages decided randomly is completed. Then, we insert them independently into the background events based on user-specific campaign duration. Algorithm 1 presents the workflow for campaign generation and describes the details in the following sections.

4.3 Malicious Behavior Event Generation

4.3.1 Targeted Attack Lifecycle Transition

As described in Section 2.3, threat actors follow a systematic and strategic approach, executing their actions sequentially. We adhere to Mandiant's adversary lifecycle to generate the sequence of malicious events. Initially, the campaign begins with two stages: *initial compromise* and *establish foothold*. Subsequently, the four stages: *Escalate Privileges*, *Internal Reconnaissance*, *Move Laterally*, and *Maintain Presence*, may occur none or multiple times during a campaign. Finally, the last step, *Complete Mission*, is executed none or once, marking the successful completion of the attack mission. These steps guide our campaign generation process in selecting attack templates. Given the current stage, the sequence of generated events and the desired number of stages in the campaign, we determine the next stage accordingly.

4.3.2 Attack Template Selection

The selection of an attack pattern template is based on the current stage and the generated attack events. A campaign information table is created to maintain the selected attack templates associated with each stage of the lifecycle. When given the current stage, each template in the stage is examined to determine which one's prerequisites are satisfied by the outcome of the generated attack events. More specifically, the campaign information table holds the outcomes of the generated attack events in the previous stage and is used to match each template's prerequisites in the current stage.

4.3.3 Malicious Artifact Substitution

Once an attack template is chosen, the system entities within the template are replaced to create an attack event that represents a variety of attack behavior. We collect malicious artifacts from VX-Underground [3] and VirusTotal [2]. From VX-Underground, the largest online repository of malicious software, we collected approximately 10,000 hash values of malware used in APT campaigns over five years. Next, we extract artifacts associated with system entities from behavior reports by querying each collected hash value to VirusTotal, which provides behavioral information from a sandbox. For example, by querying a sample hash value like 78a3e4702d9fc13b2ef917211cd65b44, we can find a file located at *%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\twainResolver.lnk*. The dropped file, *twainResolver.lnk*, associated with the entity type *file*, can be used as an email attachment in an attack behavior such as *Spearphishing Attachment*.

Once we have collected attack events for all stages as a sequence of labeled audit events, we reorganize the sequence to ensure that the events maintain causal relationships within the same process. Specifically, the process ID of an event e_j

follows the process ID of an event e_k when they are correlated, where $\forall\ k \in [0, j-1]$ from the preceding stages. Thus, we obtain a series of consolidated malicious events spanning multiple stages, forming the foundation for the development of malicious campaigns. Once we collect attack events for all stages as a sequence of labeled audit events. We reorganize the sequence to ensure that the events have causal relations in the same process. Thus, we have a series of consolidated malicious events across multiple stages, forming the basis for the development of malicious campaigns.

4.4 Malicious Event Insertion

When provided with a series of background events B, the generated malicious event e_k is sequentially inserted into the background events B to form the desired campaign audit events E. The time interval d for event insertions is randomly generated. Users can define the time interval d for each stage, allowing the generated campaigns to persist for various durations, ranging from less than an hour to, for example, more than a year. In Figure 2 ④, we visually illustrate the concept of malicious event insertion. It is essential to emphasize that there is no requirement to continuously insert events linked to a technique into the background events. For example, events associated with establishing a foothold, as shown in Figure 2 ④, do not need to be continuously inserted.

Algorithm 1 APT Campaign Generation

Input: Attack pattern templates T, Background events B
Output: Synthesized audit events E

Set $i \leftarrow 0$, $j \leftarrow 0$, $seq \leftarrow \{\emptyset\}$
Set $E \leftarrow B$, $ms \leftarrow B$'s initial time
$n \leftarrow$ Generate a random integer $n > 3$
while $|seq| < n$ **do**
  $t_j \leftarrow$ AttackPattenTemplateSelection(i, seq)
  // select Template t_j at Stage i, s.t. t_j meets e_k's requirement, where $\forall e_k \in seq$
  $e_j \leftarrow$ ArtifactSubstitution(t_j)
  // Substitute Template t_j's artifact as an event e_j
  $seq \leftarrow \{seq \cup \{e_j, j\}\}$
  $i \leftarrow$ NextStage(i, seq, n)
  // Determine the next stage s.t. cyberattack lifecycle
  $j \leftarrow j + 1$
end while
$seq \leftarrow CausalEventsUnifier(seq)$
// Unify the system entities of the event e_k where $\forall e_k \in seq$
for each event $e_k \in seq$ **do**
  $d \leftarrow$ Generate a random time interval
  $ms \leftarrow ms + d$
  $E \leftarrow$ EventInsertion(ms, e_k, E)
  // Insert event e_k at time step ms to E
end for

5 APT Attack Detection

ATP attack detection consists of provenance graphs construction, Technique hunting, and campaign matching. The initial step involves the processing of audit events to construct data provenance, ensuring causality in the provenance analysis. Subsequently, we introduce a neural network designed to identify attack behaviors for TTP hunting (Section 5.1). Finally, we evaluate the likelihood between the identified attack behaviors and known campaigns to determine whether an attack is unfolding (Section 5.2).

Reduce the complexity of the graph to ensure the effectiveness and precision of the detection. We followed two graph reduction techniques from previous work, *Causality preserved reduction (CPR)* [65] and *LogApprox* [45]. We adopt *CPR* to merge edges between vertices with identical operations and only retain the latest edge according to their timestamps. We also apply *LogApprox* to develop regular expressions that define common access patterns to simplify the repetition of file operations.

5.1 Technique Hunting

Due to the imbalance of the numbers of benign processes and malicious processes, we first identify benign processes from a provenance graph G and then remove them for ability recognition. Next, we develop a neural network model to identify malicious audit events from the remaining processes. This procedure involves event embedding, abnormal detection, and malicious event recognition.

5.1.1 Event Embedding

An embedding function transforms textual information into a numerical vector, called embedding, which may preserve the semantics of input values, or even structural relations. Here, we consider SecureBERT [7] to convert the system entities and relations of a graph into fixed-length vectors, and employ principal component analysis (PCA) [34] for dimension reduction. SecureBERT, a cybersecurity language model, extends the capabilities of the state-of-the-art language model architecture called BERT [35] by training on a variety of cybersecurity text resources. With pre-training specifically in cybersecurity language, SecureBERT has an inherent grasp of the semantics at both word and sentence levels, forming a fundamental basis for the embedding function for audit events.

As shown in Figure 5, an event consists of a source node v_i, a destination node v_j, and an edge $e_{i,j}$ that connects v_i to v_j. First, each of them is tokenized; for example, the path *c:\program files\mozilla firefox\firefox.exe* is segmented into 13 tokens, *[c] [:\] [program] [files] [\] [mo##] [##zilla] [firefox] [\] [fire##] [##fox] [.] [exe]* , seen as words in a sentence. SecureBERT learns the latent information concerning

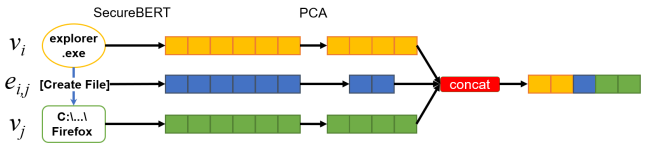

Figure 5: Embedding of an audit event.

the directory hierarchy based on the context of the given tokens; that is, any executable file located in *C:\Program Files* is recognized as an installed application within the operating system. Also, the word *firefox.exe* corresponds to four tokens, [fire##] [##fox] [.] [exe], which handles rare words by using a segmentation method. This gives us the flexibility to manage unknown words or new application names.

Next, SecureBERT produces a numerical vector for each given word. A node or edge is represented by averaging all the vectors, and the resulting representations are concatenated together. We also apply PCA (Principal Component Analysis) [34] to reduce the vector dimensions before concatenating them for efficient computation. The subsequent work, anomaly and technique detection, leverage these embeddings, as features to represent a single event.

5.1.2 Anomaly Detection

In real-world scenarios, there is a significant imbalance between the number of attacks and benign events. For example, in the DARPA Five-Dimensional dataset [19], we noted that the number of attack events in a scenario is 5,308, while the number of benign events is approximately 14,005,875. A classifier tends to exhibit a preference for the majority class due to its inability to effectively learn from the minority class in an imbalanced dataset. In this study, we use the one-class support vector machine (SVM) [63] to identify abnormal processes by detecting any malicious events occurring within a process. The one-class SVM learns benign patterns to establish a decision boundary, categorizing samples situated far from the learned benign patterns as potentially malicious.

In the training dataset for anomaly detection, we generated 1,000,000 normal events as benign events and 169,000 attack instances as abnormal events. These attack instances are generated by 169 abilities in our implementation, resulting in 100 malicious processes each. Each event is represented as an embedding and used as input to train the anomaly detection classifier. For the test dataset, we evaluated 20,000 events, with an equal split between benign and malicious events. The results indicate a high precision of 78.4% and a recall of 88.0% against both normal and malicious events.

5.1.3 Attack Pattern Detection

After the anomaly detection, events involving malicious entities within processes are processed for ability recognition. Previous studies [8] have shown Recurrent Neural Networks

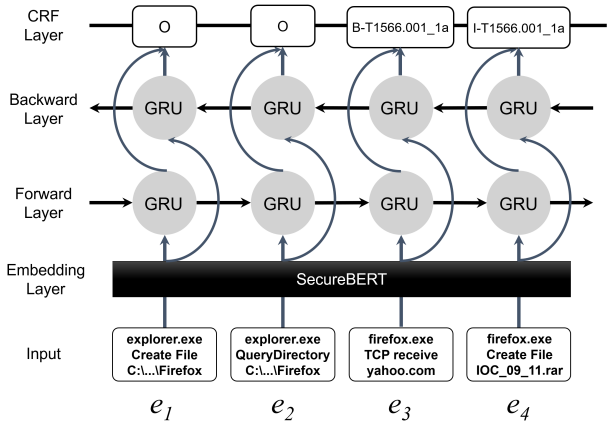

Figure 6: The neural network architecture of Bi-GRU-CRF for Attack behavior (ability) detection.

(RNNs) are an effective approach to processing attack sequences and differentiate reflected patterns in attack and non-attack sequences. Similarly, we consider Bidirectional Gated Recurrent Units (Bi-GRU) [14], a subtype of RNNs, to identify ability in chronological order. The effectiveness of GRU extends to various sequence-based learning tasks, including machine translation [14] and Speech Recognition [50]. We utilize Bi-GRU to automatically learn a model proficient in identifying patterns among sequences. While a single-directional (forward) GRU promptly processes a window size of events in a given sequence, bi-directional (forward and backward) GRUs extend their capability by backtracking to the causally relevant antecedent node for the current event.

Next, we incorporate a Conditional Random Field (CRF) [37] for the jointly decoding of labels throughout the entire sequence. When tackling sequence labeling tasks, or more broadly, general structured prediction tasks, it is advantageous to take into account the correlations among labels in neighboring positions. This approach allows for the collaborative decoding of the optimal chain of labels for a given input sequence. Take an example in Figure 6, the event e_4 is more likely to be followed by the event e_3 than another events, which highly correlated to the attack behavior "Spearphishing Attachment." In light of this, we decide to jointly model the label sequence using a CRF layer, as opposed to the alternative method of independently decoding each label.

A depiction of our network's architecture for attack behavior detection can be found in Figure 6. Four events are fed into the model chronologically, that is, the *Windows File Explorer* (*explorer.exe*) open a file (Firefox), queries the directory in the file is located, the file receives a TCP packet, and downloads a malicious payload (*IOC_09_11.rar*). For each event where two nodes establish a connection via an edge, SecureBERT computes the representation for each node according to the details in Section 5.1.1. These event representations are then combined through concatenation to create the inputs, which are subsequently input into the Bi-GRU networks. The re-

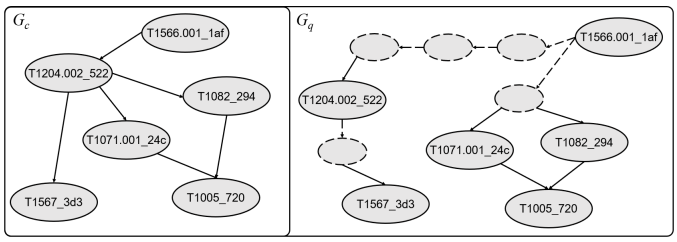

Figure 7: A simplified APT28 campaign graph (G_c), a discovered technique query graph (G_q).

sulting output vectors from Bi-GRU are further processed through the CRF layer to jointly decode the most optimal label sequence, following the BIO2 annotation [58], that is, O representing outside of labels and the I-XXX for events inside an ability of label XXX, and B-XXX showing it begins an ability of label XXX. Our synthesized ability dataset consists of 16,900 sequences, where 169 abilities contribute to the generation of 100 malicious processes and split them into 8:1:1 for training, validation, and test sets, respectively. Results revealed a high precision rate of 97.81% and a recall rate of 97.24% for each of the ability behaviors.

Figure 7 excerpts several discovered abilities that are transformed into a query graph G_q for matching known campaign G_c. The definition of the query graph G_q entails that a node v signifies discovered techniques, and a directed edge $e_{i,j}$ indicates the temporal causal connection between two abilities. When an event is labeled with an ability, the corresponding techniques are employed to represent the discovered node. In cases where multiple events within the same process are classified under the same discovered ability, they are consolidated into one technique; otherwise, they are considered independently. If two labeled events, occurring in distinct processes but involving the same entity, are identified, a directed path is established to link their associated techniques in a temporal order. We emphasize that the query graph constructed at this stage includes causally connected, discovered attack behaviors, which can be regarded as a summary of the actual attack graph.

5.2 APT Campaign Matching

We model the techniques used in a campaign appearing as a labeled and directed graph, which we refer to as a campaign graph G_c, where $c \in C$. This data can be retrievable from well-documented CTI reports or reputable knowledge bases like the MITRE ATT&CK website. When evaluating the discovered techniques represented as a query graph G_q and determining the likelihood $(G_q :: G_c)$ of its match to a known campaign G_c, as Figure 7 shown, we can frame this as a subgraph isomorphism problem. This challenge can be tackled using algorithms such as the Ullman algorithm [59] and VF2 [15]. However, we have observed that the predicted nodes within G_q often do not align consistently with the nodes in the given known campaign G_c in many cases. For example,

Algorithm 2 APT Campaign Matching

Input: a query graph G_q,
a set of known campaign graphs G_C
Output: an ordered set of known campaigns C

for each campaign graph $G_c \in G_C$ **do**
  $G'_q \leftarrow$ ExtractSubgraph(G_q, G_c)
  // Extract V and E from G_q matching G_c
  $cost_c \leftarrow GED(G'_q, G_c)$ in Eq (1)
end for
for each campaign $c \in C$ **do**
  $score_c \leftarrow max(cost) - cost_c$
  $score_{norm,c} \leftarrow \frac{score_c}{||score||_2}$
end for
$C \leftarrow argsort(\left[\frac{\exp(score_{norm})}{\sum_{i=1}^{C}\exp(score_{norm,i})}\right])$

an APT attack may change from one to another. Consequently, the techniques implemented by a known campaign may be different in a given instance. To address this challenge, we employ the graph edit distance (GED) [6] to quantify the cost of editing G_q to resemble G_c. This method considers the costs associated with deletion, substitution, and insertion operations.

$$GED(G_q, G_c) = \min_{o_1,\ldots,o_m \in \gamma(G_q,G_c)} \sum_{i=1}^{m} cost(o_i) \qquad (1)$$

where $cost(o_i)$ denotes the cost function assessing the strength of an edit operation o_i, and $\gamma(G_q, G_c)$ represents the set of editting nodes and paths transforming G_q into G_c.

Algorithm 2 presents the workflow of campaign matching. First, we form a subgraph G'_q of the predicted graph by matching nodes V and edges E to a given known campaign graph G_c. Next, the cost function (GED) computes the cost of operations for transforming G'_q into G_c. To rank the likelihood of a given set of known campaigns G_c, where $c \in C$, we initially invert the cost values by subtracting the maximum value as a score and then normalize them. Finally, the softmax function is used for sorting the likelihood of the known campaigns.

6 Evaluation

In this section, we evaluate the proposed framework on 28 attack scenarios. The section begins by outlining our datasets, experimental settings, and the specifications of the machine's hardware and software configurations. Three evaluations form the core of our analysis: 1) assessing the accuracy of TTP hunting, 2) conducting an ablation study and 3) gauging the effectiveness of campaign attribution. In addition, we offer a case study that demonstrates how our approach operates in an actual real-world situation.

6.1 Evaluation Settings

Dataset. We implemented 169 abilities with a red-team emulator, MITRE CALDERA [1], for generating malicious audit events at every attack stage. For benign system events, similar to prior research [8,44], we endeavor to imitate a range of normal user activities on the same machine, including watching YouTube, downloading files from github, browsing cnn.com, checking Gmail. To report the effectiveness of our framework, due to the limited availability of attack datasets and system logs containing labeled techniques and campaigns, we have created 8 attack scenarios based on cyber threat intelligence reports about real-world APT campaigns, that is, Higaisa [40], admin338 [41], APT28 [10], FIN7 [42], CobaltGroup [49], Gamaredon [13], Patchwork [17], GorgonGroup [60], more details in Apendix.B. Additionally, we generated 20 more attack activities using our threat model in Section 4. For each self-generated scenario, we followed the cyber attack lifecycle outlined in Table 1, along with providing a comprehensive overview of labeled abilities in Appendix.C. The labeling of malicious audit events is conducted using the BIO2 scheme, as depicted in Figure 6. In total, 96 abilities were assessed with 57 techniques among 28 attack campaigns.

Metrics. In line with the event-based methodology for attack pattern detection, following a similar work [8], we present the detection results based on events. It is important to acknowledge that an event labeled as either B-XXX or I-XXX is counted associated with the label XXX. For each attack scenario dataset, we present macro-average precision (P), recall (R), and F1 scores, computed as the arithmetic mean of individual class classification metrics. Please be aware that each scenario involves a distinct set of abilities, and the evaluations consider only the classes within the attack datasets, excluding the label O.

Implementation details. For our experiment, we employed an operating system setup comprising Windows 10 with 8GB of memory as our victim system and an AMD EPYC 7282 16-Core Processor CPU, as well as Ubuntu 20.04 with 1TB of memory and an NVIDIA A100 80GB GPU for model development. The neural network model training, serving as the backend, utilized PyTorch 1.13.1 and Python 3.10. We employed ThunderSVM [63] to train the one-class SVM, configuring the kernel as linear and setting the *nu* value to 0.02. SecureBERT from huggingface [33] is employed to transform an event, consisting of two nodes and one edge, resulting in three 768-dimensional vectors. These vectors are then subjected to PCA reduction with scikit-learn library [54], yielding dimensions of 256, and 16 for node and edge, respectively. Finally, the reduced vectors are concatenated to generate a 528-dimensional embedding vector. In terms of the ability detection model parameters, we employed a bi-GRU model with a hidden size of 128 dimensions. The window size is configured to be 256, and the stride for the subsequent windows is set to 64. The training process utilized the Adam optimizer with a learning rate of 0.001, a mini-batch size of 64, and was conducted over 10 epochs. We employed the NetworkX 2.8 package [48] for calculating graph edit distance. The costs associated with deletion, substitutions, and insertions were uniformly set to 1.

6.2 Evaluation on Effectiveness

In this section, we compare the performance of the proposed ability detection and Sigma [5] on the 28 attacks. To our knowledge, there is an absence of a standardized benchmark aimed at evaluating the accuracy of fine-grained attack patterns. Thus, we have chosen Sigma as our baseline for comparative evaluation. Sigma, as a generic and open signature format, serves as a generic signature format for describing relevant log events. Many defensive security practitioners around the world collaboratively contribute their detection rules. A major portion of these rules align with the MITRE ATT&CK framework; 70 rules mapped to the techniques could be found in our attack scenarios.

The effectiveness of ability detection at identifying events for each dataset is reported in Table 1. Our methodology exhibits substantial performance distinctions compared to Sigma. Across various attack scenarios, both methods showcase significant fluctuations in their performance. On average, our experimental results demonstrate 62.05% precision, 69.88% recall, and 60.57% F1, whereas Sigma's results are notably lower at 25.03%, 20.78%, and 18.45%, respectively. The highest performance was attained in dataset C-1, yielding values of 90.32%, 90.48%, and 81% for precision, recall, and F1, respectively. On the contrary, the least favorable results were observed in dataset G-7, with percentages of 20.89%, 65.38%, and 27.7% for precision, recall, and F1. One of the contributing factors is the occurrence of high false positives associated with specific attack behaviors across different datasets. Another potential factor is the significant imbalance in the ratio of malicious events (i.e., 0.001%), leading to a substantial number of benign activities that impact the assessment of attack behavior. Please note that, regarding Sigma rules, which are crafted by human experts, there is a possibility of incomplete matching with attack behaviors. Thus, no malicious behaviors were accurately detected in datasets C-3, G-7, G-14, and G-17. This illustrates the advantages of synthesized datasets, wherein a learning-based method can acquire and comprehend attack patterns from the provided data.

6.3 Ablation Study

Our detection method includes the embedding technique (SecureBERT), anomaly detection (one-class SVM), neural network model for ability detection (BiGRU-CRF). We conducted an ablation study to understand the contributions of each component to ability detection using the C-1 dataset. Initially, SecureBERT is replaced with either BERT [35],

Table 1: Event-based investigation results.

Dataset					SVM+BiGRU-CRF			Sigma		
ID	Attack Seq.	Ability	Event	MalEvent	P	R	F1	P	R	F1
C-1	{1,2,6,4,4,6,6}	7	607,416	0.005%	90.32%	90.48%	87.00%	33.37%	36.11%	33.40%
C-2	{1,2,4,4,4,4,4,4}	8	950,436	0.006%	63.78%	60.34%	52.85%	12.52%	18.15%	11.30%
C-3	{1,2,2,4,4,7}	6	1,203,013	1.175%	56.30%	62.45%	57.02%	0.00%	0.00%	0.00%
C-4	{1,2,6,6}	4	2,072,151	0.001%	40.95%	38.33%	28.94%	30.00%	40.00%	33.33%
C-5	{1,2,4}	3	961,920	0.118%	54.82%	72.31%	58.44%	0.28%	29.75%	0.54%
C-6	{1,2,2,6,6,4,4,6,7}	9	442,729	0.013%	73.51%	77.75%	73.21%	25.02%	17.08%	16.71%
C-7	{1,2,3,4,4,4,6,5}	8	155,296	9.095%	68.60%	68.87%	67.55%	8.13%	21.96%	9.14%
C-8	{1,2,6,6,6,6,6}	7	844,723	0.006%	33.22%	59.84%	37.06%	4.77%	28.57%	7.17%
G-1	{1,2,3,3,4,6,6,4,7}	9	558,368	0.006%	70.05%	69.14%	64.22%	27.78%	15.74%	15.03%
G-2	{1,2,4,6,3,4,6,7}	8	232,414	0.065%	78.73%	81.06%	77.77%	33.33%	16.75%	20.04%
G-3	{1,2,4,6,4,3,3,7}	8	424,376	0.013%	86.44%	79.61%	81.91%	38.64%	17.71%	20.73%
G-4	{1,2,3,6,3,4,6,4,7}	9	359,970	0.031%	60.15%	70.56%	62.42%	41.50%	29.32%	30.91%
G-5	{1,2,6,3,4,6,4,3,7}	9	423,459	0.047%	50.27%	65.40%	50.22%	46.67%	25.53%	26.44%
G-6	{1,2,3,6,4,3,7}	7	96,176	0.133%	78.63%	76.91%	76.64%	22.92%	25.00%	23.86%
G-7	{1,2,3,6,6,7}	6	3,451,541	0.001%	20.89%	65.38%	27.70%	0.00%	0.00%	0.00%
G-8	{1,2,3,6,4,4,3,6,7}	9	2,348,338	0.004%	60.98%	88.34%	64.20%	59.26%	37.20%	40.51%
G-9	{1,2,6,3,3,4,7}	7	1,936,581	0.010%	58.71%	74.48%	58.19%	45.83%	42.78%	32.68%
G-10	{1,2,4,3,3,6,4,7}	8	233,526	0.078%	73.49%	78.88%	75.38%	33.33%	5.63%	8.87%
G-11	{1,2,3,6,4,3,6,7}	8	425,309	0.007%	69.02%	64.81%	62.20%	42.86%	23.81%	27.89%
G-12	{1,2,4,6,4,6,3,7}	8	3,361,052	0.002%	42.94%	66.69%	41.98%	29.17%	26.79%	27.50%
G-13	{1,2,6,6,4,4,7}	7	285,053	0.014%	58.81%	68.75%	61.24%	14.29%	14.29%	14.29%
G-14	{1,2,3,7}	4	402,268	0.006%	66.59%	62.06%	59.86%	0.00%	0.00%	0.00%
G-15	{1,2,6,4,6,7}	6	1,305,571	0.002%	45.53%	57.14%	47.05%	33.33%	20.83%	23.33%
G-16	{1,2,6,4,6,3,4,3,7}	9	86,357	0.141%	80.39%	77.16%	77.72%	30.96%	13.78%	16.01%
G-17	{1,2,3,4,3,7}	6	298,825	0.007%	65.58%	62.12%	57.70%	0.00%	0.00%	0.00%
G-18	{1,2,4,4,3,6,6,3,7}	8	403,077	0.048%	70.33%	79.86%	72.57%	33.33%	22.56%	24.12%
G-19	{1,2,4,4,3,3,7}	7	372,137	0.010%	51.09%	58.82%	53.54%	3.57%	2.38%	2.86%
G-20	{1,2,6,3,4,3,4,7}	8	718,873	0.004%	67.35%	78.97%	61.31%	50.00%	50.00%	50.00%
Avg.		7.25	891,463	0.395%	62.05%	69.88%	60.57%	25.03%	20.78%	18.45%

Table 2: An ablation study on the C-1 dataset, where SBERT denotes SecureBERT.

Model	P	R	F1
SBERT + SVM + BiGRU-CRF	90.32%	90.48%	87.00%
BERT + SVM + BiGRU-CRF	84.97%	91.90%	82.69%
TransE + SVM + BiGRU-CRF	46.02%	43.10%	43.06%
TransH + SVM + BiGRU-CRF	42.81%	39.52%	39.99%
SBERT + BiGRU-CRF	87.79%	80.65%	77.40%
SBERT + MalEvent + BiGRU-CRF	100.00%	93.65%	95.92%
SBERT + SVM + Bi-GRU-FC	86.07%	90.77%	84.62%
SBERT + SVM + GRU-CRF	72.73%	80.00%	73.68%
SBERT + SVM + GraphSage	60.08%	82.14%	62.36%

TransE [11], or TransH [62] for embedding investigation. Following this, one-class SVM is either removed or replaced with the gold malicious event to examine anomaly detection. Lastly, the CRF layer is replaced with a fully connected (FC) layer, a single GRU is adopted, or the sequential neural network is substituted with a graph-based network, GraphSage [25].

Table 2 demonstrates the impact of various modifications. In terms of embedding techniques, we make comparisons between our approach (SecureBERT) and the other three methods. The findings indicate that a domain-specific pre-trained model, SecureBERT, surpasses the performance of an existing language model (BERT) and translation-based embedding techniques like TransE or TransH in effectively representing cybersecurity-related tokens in texts. One interesting observation is that BERT excels in correctly identifying more events when discerning the technique *T1204.002 User Execution: Malicious File*, yet experiences a decline in performance when attempting to recognize *T1566.001 Phishing: Spearphishing Attachment*. Static translation-based embedding models provide fixed representations that overlook textual changes, such as encountering unknown filenames in a system entity. In contrast, dynamic embeddings (such as BERT and SecureBERT) exhibit adaptability, adjusting to changes in context.

When evaluating the effectiveness of anomaly detection, the exclusion of one-class SVM and the incorporation of truly labeled malicious events respectively serve as the lower and upper bounds of performance for analyzing attack behaviors. The occurrence of malicious events in a real-world campaign is notably limited. Consequently, the implementation of effective outlier detection emerges as a crucial factor in the subsequent processing stages, as evidenced by the outcomes observed in the SBERT + MalEvent + BiGRU-CRF model.

Please be aware that we adopt the default setting for anomaly detection, ensuring fair performance. This approach allows for a higher tolerance of false positives, facilitating subsequent recognition of attack behavior.

When it comes to identifying abilities, the replacement of each neural network component (FC layer, single GRU, and GNN) demonstrates a negative impact on performance. Adding a CRF layer for joint decoding achieves obvious improvements over the Bi-GRU-FC model, which demonstrates the joint decoding of attack sequences can bring about a substantial improvement in the overall performance of neural network models. In the case of recognizing *T1204.002 User execution*, our detection method successfully establishes the connection among system entities (*WINWORD.exe*, *cmd.exe*, and *Retrive4075693065230196915.vbs*) over 11 discontinuous events. Conversely, the Bi-GRU-FC model fails to discern the association between two processes, *WINWORD.exe* and *cmd.exe*, treating these events as independent instances.

Next, the bi-directional GRU has a significant impact on attack behavior recognition, as capturing both forward and backward relations in a sequence proves to be crucial. For instance, we noticed that an event checking for the existence of a shortcut file {*cscript.exe, CreateFile, C:\...\Startup\sllauncherENU.dll (copy).lnk*) might suggest an attempt to create a shortcut (*T1547.009 Shortcut Modification*}. However, this observation may not consistently remain valid, particularly when subsequent events involve writing the shortcut to the startup folder {*cscript.exe, WriteFile, C:\...\Startup\sllauncherENU.dll (copy).lnk*}. Consequently, SBERT + SVM + GRU-CRF misclassified these events, whereas our approach accurately assigns them to the correct technique.

Lastly, a graph-based neural network, GraphSage [25], aggregates node information from their local 2-hop neighborhoods to capture broader contextual information within a graph. This neural network architecture is expected to proficiently combine both immediate and adjacent node attributes, delivering a comprehensive understanding of each entity's role and interactions within a graph. However, the absence of sequential information may present a notable challenge in the recognition of abilities, resulting in a dramatic drop in precision.

6.4 Campaign Evaluation

Twenty-eight campaigns constitute the set of known campaigns G_C used to assess the effectiveness of campaign matching. Figure 8 shows that top-k score values achieved for the top 1, 3, and 5 campaigns are 0.357, 0.714, and 0.857, respectively. This indicates that the predicted techniques establish comprehensive temporal causal relations for attribution, distinguishing the most likely scenarios from others. Unfortunately, G-18 encountered a notable increase in substitution and insertion costs due to the forecasted abundance of nodes in diverse

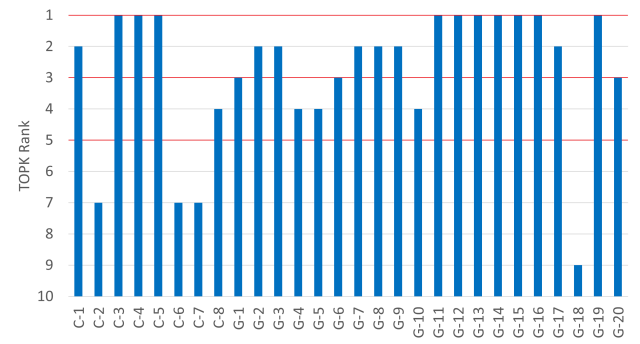

Figure 8: Results on campaign matching.

processes, ultimately yielding only the top 9. To sum up, our approach delivers the likelihood of potential campaigns, offering essential insights for SOC analysts and defenders.

6.5 APT 28 Case Study

The attack scenario, C-3 APT28 serves as a case study to illustrate the identification process of events associated with techniques, highlighting the strengths and limitations of our approach. Table 3 provides an overview of the classification results for identifying techniques implemented by threat actors. Our approach demonstrates successful recognition of the majority of events associated with *System Information Discovery*, *Data from Local System*, and *Exfiltration Over Web Service*. Events involving a dropped file (WinRAR.exe) executing *PowerShell.exe* are accurately attributed to *System Information Discovery*. Furthermore, the identification of over ten thousand events querying files in a specified folder using *PowerShell.exe*, such as *{powershell.exe, QueryDirectory, C:\Users\lmelendez}*, followed by saving the collected data in a file *output.json* and transmitting it to a C2 server, is appropriately labeled as *Data from Local System* and *Exfiltration Over Web Service*. The success of our approach in accurately detecting certain audit events stems from a comprehensive and context-aware analysis of system entities and their interactions within the given graph.

We also achieved moderately acceptable results in detecting *User Execution: Malicious File*, successfully capturing two disconnected events: *WinRAR.exe, Process Create, cmd.exe* and *cmd.exe, CreateFile, HASH.bin*. These events indicate that a user triggered the exploited file *WinRAR.exe*, leading to the creation of an executable file with a random hash using *cmd.exe*. However, our model failed to identify similar events when the *cmd.exe* subsequently manipulated *cscript.exe* for the same operations, the creation of an executable file with a random hash. In other words, our model encountered difficulties in recognizing a large volume of similar events where *cmd.exe* subsequently manipulated *cscript.exe* for identical operations, specifically the creation of an executable file with a random hash. This indicates our approach may suffer from the limitation of capturing the causal relation from the immedi-

Table 3: Analysis of APT28: the classification results.

Technique	P	R	F1
Spearphishing Attachment	0.53%	16.67%	1.04%
User Execution: Malicious File	58.33%	43.75%	50.00%
Web Protocols	26.67%	14.29%	18.60%
System Information Discovery	100.00%	100.00%	100.00%
Data from Local System	83.69%	100.00%	91.12%
Exfiltration Over Web Service	68.57%	100.00%	81.36%

ately manipulated system entities but also the other correlated system entities of those immediate ones. Unfortunately, our method exhibits a high rate of false positives when identifying *Phishing: Spearphishing Attachment*. The misclassification might be linked to the events involving connections to a mail server for attachment downloads, sharing commonalities with the behavior of *Remote Access Software*, known for connecting to external sites for remote access tool downloads.

When the predicted techniques are considered as a query graph G_q, APT-28 emerges as the most matching result when compared to the pool of known campaigns G_C. One reason for this could be that the techniques are accurately identified even when some events are misclassified. Additionally, the number of editing operations in Equation 1 is 26, which is relatively lower compared to other campaigns.

7 Related Work

Learning-based Provenance analysis. Learning-based provenance analysis methods have shown notable improvements in detecting intrusions [69] and have accelerated the investigative process of various attacks [8, 38, 67, 68]. In this work, our primary focus is on fine-grain technique discovery, and we tailor our methods accordingly to address this specific emphasis. WATSON [67] and ShadeWatcher [68] focus on translation-based embedding techniques, i.e., TransE and TransH, to comprehend the structure within an event. In contrast, our methodology involves leveraging a domain-specific pre-trained language model (SecureBERT) to understand the contextual information within a system entity, yielding a better performance in our ablation study. We adhere to the methodologies proposed by Atlas [8] and DeepAG [38], employing recurrent neural networks to process audit events. Additionally, we integrate a conditional random field for joint label decoding, which has been instrumental in achieving superior performance, as evidenced by our ablation study.

Attack pattern recognition. HOLMES [47] was designed to generate a detection signal score alongside 16 rule-matching attack patterns to construct a high-level scenario graph (HSG) embedded with Tactics, Techniques, and Procedures (TTPs). This framework serves as an indicator of the presence of a specific set of activities associated with an APT campaign. RapSheet [28] leveraged a set of 67 rules within a commercial Endpoint Detection and Response (EDR) tool to create a tactical provenance graph aligned with MITRE ATT&CK, with a specific emphasis on alert correlation. KRYSTAL [36] utilizes Sigma [4] to detect known attack patterns and converts them into SPARQL query expressions against the provenance graph for potential threat detection. Establishing rules to identify recognized attack patterns proves to be an effective strategy; however, the manual creation and upkeep of this set of rules and signatures can be demanding in terms of labor. Our study adopts a data-driven approach, utilizing a red-team emulator to gather audit events from a host and subsequently constructing a machine-learning detection model designed to identify attack patterns.

APT campaign attribution. A multi-view analysis in [53] identified APT groups and their associated attack campaigns by examining their methodologies, TTP; that is, classifying malware, considering characteristics of malware like Opcode sequence, Bytecode sequences, and headers, attributing to 12 APT groups. CSKG4APT [51] profiled an attacker organization based on the construction of an APT knowledge graph by drawing observations from real-world APT attack scenarios, and an analysis engine by applying the intrusion analysis diamond model [12]. However, both approaches attribute a threat actor through the observation of malware code sequences and analysis of CTI reports, which diverges from our primary focus on the identification of a substantial volume of audit events. Another relevant work, APTer [52], evaluates numerous alerts from IDS/IPS and SIEM systems, mapping them to MITRE ATT&CK TTPs trained for attribution to an APT group. In this work, we propose a graph-matching method, specifically graph edit distance, to align the identified techniques with known campaigns. Our approach enhances the interpretation of security alerts by providing a comprehensive context that incorporates the relations within the discovered attack patterns.

8 Conclusion

This study aims to develop a generative threat model for synthesizing APT attack scenarios seamlessly integrated into audit logs. Moreover, we construct a machine learning-based methodology targeting discovering malicious behaviors, with the ultimate goal of identifying potential APT threat actors. The extensive evaluation results demonstrated that our method excelled in recognizing over 60% of events tied to techniques, surpassing Sigma's performance (less than 20%). This resulted in the attribution of 86% of campaigns within the top 5 ranks, highlighting the superior capabilities of our methodology in threat detection and attribution.

References

[1] Mitre caldera. https://github.com/mitre/caldera, 2023.

[2] Virus total. https://www.virustotal.com/gui/home/upload, 2023.

[3] Vx underground. https://vx-underground.org/, 2023.

[4] Sigma - generic signature format for siem systems. https://github.com/SigmaHQ/sigma, 2024.

[5] Sigma rule repository. https://github.com/SigmaHQ/sigma, 2024.

[6] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*, 2015.

[7] Ehsan Aghaei, Xi Niu, Waseem Shadid, and Ehab Al-Shaer. Securebert: A domain-specific language model for cybersecurity. In *International Conference on Security and Privacy in Communication Systems*. Springer, 2022.

[8] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. {ATLAS}: A sequence-based learning approach for attack investigation. In *30th USENIX security symposium (USENIX security 21)*, 2021.

[9] Andy Applebaum, Doug Miller, Blake Strom, Chris Korban, and Ross Wolf. Intelligent, automated red team emulation. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016.

[10] APT28. Cve-2023-38831 exploited by pro-russia hacking groups in ru-ua conflict zone for credential harvesting operations. https://blog.cluster25.duskrise.com/2023/10/12/cve-2023-38831-russian-attack, 2023.

[11] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[12] Sergio Caltagirone, Andrew Pendergast, and Christopher Betz. The diamond model of intrusion analysis. *Threat Connect*, 298(0704), 2013.

[13] CERT-EE. Gamaredon infection: From dropper to entry. https://www.ria.ee/sites/default/files/content-editors/kuberturve/tale_of_gamaredon_infection.pdf, 2021.

[14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[15] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10), 2004.

[16] Gideon Creech. *Developing a high-accuracy cross platform Host-Based Intrusion Detection System capable of reliably detecting zero-day attacks*. PhD thesis, UNSW Sydney, 2014.

[17] Cymmetria. Unveiling patchwork-the copy-paste apt. https://web.archive.org/web/20180825085952/https://s3-us-west-2.amazonaws.com/cymmetria-blog/public/Unveiling_Patchwork.pdf, 2016.

[18] DARPA. Operationally transparent cyber (optc) data release. https://github.com/FiveDirections/OpTC-data, 2021.

[19] DARPA. Transparent computing engagement. https://github.com/darpa-i2o/Transparent-Computing, 2021.

[20] Roman Daszczyszak, Dan Ellis, Steve Luke, and Sean Whitley. MITRE ATT&CK: Ttp-based hunting. https://www.mitre.org/sites/default/files/2021-11/prs-19-3892-ttp-based-hunting.pdf, 2021.

[21] Hailun Ding, Juan Zhai, Yuhong Nan, and Shiqing Ma. {AIRTAG}: Towards automated attack investigation by unsupervised learning with log texts. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[22] E-ISAC. Analysis of the cyber attack on the ukrainian power grid. https://nsarchive.gwu.edu/sites/default/files/documents/3891751/SANS-and-Electricity-Information-Sharing-and.pdf, 2016.

[23] FireEye. M-trends 2019: Celebrating 10 years of incident response reporting. https://www.mandiant.com/resources/blog/mtrends-2019-celebrating-ten-years-of-incident-response-reporting, 2019.

[24] FireEye. Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor. https://www.mandiant.com/resources/blog/evasive-attacker-leverag

es-solarwinds-supply-chain-compromises-with-sunburst-backdoor, 2020.

[25] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[26] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. In *Network and Distributed System Security Symposium*, 2020.

[27] Xueyuan Han, Xiao Yu, Thomas Pasquier, Ding Li, Junghwan Rhee, James Mickens, Margo Seltzer, and Haifeng Chen. {SIGL}: Securing software installations through deep graph learning. In *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[28] Wajih Ul Hassan, Adam Bates, and Daniel Marino. Tactical provenance analysis for endpoint detection and response systems. In *IEEE Symposium on Security and Privacy*, 2020.

[29] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*, 2019.

[30] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, 2017.

[31] Md Nahid Hossain, Sanaz Sheikhi, and R Sekar. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020.

[32] Yi-Ting Huang, Chi Yu Lin, Ying-Ren Guo, Kai-Chieh Lo, Yeali S. Sun, and Meng Chang Chen. Open source intelligence for malicious behavior discovery and interpretation. *IEEE Transactions on Dependable and Secure Computing*, 19(2), 2022.

[33] Huggingface. Securebert: A domain-specific language model for cybersecurity. https://huggingface.co/ehsanaghaei/SecureBERT, 2024.

[34] IT Jolliffe. Principal component analysis. *Technometrics*, 45(3), 2003.

[35] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[36] Kabul Kurniawan, Andreas Ekelhart, Elmar Kiesling, Gerald Quirchmayr, and A Min Tjoa. Krystal: Knowledge graph-based framework for tactical attack discovery in audit data. *Computers & Security*, 121:102828, 2022.

[37] John D Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[38] Teng Li, Ya Jiang, Chi Lin, Mohammad S Obaidat, Yulong Shen, and Jianfeng Ma. Deepag: Attack graph construction and threats prediction with bi-directional deep learning. *IEEE Transactions on Dependable and Secure Computing*, 20(1):740–757, 2022.

[39] Lockheed Martin Corporation. Gaining the Advantage: Applying Cyber Kill Chain Methodology to Network Defense. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf, 2015.

[40] Malwarebytes. New lnk attack tied to higaisa apt discovered. https://www.malwarebytes.com/blog/news/2020/06/higaisa, 2020.

[41] Mandiant. China-based cyber threat group uses dropbox for malware communications and targets hong kong media outlets. https://www.mandiant.com/resources/blog/china-based-threat, 2015.

[42] Mandiant. Fin7 evolution and the phishing lnk. https://www.mandiant.com/resources/blog/fin7-phishing-lnk, 2017.

[43] APT Mandiant. Exposing One of China's Cyber Espionage Units (2013). Accessed August 24, 2020.

[44] Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[45] Noor Michael, Jaron Mink, Jason Liu, Sneha Gaur, Wajih Ul Hassan, and Adam Bates. On the forensic validity of approximated audit logs. In *Annual Computer Security Applications Conference*, 2020.

[46] Sadegh M Milajerdi, Birhanu Eshete, Rigel Gjomemo, and VN Venkatakrishnan. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019.

[47] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *IEEE Symposium on Security and Privacy*, 2019.

[48] NetworkX. Network analysis in python. https://networkx.org/, 2024.

[49] ptsecurity. Cobalt strikes back: An evolving multinational threat to finance. https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Cobalt-2017-eng.pdf, 2017.

[50] Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2), 2018.

[51] Yitong Ren, Yanjun Xiao, Yinghai Zhou, Zhiyong Zhang, and Zhihong Tian. Cskg4apt: A cybersecurity knowledge graph for advanced persistent threat organization attribution. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[52] Vinay Sachidananda, Rajendra Patil, Akshay Sachdeva, Kwok-Yan Lam, and Liu Yang. Apter: Towards the investigation of apt attribution. In *2023 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2023.

[53] Dilip Sahoo. Cyber threat attribution with multi-view heuristic analysis. *Handbook of Big Data Analytics and Forensics*, 2022.

[54] scikit-learn: machine learning in Python. sklearn.decomposition.pca. sklearn.decomposition.PCA, 2024.

[55] Yun Shen and Gianluca Stringhini. {ATTACK2VEC}: Leveraging temporal word embeddings to understand the evolution of cyberattacks. In *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

[56] Blake Strom, Andy Applebaum, Doug Miller, Kathryn Nickels, Adam Pennington, and Cody Thomas. MITRE ATT&CK: Design and philosophy. https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf, 2018.

[57] Microsoft Sysinternals. Process monitor v3.96. https://learn.microsoft.com/en-us/sysinternals/downloads/procmon, 2023.

[58] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003.

[59] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1), 1976.

[60] Unit42. The gorgon group: Slithering between nation state and cybercrime. https://unit42.paloaltonetworks.com/unit42-gorgon-group-slithering-nation-state-cybercrime/, 2018.

[61] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A Gunter, et al. You are what you do: Hunting stealthy malware via data provenance analysis. In *NDSS*, 2020.

[62] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[63] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19, 2018.

[64] Guo-Wei Wong, Yi-Ting Huang, Ying-Ren Guo, Yeali Sun, and Meng Chang Chen. Attention-based api locating for malware techniques. *IEEE Transactions on Information Forensics and Security*, 19, 2024.

[65] Zhang Xu, Zhenyu Wu, Zhichun Li, Kangkook Jee, Junghwan Rhee, Xusheng Xiao, Fengyuan Xu, Haining Wang, and Guofei Jiang. High fidelity data reduction for big data security dependency analyses. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016.

[66] Fan Yang, Jiacen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. Prographer: An anomaly detection system based on provenance graph embedding. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[67] Jun Zeng, Zheng Leong Chua, Yinfang Chen, Kaihang Ji, Zhenkai Liang, and Jian Mao. Watson: Abstracting behaviors from audit logs via aggregation of contextual semantics. In *NDSS*, 2021.

[68] Jun Zengy, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 489–506. IEEE, 2022.

[69] Michael Zipperle, Florian Gottwalt, Elizabeth Chang, and Tharam Dillon. Provenance-based intrusion detection systems: A survey. *ACM Computing Surveys*, 55(7), 2022.

Appendix A

In this appendix, we introduce the threat model and assumptions, as well as the figure illustrating the provenance graph data model.

Appendix A.1 Threat Model and Assumptions

In this framework, we designate the underlying operating system, auditing engine, and monitoring data as elements of the trusted computing base (TCB). The threat model described here is consistent with those utilized in comparable studies on system auditing [28, 46, 47]. We presume the system's benign nature, with the attack originating externally to the enterprise. The attacker utilizes remote network access to infiltrate systems following the cyber attack lifecycle [43]. All behaviors are assumed to be audited at the kernel layer, with activities logged as audit records, despite potential efforts by attackers to conceal their footprints while carrying out malicious actions. The primary focus of this work is on behaviors that occur within a single host. However, the principles we present can be readily applied to behaviors spanning multiple machines.

Appendix A.2 Data Model

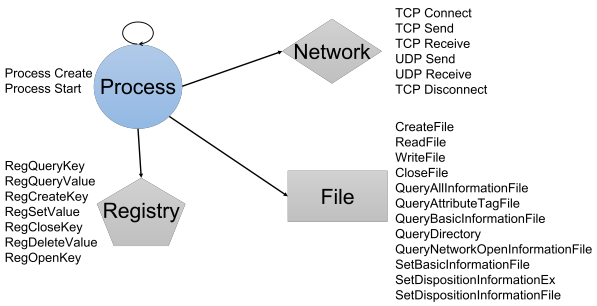

Figure 9: Data model of provenance graph

Appendix B

Threat Actor	Reference
Higaisa	New LNK attack tied to Higaisa APT discovered. [40]
admin338	China-based Cyber Threat Group Uses Dropbox for Malware Communications and Targets Hong Kong Media Outlets. [41]
APT28	CVE-2023-38831 Exploited by Pro-Russia Hacking Groups in RU-UA Conflict Zone for Credential Harvesting Operations. [10]
FIN7	FIN7 Evolution and the Phishing LNK. [42]
CobaltGroup	Cobalt Strikes Back: An Evolving Multinational Threat to Finance. [49]
Gamaredon	Gamaredon Infection: From Dropper to Entry. [13]
Patchwork	Unveiling Patchwork - The Copy-Paste APT. [17]
GorgonGroup	The Gorgon Group: Slithering Between Nation State and Cybercrime. [60]

Appendix C

Campaign	Cyber attack Lifecycle						
	Inital Compromise	**Establish Foothold**	**Escalate Privileges**	**Internal Reconnaissance**	**Move Laterally**	**Maintain Presence**	**Complete Mission**
APT28	- phishing Attachment	- Malicious File Execution - Web Protocols	- System Information Discovery - Data from Local System				- Exfiltration Over Web Service
Gamaredon	- phishing Attachment	- Malicious File Execution - Web Protocols	- Modify Registry - Registry Run Keys - WMI - System Information Discovery - Scheduled Task				- Internal Defacement
admin338	- phishing Attachment	- Malicious File Execution	- Local Account - File and Directory Discovery - Local Groups - System Network Configuration Discovery - System Network Connections Discovery - System Service Discovery				N/A
CobaltGroup	- phishing Attachment	- Remote Access Software	- Network Service Discovery				N/A
FIN7	- phishing Attachment	- Ingress Tool Transfer	- Registry Run Keys - Scheduled Task				N/A
GorgonGroup	- phishing Attachment	- PowerShell	- Portable Executable Injection - Startup Folder - Shortcut Modification - Disable or Modify Tools - Hidden Window				N/A
Higaisa	- phishing Attachment	- Malicious File Execution	- Registry Run Keys - System Information Discovery - System Network Configuration Discovery - Masquerade Task or Service - Scheduled Task/Job: Scheduled Task				N/A
Patchwork	- phishing Attachment	- PowerShell	- Bypass User Account Control - Data from Local System - System Owner/User Discovery - Security Software Discovery - Startup Folder - Remote Desktop Protocol				N/A

Campaign	Cyber attack Lifecycle						
	Inital Compromise	**Establish Foothold**	**Escalate Privileges**	**Internal Reconnaissance**	**Move Laterally**	**Maintain Presence**	**Complete Mission**
G-1	- phishing Attachment	- Ingress Tool Transfer	- Security Account Manager - Default Accounts - Security Software Discovery - Local Groups - Dynamic-link Library Injection - Winlogon Helper DLL				- Inhibit System Recovery
G-2	- phishing Attachment	- Ingress Tool Transfer	- LSASS Memory - Browser Bookmark Discovery - Process Discovery - Office Application Startup - Registry Run Keys				- Resource Hijacking
G-3	- phishing Attachment	- Malicious File Execution	- Network Sniffing - Security Account Manager - File and Directory Discovery - Process Discovery - Disable Windows Event Logging				- Inhibit System Recovery
G-4	- phishing Attachment	- Ingress Tool Transfer	- Security Account Manager - LSASS Memory - Local Account - WMI - Startup Folder - Shortcut Modification				- Inhibit System Recovery
G-5	- phishing Attachment	- Ingress Tool Transfer	- Default Accounts - Security Account Manager - Video Capture - PowerShell - Registry Run Keys - Disable or Modify System Firewall				- Inhibit System Recovery
G-6	- phishing Attachment	- PowerShell	- Security Account Manager - NTDS - System Owner/User Discovery - Registry Run Keys				- Resource Hijacking
G-7	- phishing Attachment	- Web Protocols	- Bypass User Access Control - Masquerade Task or Service - Registry Run Keys				- Inhibit System Recovery
G-8	- phishing Attachment	- WMI	- LSASS Memory - Default Accounts - Password Policy Discovery - Browser Bookmark Discovery - Disable Windows Event Logging - Office Test				- Inhibit System Recovery
G-9	- phishing Attachment	- Ingress Tool Transfer	- NTDS - Security Account Manager - WMI - PowerShell				- Inhibit System Recovery
G-10	- phishing Attachment	- Ingress Tool Transfer	- Bypass User Access Control - Network Sniffing - Software Discovery - System Owner/User Discovery - Modify Registry				- Inhibit System Recovery

Campaign	Cyber attack Lifecycle						
	Inital Compromise	**Establish Foothold**	**Escalate Privileges**	**Internal Reconnaissance**	**Move Laterally**	**Maintain Presence**	**Complete Mission**
G-11	- phishing Attachment	- WMI	- Default Accounts - Network Sniffing - System Network Configuration Discovery - Disable Windows Event Logging - Modify Registry				- Defacement
G-12	- phishing Attachment	- PowerShell	- NTDS - Browser Bookmark Discovery - System Network Configuration Discovery - Modify Registry - Rename System Utilities				- Defacement
G-13	- phishing Attachment	- PowerShell	- System Network Configuration Discovery - System Network Connections Discovery - Portable Executable Injection - Rename System Utilities				- Inhibit System Recovery
G-14	- phishing Attachment	- Ingress Tool Transfer	- LSASS Memory				- Inhibit System Recovery
G-15	- phishing Attachment	- Ingress Tool Transfer	- Local Account - Modify Registry				- Inhibit System Recovery
G-16	- phishing Attachment	- Ingress Tool Transfer	- OS Credential Dumping - Security Account Manager - System Network Configuration Discovery - File and Directory Discovery - Modify Registry - DLL Search Order Hijacking				- Inhibit System Recovery
G-17	- phishing Attachment	- PowerShell	- OS Credential Dumping - LSASS Memory - Audio Capture				- Resource Hijacking
G-18	- phishing Attachment	- Ingress Tool Transfer	- Bypass User Access Control - Process Discovery - Domain Trust Discovery - PowerShell - Modify Registry				- Inhibit System Recovery
G-19	- phishing Attachment	- PowerShell	- OS Credential Dumping - Bypass User Access Control - PowerShell - Local Account				- Inhibit System Recovery
G-20	- phishing Attachment	- Ingress Tool Transfer	- NTDS - Credentials in Registry - Network Share Discovery - Peripheral Device Discovery - Disable or Modify System Firewall				- Endpoint Denial of Service