electronics

Article

ConGraph: Advanced Persistent Threat Detection Method Based on Provenance Graph Combined with Process Context in Cyber-Physical System Environment

Linrui Li and Wen Chen *

School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China; 2021226245054@stu.scu.edu.cn
* Correspondence: wenchen@scu.edu.cn

Abstract: With the wide use of Cyber-Physical Systems (CPS) in many applications, targets of advanced persistent threats (APTs) have been extended to the IoT and industrial control systems. Provenance graph analysis based on system audit logs has become a promising way for APT detection and investigation. However, we cannot afford to ignore that existing provenance-based APT detection systems lack the process–context information at system runtime, which seriously limits detection performance. In this paper, we proposed ConGraph, an approach for detecting APT attacks using provenance graphs combined with process context; we presented a module for collecting process context to detect APT attacks. This module collects file access behavior, network access behavior, and interactive relationship features of processes to enrich semantic information of the provenance graph. It was the first time that the provenance graph was combined with multiple process–context information to improve the detection performance of APT attacks. ConGraph extracts process activity features from the provenance graphs and submits the features to a CNN-BiLSTM model to detect underlying APT activities. Compared to some state-of-the-art models, our model raised the average precision rate, recall rate, and F-1 score by 13.12%, 25.61%, and 24.28%, respectively.

Keywords: APT; CPS; attack detection; process context

Citation: Li, L.; Chen, W. ConGraph: Advanced Persistent Threat Detection Method Based on Provenance Graph Combined with Process Context in Cyber-Physical System Environment. *Electronics* **2024**, *13*, 945. https://doi.org/10.3390/electronics13050945

Academic Editor: Cheonshik Kim

Received: 31 January 2024
Revised: 27 February 2024
Accepted: 28 February 2024
Published: 29 February 2024

1. Introduction

Cyber-Physical Systems (CPS) integrate sensing, computing, control, and networking technologies to support monitoring and feedback loops of cyber systems and continuously improve the performance of physical processes [1]. In recent years, CPS technologies have been applied in a variety of scenarios such as smart homes, power grids, the Internet of Vehicles, and healthcare. However, such systems, like traditional network information systems, have become the targets of advanced persistent threats (APTs) [2–5]. According to ENISA [6] and NIST [7] guidelines, a recent trend in the IoT domain concerns the certification of security features of IoT devices [8,9]. Therefore, it is important to study APT attack-detection methods in CPS environments.

APT attacks are long-term and complex cyberattacks carried out by highly skilled and organized attackers. Government agencies, large corporations, research institutes, and other important organizations are typically the targets of such attacks. APT attackers meticulously design attack steps and customize attack methods to steal sensitive data or disrupt systems while ensuring stealth. APT attacks can pose a significant security threat to various types of CPS systems. For instance, in 2010, the 'Stuxnet' worm attacked Iranian nuclear facilities [2]. In 2015, the Ukrainian power grid was attacked by the 'BlackEnergy' malware [3]. In 2016, the 'Mirai' malware compromised a large number of IoT devices and used them to launch large-scale distributed denial-of-service (DDoS) attacks, disrupting numerous mainstream sites [4]. In 2018, the 'VPNFilter' malware affected 500,000 IoT devices in at least 54 countries and regions, creating a massive botnet [5].

APT attacks are often difficult to detect due to their long term and stealth. Attackers can remain undetected in the system for a long period of time and utilize a series of attack techniques to damage the user systems [10]. Traditional intrusion detection techniques are typically only able to detect single-point network intrusions, making it challenging to detect the complex attack patterns of APT [11,12]. Existing methods for detecting APT attacks include rule-based, anomaly-based, and provenance-based detection. While rule-based detection can be efficient, it has limitations. For example, it is weak in detecting unknown threats. Fixed rules can be bypassed when an adversary uses 0-day vulnerabilities for an attack, so the method relies on a long-maintained rule database. Secondly, designing rules requires the expertise of professionals in the fields of threat modeling, operating systems, and computer networks, which can be a labor-intensive process. Anomaly-based detection techniques alert processes within the system when their behavior deviates from the normal pattern by learning the behavioral patterns within the system under normal operating conditions [13]. This approach is capable of detecting unknown attacks such as 0-day attacks, but due to the long-term character of APT attacks, the model may be subject to poisoning attacks while learning normal behavior within the system, and the detection results tend to have a high false alarm rate. Additionally, system evolution can make it challenging to characterize normal behavior, which may degrade detection performance [14]. Provenance-based detection methods utilize provenance data, such as operating system audit logs, to detect APT attacks. To improve detection performance, researchers extract key information from system logs and audit records and convert it into a directed acyclic graph. Nodes in the graph represent system entities (e.g., processes, files, and network connections) and edges represent causal relationships between them. A simplified APT attack process is depicted in Figure 1, where the attacker gains access to the user terminal by exploiting Firefox vulnerabilities and delivers the payload. Subsequently, 'cmd' process executes the malicious commands sent by the attacker and the 'svchost' process performs information gathering and sends it to the attack host. The provenance graph provides a clear understanding of the interaction between system entities and can connect causally related events, even if they are separated by a long period of time [15].

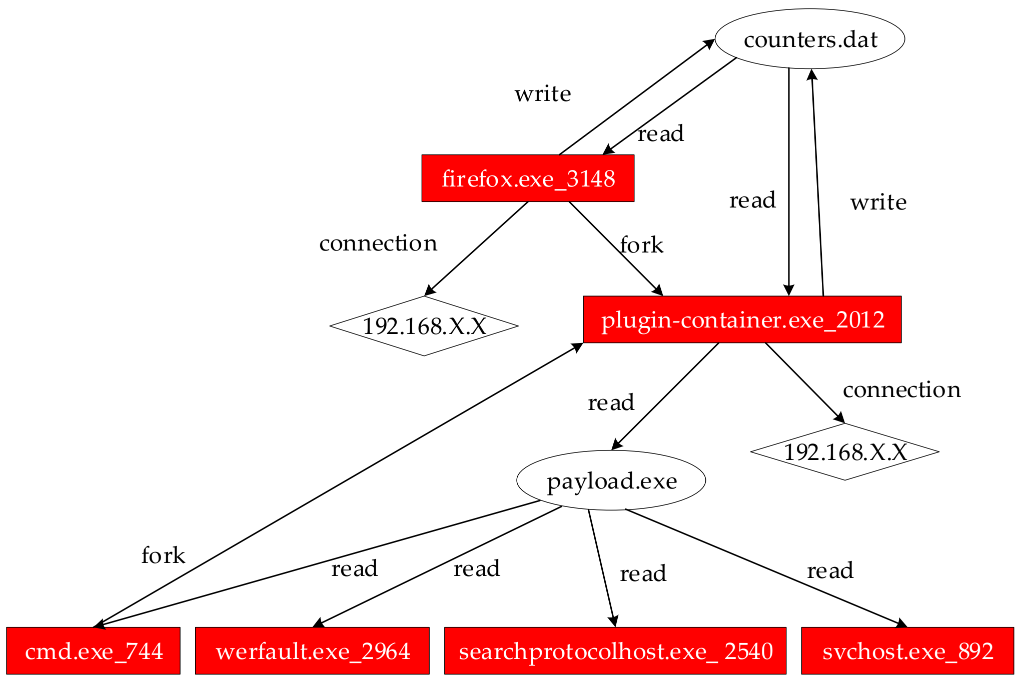


Figure 1. Example of provenance graph.

However, existing provenance-based APT detection systems face several challenges: (a) The collection of provenance data for the system mainly focuses on audit logs and lacks critical context information, such as the interactions between processes and the security characteristics of the files in the runtime of the system. This limitation leads to a high false positive in detecting APT attacks; (b) The large size of the provenance graph makes it difficult to extract the sequences characterizing process behavior from the graph. In this paper, we propose ConGraph, an APT attack detection method based on a provenance graph and process context in a CPS environment. To construct the provenance graph, ConGraph collects provenance data, such as system audit logs. Furthermore, it also collects process-behavioral records and file features related to process operations to form the process–context information. The provenance graph is compressed using a graph-compression algorithm and then fused with process–context information. Next, feature sequences representing the node behaviors are extracted from the fused provenance graph. Finally, the sequence-based model CNN-BiLSTM is used to detect APT attacks.

The main contributions of this paper are summarized below:

(1) A new APT attack detection method ConGraph is proposed. It incorporates rich process–context information and file features at system runtime during the construction of a provenance graph to model fine-grained process behavior patterns.
(2) We present a module for collecting contextual information to detect APT attacks, which collects features such as the file-access behavior of processes, network-access behavior, and interactive relationships between processes to enrich the provenance-graph structure. The detection efficiency is improved by using a graph-compression algorithm to reduce the computational scale.
(3) We reproduced experiments based on a public dataset to collect process context during system runtime and evaluated our method. The experimental results demonstrate the effectiveness of our approach in detecting APT activities.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Section 3 describes the threat model and related definitions in the paper. Section 4 introduces the architecture of ConGraph in detail. The experiments and results of ConGraph are described in Section 5. Section 6 summarizes our work and outlines future work.

2. Related Work

Provenance-based methodologies have emerged as a promising area of research in detecting APT attacks. These methods aim to extract key information from audit logs to distinguish APT attack behaviors from normal system activities and common system errors. The methods employed in provenance-based APT attack detection are broadly classified into three types: rule-based, statistical-based, and learning-based [16].

Rule-based approaches utilize a priori knowledge such as attack patterns of known vulnerabilities to construct heuristic rules to detect APT attacks. SLEUTH [17] pioneered the reconstruction of APT attacks using provenance graphs. It categorizes system audit logs into four levels of confidentiality: public, private, sensitive, and secret. Labels and policies are introduced to assign different weights to the dependency graphs and identify system entities and events that are most likely involved in an APT attack. Poirot [18] suggested combining threat intelligence with provenance graphs to detect APT attacks. This involves collecting external IoC relationship information, extracting and constructing a query graph of attack behaviors, matching and aligning the attack graph with the system provenance graph through a graph-matching algorithm, and ultimately generating alerts and forensic analysis results. Holmes [19] proposes a hierarchical framework for detecting APT attacks. The key component of this framework is an intermediate layer called HSG. HSG aligns low-level APT attacks with provenance graphs based on rules derived from domain knowledge (e.g., ATT&CK framework). These rules map low-level audit data to attack intent, bridging the semantic gap between low-level audit data and attack intent. CONAN [20] presents a state-based framework that abstracts each system entity, such as a process or a file, into a Finite State Automata (FSA)-like structure. The state is inferred through predefined rules, and the state sequences are used for APT detection. Rule-based detection methods are efficient and easy to deploy. However, designing effective rules relies heavily on in-depth domain knowledge, and rules can easily fail if conditions change. Therefore, rule-based methods have major limitations when facing unknown APT attacks.

Statistical-based approaches are utilized to detect anomalies in the provenance graph by constructing statistical features of the system's behavior. PrioTracker [21] proposes a causal tracker that prioritizes anomalous dependencies in the tracking process. To distinguish anomalous system events, PrioTracker builds a reference model of daily activities in the system as a way to quantify the scarcity of each event. After that, the priority score of each event is calculated based on its scarcity and topological features in the provenance graph, and finally, the information paths with high priority scores are searched. Nodoze [22] generates a causal graph of alert events and assigns an anomaly score to each edge in the dependency graph based on the frequency of occurrence of the relevant events. However, since statistical analysis considers direct (causal) links in the provenance graph and lacks the expression of semantic relationships between system entities, this leads to a large number of false positives in the detection.

To overcome the limitations of the aforementioned approaches, researchers have begun developing APT detection models using learning-based methods. Unicorn [15] proposed constructing histograms and computing graph sketches using provenance graphs of normal behaviors within the system to detect APT attacks in the absence of prior knowledge of the attack. However, due to the limitations of graph–kernel methods, they have difficulty detecting stealth threats. THREATRACE [23] utilizes GraphSAGE to learn the structural information of different types of entities within the system in the provenance graph. It identifies anomalous nodes within the system through semi-supervised learning and multi-modeling frameworks to detect and track APT attacks. SHADEWATCHER [16] maps the concept of system-entity interactions to user-item interactions in recommendations. It detects cyber threats by predicting system entities' preferences for their interacting entities and utilizes graph neural networks to improve detection.

Furthermore, log-based anomaly detection has shown advancements in detecting APT attacks. Log2Vec [24] transforms log entries into a heterogeneous graph and applies heterogeneous graph embedding techniques to represent each log entry as a low-dimensional vector. This method distinguishes between malicious and benign log entries based on their respective vectors. DeepLog [25] is a deep-learning-based log-anomaly detection method that models system logs as natural language sequences and uses long short-term memory (LSTM) neural networks to learn normal log patterns. DeepAG [26] utilizes the Transformer model to map log sequences into vector form, thus reducing the loss of semantic information. It proposes a bi-directional model that includes forward and backward LSTMs for the detection of abnormal logs.

3. Preliminary

3.1. Threat Model

In this paper, our goal is to detect anomalous entities in a host caused by intrusion activities. We assume that the adversary comes from outside the system and its goal is to steal valuable information inside the system. The attacker has the following characteristics.

(1) Persistence. The attack activity lasts for a long time.
(2) Stealthy. The attacker will try to mix the malicious activity with a lot of normal activity and try to disguise the malicious activity as normal activity.
(3) An attack pattern exists in the provenance graph. In order to accomplish a malicious activity that is different from the benign activity, the attacker's behavior should have some attack patterns in the provenance graph. The malicious node has a different local structure or contextual information compared to benign nodes in the provenance graph.
(4) C&C communication. The victim host needs to communicate with the attacker to complete the attack commands and steal high-value information from the victim host.

For example, as shown in Figure 2, we provide a typical APT attack scenario which will be used as an example in this paper. In a wireless sensor network (WSN), numerous sensor nodes gather data from a monitoring area and form a self-organized network. The data collected by multiple sensor nodes are transmitted and processed through an intermediate node host to reach the sink node. Then, the sink node transmits the data to the control server through the network. An attacker creates a malicious Office file from the Internet and sends it to the target host via a phishing email. Personnel at the control center download the email and open the malicious Office file without checking its security. The malicious file exploits an Office vulnerability (e.g., CVE-2017-11882 [27]) to gain privileges on the target system. The vulnerability was exploited by the attacker to send an executable program to the target host. This program was used to perform a port scan and gather information from the target network. The attacker then established a silent connection with the host and moved laterally to attack the servers within the control center. Figure 2 depicts the topology of the threat model. The control center comprises the wireless sensor network control server and the corresponding database. It is responsible for controlling the sensor nodes in the WSN and distributing relevant information to the Internet. The control center operates in an intranet environment and communicates with the Internet through edge machines and with the wireless sensor network through sink nodes.

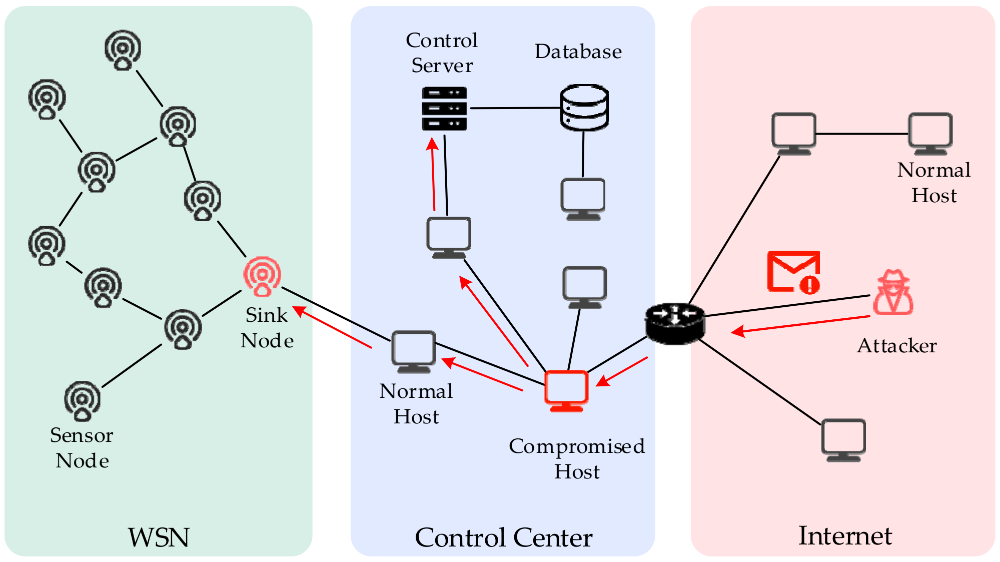


Figure 2. Threat model.

Furthermore, it is assumed that ConGraph itself is not under attack, and the provenance data has not been tampered with by attackers to obfuscate the model.

3.2. Definitions

Definition 1. *Active node. Active nodes in the provenance graph are those with active behaviors or are related to critical behaviors. There are two categories of active nodes: first, nodes with non-zero out-degree, indicating active interactions with other nodes; and second, nodes that serve as the destination nodes of critical behaviors, which may play a key role in the attacker's intrusion and attack propagation process.*

Definition 2. *Critical behaviors. Critical behaviors are a set of edge types in the provenance graph that represent specific system activities. Examples of critical behaviors include file writes, file encryption, and processes connecting to the network. Nodes associated with critical behaviors usually involve file modifications or network access and may become crucial links in an attack path or malicious activity.*

Definition 3. *Action subgraph. The action subgraph is a subgraph related to the active node v_i extracted from the provenance graph that fused with the process context. It can be denoted as $\hat{G}_i = (\hat{V}^i_{active}, E^i_{active}, \hat{C}_i)$, where $\hat{V}^i_{active}$ denotes the nodes that have interaction with the active node v_i, E^i_{active} denotes the action set of v_i (Section 4.4), and $\hat{C}_i$ denotes the collection of the process context of all the nodes within $\hat{V}^i_{active}$. The action subgraph contains the interactions of the active node v_i with other system entities. ConGraph constructs the action sequence of a node by parsing the action subgraph of the active node.*

Definition 4. *Action sequence. The action sequence describes in textual form the interactive behavior of the process within the system, as well as contextual information about the relevant system entities. For process v_i, its action sequence seq^i can be represented as a set of ordered sequences $< seq^i_1, seq^i_2, seq^i_3, \cdots, seq^i_j >$, and each subsequence seq^i_j can be represented as a quintuple $< v_m, context(v_m), action(e), v_n, context(v_n) >$, where v_m and v_n denote the system entities associated with process v_i; e denotes the edge connecting v_m and v_n; $action(\cdot)$ denotes the behavior represented by the edge; and $context(\cdot)$ denotes the process context (Section 4.3).*

4. Proposed Model of APT Detection

4.1. System Overview of ConGraph

The architecture of ConGraph is illustrated in Figure 3, it consists of four components: (1) Log Preprocessing, (2) Process–Context Collector, (3) Sequence Construction and Context Fusion, and (4) Model Training and Detection.

(1) Log Preprocessing. The module involves the systematic collection and analysis of system audit logs, browser history, and DNS traffic through an array of tools—system audit tools (Windows ETW, Linux Auditd, etc.), user browsers (Google Chrome, Firefox, etc.), and network analysis tools (Wireshark, etc.). These data are subsequently structured into a provenance graph, arranged chronologically. In this graph, nodes represent system entities, like processes and files, while edges correspond to system events, exemplified by process creation.

(2) Process–Context Collector. ConGraph collects API calls of processes at system runtime through an API call logging tool to obtain interaction characteristics, network characteristics, and file access characteristics of the processes. This enriches the semantic information of the nodes in the provenance graph. Furthermore, considering the complexity of matching certain process behavioral characteristics through API calls, we utilize rule matching to gather and collect these process behavioral features and other file security features, such as the file sensitivity level.

(3) Sequence Construction and Context Fusion. The module will be divided into five parts: ⓐ identify active nodes, ⓑ context fusion, ⓒ sequence construction, ⓓ sequence sampling, and ⓔ sequence embedding. First, ConGraph constructs the active node set by identifying the nodes with active behavior from the provenance graph. Next, the process context is fused with the provenance graph generated in Step 1. This is conducted using the node information from the active node set and the process-context information gathered by the process–context collector. The process–context information includes the behavioral characteristics of the process actions and the accessed files' features. ConGraph extracts the action sequence of the active node from the fused provenance graph and completes the anonymized representation of the information. In addition, due to the great imbalance between attack sequences and benign sequences, over-sampling and under-sampling processes are performed on attack and benign sequences, respectively, during the training phase. Finally, we perform word embedding on the generated sequences to convert the anonymous representations into numerical representations. This improves the model's ability to semantically distinguish between attack sequences and benign sequences.

(4) Model Training and Detection. Through the above module, we collect the action sequences of attacking nodes and benign nodes. In the model training phase, we use

the CNN-BiLSTM model to learn the behavioral patterns of APT attack processes and their corresponding process–context features. BiLSTM is capable of depicting the time-based behavioral features of the active node and learning the correlations between other system entities of the attack node. To capture the implicit features of APT attacks in sequences, we introduce a CNN layer to the model. During the detection phase, the contextual information is combined with the nodes' feature sequences to classify process behavior into normal access or attacks.

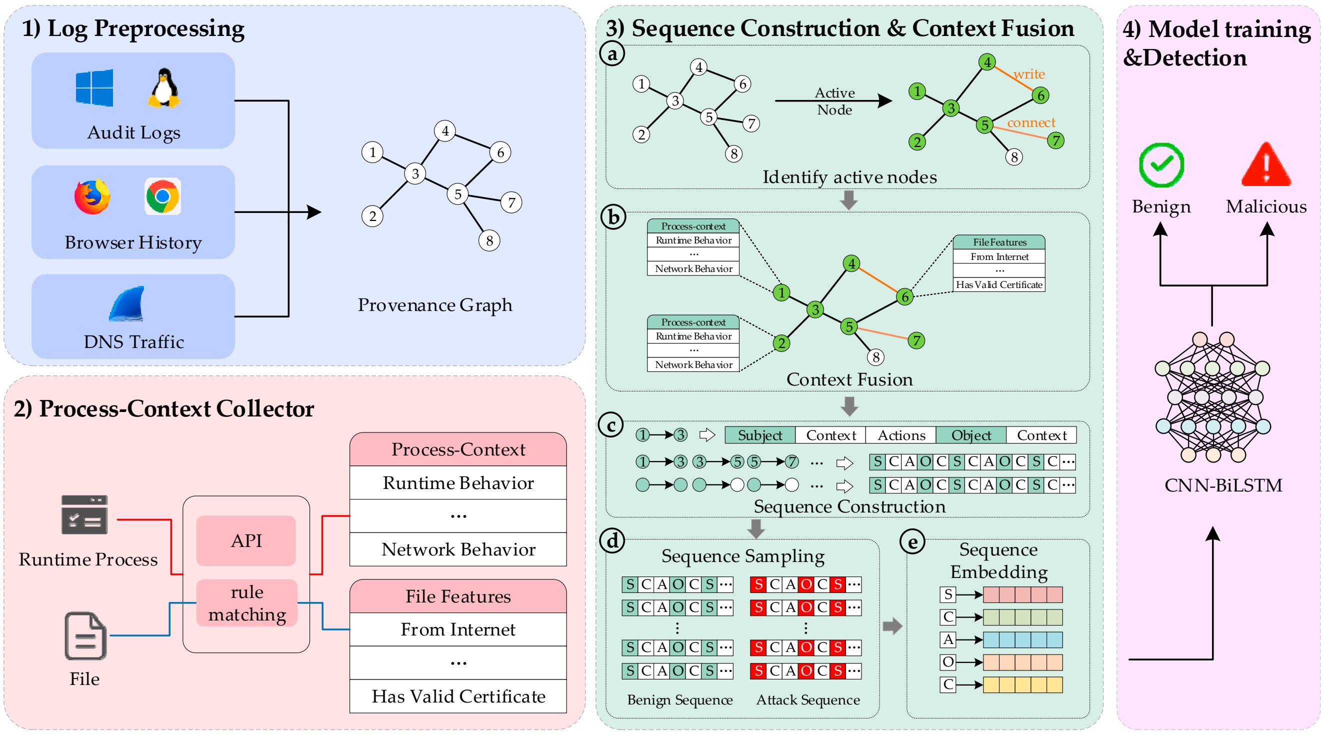


Figure 3. Overall architecture of ConGraph.

4.2. Log Preprocessing

ConGraph utilizes system audit tools (Windows ETW or Linux Auditd), user browsers (Google Chrome or Mozilla Firefox), and a traffic analysis tool (Wireshark) to collect system audit logs, user browsing history, and DNS resolving logs, respectively, to build a provenance graph. In the graph, nodes represent system entities such as processes and files, while edges represent system events like writing to files and receiving data from IP addresses. The provenance graph contains rich causal relationships, such as derived relationships between processes, files, and network accesses, which helps to model the relationships of system entities with long time distance and adapts to the problem that APT attacks are difficult to detect due to their long duration [28].

In addition, in order to reduce the storage and computational overhead of the system. ConGraph further performs the following optimizations on the constructed provenance graph to remove redundant and attack-irrelevant nodes and edges without losing the attack information.

(1) Merge all edges between two nodes that have the same category (e.g., file reads or writes) and retain only the edges with the earliest timestamps. Duplicate edges between nodes indicate repeated operations of the nodes over a period of time. Studies [17,29] have shown that these duplicate edges do not provide additional valid information for analyzing attacks. The same preprocessing will be used in the model training and detection phases, ensuring that the method does not affect the identification of the behavior of the attacking entities.

(2) If some nodes and edges are involved in the same type of events, such nodes are grouped together. For example, if there is communication between process P_1 and three network nodes N_1, N_2, and N_3 over a period of time, there will exist a set of edges $\{P_1 \to N_1,\ P_1 \to N_2,\ P_1 \to N_3\}$. At this point, we combine the three edges between N_1, N_2, and N_3 to form $\{P_1 \to N_{1-2-3}\}$, where N_{1-2-3} stands for the fused node.

(4) Remove all isolated points from the provenance graph. APT attacks usually require a set of processes and files to cooperate with each other in order to realize the corresponding attack purpose; therefore, we consider isolated nodes irrelevant to the attack.

4.3. Process–Context Collector

ConGraph further collects process and file node semantic information in the provenance graph by collecting process context at system runtime. The process context includes process API call sequences, network behavior, and features of accessed files. The process context is enumerated in Table 1.

Table 1. Process context collected at system runtime.

Categories		No.	Description
Process Feature	Runtime Behavior	1	Keylogger
		2	Recording microphone
		3	Grab screen
		4	Execute sensitive commands
		5	Access sensitive files
		6	the process has no GUI
	Network Behavior	7	The ancestor process has network connections
		8	The process accesses the Internet
		9	Download file from the Internet
File Feature		10	The file is downloaded from the Internet
		11	The file does not contain a valid signature
		12	The file is a system-sensitive file

Specifically, for process behaviors, such as screen recording and keyboard logging, an API call collection tool is utilized (e.g., API Monitor in Windows) to capture API call sequences from various processes during their operation. This enables a detailed recording of each process's behavioral patterns. By identifying the API calls linked to these specific behaviors, profiles of API sequences corresponding to these behaviors can be constructed. Subsequently, by matching these API call sequences with the pre-established API sequence features during system runtime, we are able to obtain the process context.

In addition, to further analyze other behaviors of processes, including whether the process has a graphical user interface (GUI), whether the process accesses sensitive files and other process runtime characteristics, as well as file characteristics, ConGraph scans the system's runtime processes and uses the Windows SDK and a rule-based approach to make judgments in order to determine whether the process has a user interface and other specific features. In addition, it is noted that whether processes try to access sensitive files can be discovered by checking the relevant attributes and paths of files.

In the CPS environment, ConGraph gathers node information from connected devices. For instance, in a WSN, the sink node collects and processes data from sensor nodes before transmitting it to the management computer. ConGraph collects statistical information, such as sensor node resource occupancy, the number of packets received and sent by nodes, node packet loss rate, and node energy consumption, in the management computer. These features will serve as the process context for the WSN control process. They are used to identify anomalies in WSN and detect potential attack activity in the system.

According to the process context shown in Table 1, the context of the process node v_p and the file node v_f are defined in (1), (2):

$$\text{context}\left(v_p\right) =< Pid, Timestamp, RuntimeBehavior, NetworkBehavior > \tag{1}$$

$$\text{context}\left(v_f\right) =< FilePath, Timestamp, FileFeature > \tag{2}$$

In order to facilitate the fusion of context information and sequence construction, ConGraph represents the process–context information and file characteristics as a text sequence. For example, the process context context(P) of process P (Pid = 3698) at the moment of t_1 is denoted as $< 3698, t_1, grab\_screen, access\_Internet >$, which indicates that the runtime behavior of process P at the moment of t_1 consists of grabbing the screen and the network behavior consists of accessing the Internet.

The process–context collector module enables a more comprehensive depiction of the behavioral patterns of the processes in the provenance graph. This includes their associations with file nodes and details of network behavior. The resulting information enhances ConGraph's interpretability for APT attack detection and enables more effective analysis and detection of such attacks.

4.4. Sequence Construction and Context Fusion

After obtaining the provenance graph and the corresponding process–context information, they are fused to construct the corresponding sequence of provenance graphs for model training and detection.

Identify Active Nodes. ConGraph identifies all nodes from the provenance graph that are associated with critical behaviors and have active behaviors. For the provenance graph $G = (V, E)$, where V denotes the set of nodes in the provenance graph, and it is associated with three types, namely processes node, file node, and network node, and E denotes the set of edges in the provenance graph along with multiple edge types (e.g., process creation, destruction, and file read, write operations, etc.), which are used to represent the interactions of nodes within the system. The active node set $V_{active} = \{V_{out} \cup V_{WC}\}$ is constructed by traversing the provenance graph G. This involves identifying all nodes from G whose out-degree is bigger than 0.

$$V_{out} = \{v_i \,|v_i \in V, \text{out} - \text{degree}(v_i) > 0\} \tag{3}$$

and whose edge types are destination nodes in the set of critical behaviors ACT.

$$V_{WC} = \{v_{dst} \mid v_{dst} \in V, \exists \text{e}(v_{src}, v_{dst}) \in E, \text{type}(\text{e}(v_{src}, v_{dst})) \in ACT\} \tag{4}$$

Here, $ACT = \{write, connect, encrypt, \cdots\}$ denotes the set of critical behaviors; $\text{e}(v_{src}, v_{dst})$ denotes the edge between the source node v_{src} and the destination node v_{dst}, and type$(\cdot)$ represents the type of the edge. Nodes in V_{WC} typically represent files or network nodes that are actively modified or accessed by processes; such files or network nodes are usually accessed by other nodes and may become critical links in the attack path.

Next, ConGraph constructs the corresponding action set E_{active} for the nodes in V_{active}. For a given node $v_i \in V_{active}$, its action set can be represented as

$$E^i_{active} = \{\text{e}(v_{src}, v_{dst}) | i \in \{src, dst\}, \text{e}(v_{src}, v_{dst}) \in E, v_{src} \in V_{active}, v_{dst} \in V_{active}\} \tag{5}$$

where src and dst indicate that the node belongs to the source and destination nodes, respectively, and $\text{e}(v_{src}, v_{dst})$ denotes the edge between v_{src} and v_{dst}.

Context Fusion. After identifying the active nodes in the provenance graph, the collected process context is merged with the provenance-graph nodes. Based on the timestamp of the collected process context, the pid of the process, and the path of the file are matched with the set of active nodes V_{active} in the provenance graph, the provenance graph fused with the process context $G_C = (V_{active}, E_{active}, C)$ can be obtained, where C denotes the set of process contexts of the active nodes.

The process of matching nodes from V_{active} with the process context defined in Section 4.3 involves sequentially taking out nodes and matching them based on their type in the provenance graph. For a node v_p of process type, the attributes in the process context $\text{context}(v_p)$ (Equation (1)) are matched with the pid attribute of the process node, yielding the set of contexts of the process node v_p

$$C_p = \{\text{context}(v_p) | \text{pid}(v_p) = \text{pid}(\text{context}(v_p))\} \tag{6}$$

where $\text{pid}(\cdot)$ denotes the pid obtained from the process node or the process context. It is possible for a process node to contain multiple process contexts with different timestamps. The one that has the closest time stamp will be selected to construct the activity sequence of the node.

Similarly, for the file node v_f, the context set of the file node v_f is obtained by matching the path of the file with the path in the file node context information $\text{context}\left(v_f\right)$ (Equation (2))

$$C_f = \left\{\text{context}\left(v_f\right) \middle| \text{path}\left(v_f\right) = \text{path}\left(\text{context}\left(v_f\right)\right)\right\} \tag{7}$$

where path($\cdot$) denotes the path to the file obtained from the file node or context information.

Sequence Construction. In the provenance graph G_C that is fused with process context, the action subgraph $\hat{G}_i$ corresponding to the active node v_i is extracted from the provenance graph fused with the process context G_C in terms of node v_i in the active node set V_{active}.

$$\hat{G}_i = \left(\hat{V}^i_{active}, E^i_{active}, \hat{C}_i\right) \tag{8}$$

where

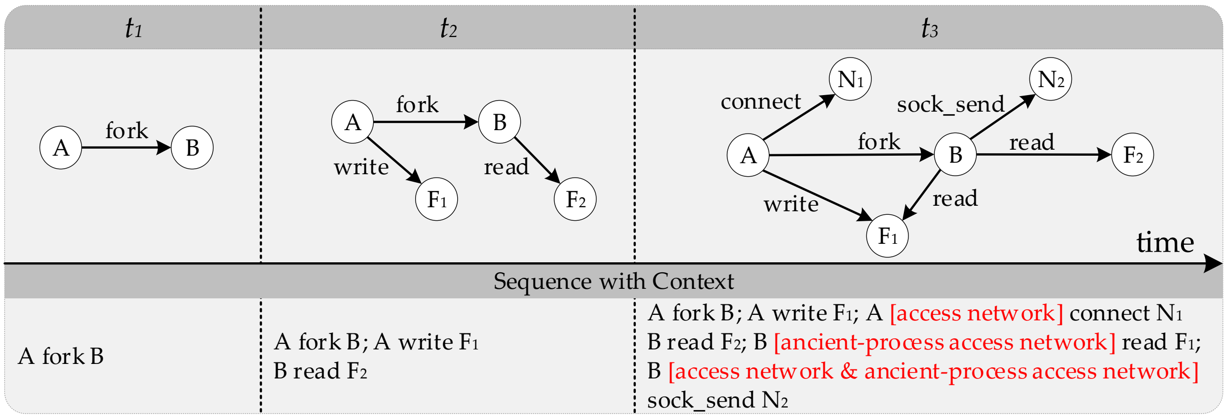
$$\begin{cases} \hat{V}^i_{active} = \{v_j | e(v_i, v_j) \text{ or } e(v_j, v_i) \in E^i_{active}\} \\ \hat{C}_i \subseteq C \end{cases} \tag{9}$$

$\hat{V}^i_{active}$ denotes the set of all nodes connected by active node v_i through edges within E^i_{active}, and $\hat{C}_i$ denotes the set of context information of all nodes within $\hat{V}^i_{active}$. With the action subgraph $\hat{G}_i$ corresponding to the active node v_i, we convert the relevant behavior of v_i within the system into its corresponding action sequence $seq^i =< seq^i_1, seq^i_2, seq^i_3, \cdots, seq^i_j >$. Specifically, for the activity subgraph $\hat{G}_i$ corresponding to node v_i, after sorting the edges in E^i_{active} into $\left\{e^i_1, e^i_2, e^i_3, \cdots, e^i_j\right\}$ according to the timestamps, ConGraph takes out the edge e^i_j and the corresponding node pair $< v^i_m, v^i_n >$ from E^i_{active}, and take out the corresponding process context of the node pairs from $\hat{C}_i$, and construct the action sequence of the edge e^i_j and the corresponding node according to Equation (10).

$$seq^i_j =< v^i_m, \text{context}\left(v^i_m\right), \text{action}\left(e^i_j\right), v^i_n, \text{context}\left(v^i_n\right) > \tag{10}$$

where $\text{context}(\cdot)$ denotes the context information of the process, and $\text{action}(\cdot)$ denotes the behavior represented by the edge.

It is worth noting that the construction of the action sequence is time-dependent. As shown in Figure 4, at the moment of t_1, process A forks process B; at the moment of t_2, process A writes the file F_1 and process B reads the file F_2, and process A does not access the network before that; at the moment of t_3, process A accesses the network, and process B reads the file F_1 and sends it over the network. Hence, the context of process B before t_3 will not record the network behavior of the "ancestor-process access network". The time-based sequence construction matches the nodes with the corresponding process context according to the occurrence time of the event to accurately characterize the runtime behavior of the process, thus improving the ability of the model to capture the association between the attack behavior and the process context.



A fork B

A fork B; A write F_1; B read F_2

A fork B; A write F_1; A [access network] connect N_1; B read F_2; B [ancient-process access network] read F_1; B [access network & ancient-process access network] sock_send N_2

Figure 4. Time-based sequence construction.

After constructing the sequence, it is necessary to anonymize the node and edge representations to retain the original semantics of the complete sequence and adapt the sequence-based model learning. The specific conversion types are shown in Table 2. Taking file nodes as an example, they will be mapped from low-level semantics to high-level semantics of file nodes according to the node names and paths. Specifically, the file type is divided into three categories: system_file for system-operation-related files, program_file for program directory files, and user_file for documents created or downloaded by the user.

Table 2. Anonymous representation of nodes and edges.

Type		Representation
Node	File	system_file, program_file, user_file
	Process	system_process, program_process, user_process
	Network	socket, web_object, domain, IP_address
Edges		bind, sock_send, write, delete, fork, resolve, web_request, refer, connect, read, executed

Sequence Sampling. Since APT attacks are stealthy, the attacker will conceal its behavior in a huge number of system operations, so the amount of attack data and normal data is unbalanced. Training models on such datasets may result in a model that tends to learn the features of normal sequences and has a weak recognition ability and low recall in detecting attack sequences. Therefore, undersample benign samples and oversample attack samples can be carried out to alleviate the sample imbalance problem. For a large number of benign samples, we calculate sequence similarity using the Levenshtein distance [30]. If the similarity between the sequence in the sample and other sequences in that sample exceeds a certain threshold, we remove it from the sample. For a smaller number of attack sequences, we oversampled the sequences using a mutation-based mechanism [31] to include more attack sequences that were not triggered by the current scenario. The oversampling mechanism based on mutation modifies the anonymous representation of corresponding positions of the sequence to other representations of the same type in order to cover similar sequences that may appear in other attacks. This preserves the original sequence behavior and improves the model's generalization ability. Specifically, for an edge sequence seq^i_j (shown in Equation (10)) in the action sequence seq^i, we replace it with other anonymous representations under the same type based on the type of node v^i_m and v^i_n, as well as edge behavior action $\left(e^i_j\right)$ after abstraction.

Sequence Embedding. Finally, the sequence is embedded to convert the anonymous representation into a numerical one, which ensures that the model can better distinguish differences between samples during the training process, thus improving the accurate identification of potential attack behaviors. For the active sequence seq^i of active node v_i, which contains the node's contextual information and anonymous behavioral representation, we use the method described in [31] to map this textualized action sequence into a vector representation.

4.5. Model Training and Detection

The sequences collected above contain all the activities of the system entities and their contextual behavioral information, both normal and attack. To learn the activity patterns of system entities over time, we use a CNN-BiLSTM model to model the process behavior and learn APT attack behavioral features. This model combines the advantages of a convolutional neural network (CNN) for extracting spatial features and the capability of a bidirectional long short-term memory network (BiLSTM) for capturing long-term dependencies to learn the interaction behaviors among system entities. The model architecture is shown in Figure 5.

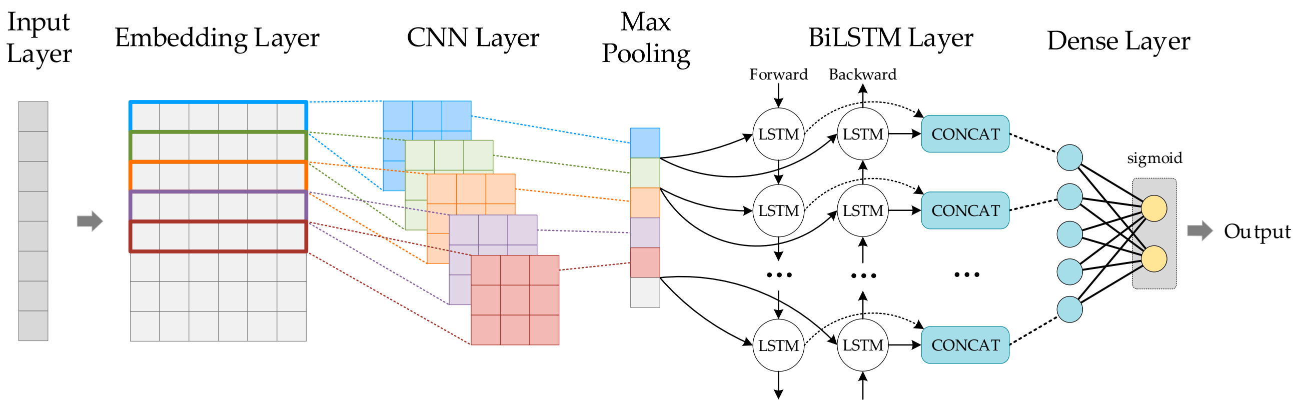


Figure 5. The architecture of training and detection model.

The input layer receives the vector representation of action sequences as input. To avoid the problem of gradient vanishing, the input layer requires fixed-length sequence data. Therefore, we truncate the sequences that exceed the set maximum length. Additionally, the sequences with insufficient length are filled with zero. The input process action sequences are transformed into dense embedding vectors by the embedding layer to effectively capture the semantic relevance of process behaviors. A one-dimensional convolutional layer (CNN) is used in conjunction with a MaxPooling layer to extract behavioral patterns of processes in the action sequence and capture implicit features. This helps to learn the correlation between the behaviors of the APT attacks in the sequences and the process context. The bidirectional LSTM layer (BiLSTM) captures long-term associations of system entities at runtime in long sequences to learn the interactions between attacking nodes and other relevant system entities. The dense layer outputs the model's predictions for process activity sequences through the sigmoid activation function.

In order to learn the features of APT attacks effectively, ConGraph uses binary cross entropy (BCE) as the loss function of the model, and BCE is defined as follows:

$$\mathrm{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i)\cdot \log(1-p(y_i)) \tag{11}$$

where N denotes the number of samples, y_i denotes the true label (0 or 1) of the ith sample; and $p(y_i)$ denotes the probability that the model predicts the ith sample to be a positive class. The binary cross entropy is adapted to the binary classification detection scenario of APT attack detection and can provide an effective gradient to prevent the problem of gradient vanishing during the training process, which improves the performance of the model to learn APT attack behaviors.

5. Experiment

5.1. Dataset and Experimental Setup

During the experiment, we construct similar APT attack environments based on the dataset provided by ATLAS [31], since the current publicly available dataset lacks process context at system runtime. The dataset contains system audit logs of APTs in six multi-host environments and many simulated user activities (e.g., browsing websites and reading emails) in a Windows environment. Table 3 details the characteristics of the APTs in the dataset.

Table 3. Overview of the ATLAS dataset.

ID	APT Campaign	Exploit CVE	Attack Features *							Numberof Entity	
			PL	PA	INJ	IG	BD	LM	DE	Attack	Non-Attack
M-1	Strategic web compromise [32]	2015-5122	✓		✓	✓	✓	✓	✓	28	17,565
M-2	Targeted GOV phishing [33]	2015-5199	✓		✓	✓	✓	✓	✓	36	24,450
M-3	Malvertising dominate [34]	2015-3105	✓		✓	✓	✓	✓	✓	36	24,424
M-4	Monero miner by Rig [35]	2018-8174		✓	✓	✓	✓	✓	✓	28	15,378
M-5	Pony campaign [36]	2017-0199		✓	✓	✓	✓	✓	✓	30	35,671
M-6	Spam campaign [27]	2017-11882		✓	✓	✓	✓	✓	✓	42	19,580

* PL: Phishing link; PA: Phishing attachment; INJ: Injection; IG: Information gathering; BD: Backdoor; LM: Lateral movement; DE: Data exfiltration.

The dataset comprises 4.4 GB of audit log data from 6 attack simulations generating 20,000 unique entities. Each attack scenario contains 24 h of audit logs and simulates a variety of normal user activities. Among them, M-1 to M-6 represent APT attack scenarios targeting six APTs across multiple hosts. The dataset includes multiple APT attack phases, such as phishing email links, phishing email attachments, injections, and lateral movement. In the provenance graph, the nodes associated with the attacks also account for less than 3% of the nodes in the provenance graph, which is similar to real-world scenarios.

To collect process context, we replicate the operational environment of this experiment by simulating normal user behavior and executing attacks over a period of time, constructing a dataset containing context information. We use this dataset to detect APT activities in one attack scenario by training models on other attack scenarios. This allows us to evaluate the effectiveness of ConGraph for APT attack detection in each scenario. For instance, if the objective is to identify attacks in scenario M-1, a model will be trained using audit logs from five other attack samples, excluding M-1. The performance of the trained model in detecting attacks will be tested using the audit logs of attack scenario M-1 to ensure that the training and testing data do not overlap. The model will be trained for each attack scenario for detection, and the sequences associated with the nodes will be randomly divided into a training set (75%) and a validation set (25%).

5.2. Comparison Experiment

We compare the proposed detection methods with the following existing models: DeepLog [25], Deepro [37], and LogBert [38]. The comparison results are shown in Table 4, where the bold data are the best results for the corresponding scenarios. DeepLog utilizes an anomaly detection model of LSTM to model system logs as natural language sequences and learns logging patterns from normal execution. Deepro designs provenance graphs with three new meta-paths to extract causality in the provenance graph and uses a customized MAGNN model for APT attack detection. LogBert is a self-supervised framework for log anomaly detection based on Transformer's Bidirectional Encoder Representation (BERT).

Table 4. Results of comparison experiment.

Scenario	Methods	Metrics		
		Precision	Recall	F1-Score
M-1	DeepLog	0.8064	0.6854	0.6496
	Deepro	0.8947	1	0.9444
	LogBert	0.7512	0.5724	0.6497
	ConGraph	**0.9936**	**0.9935**	**0.9935**
M-2	DeepLog	0.7806	0.6104	0.5408
	Deepro	0.8824	0.9357	0.9091
	LogBert	0.707	0.4725	0.5664
	ConGraph	**0.9922**	**0.992**	**0.992**
M-3	DeepLog	0.7824	0.6158	0.5493
	Deepro	0.85	0.9444	0.8947
	LogBert	0.7329	0.4758	0.577
	ConGraph	**0.969**	**0.9684**	**0.9684**
M-4	DeepLog	0.7843	0.6214	0.5582
	Deepro	**1**	0.875	0.9333
	LogBert	0.7753	0.5774	0.6619
	ConGraph	0.9603	**0.9591**	**0.9591**
M-5	DeepLog	0.7878	0.6239	0.5758
	Deepro	**0.9529**	0.8929	0.9091
	LogBert	0.6996	0.4847	0.5727
	ConGraph	0.916	**0.9051**	**0.9095**
M-6	DeepLog	0.7863	0.6274	0.5674
	Deepro	**1**	0.7333	**0.8462**
	LogBert	0.7312	0.5518	0.629
	ConGraph	0.8592	**0.8184**	0.8131
Avg.	DeepLog	0.7879	0.6306	0.5735
	Deepro	0.9255	0.8968	0.9061
	LogBert	0.7329	0.5224	0.6094
	ConGraph	**0.9467**	**0.9394**	**0.9392**

By comparing the experiments, it can be concluded that ConGraph achieves a precision of 94.67% and a recall of 93.94% in six APT attack scenarios involving multiple hosts. In scenario M1, the attacker exploits CVE-2015-5122 [32] by tricking the user into clicking on a link that redirects to a malicious website. The attacker then uses this link to exploit the CVE-2015-5122 vulnerability, causing the Firefox plugin to break and allowing the attacker to write the payload program 'payload.exe' onto the compromised host. Then, the payload program scans the host for files, establishes a connection with the attacker, and uploads all PDF files. The low recall of DeepLog and LogBert in this scenario is attributed to their reliance on log templates defined in advance. These models only extract log keys from log entries and submit them into the detection model. Furthermore, Deepro utilizes a heterogeneous graph neural network to detect APT attack activities within the provenance graph. However, due to the absence of process–context information, Deepro may misidentify certain benign activities as attacks, resulting in a decrease in precision.

Figure 6 shows the precision of ConGraph in classifying benign and malicious sequences for scenarios M1 to M6. Upon analyzing the false positives that arose during the experiments, it was discovered that in the M-5 and M-6 attack scenarios, most of the false positives were caused by benign IP addresses. These normal IP communications are similar to the malicious IP addresses that perform command and control (C&C) behaviors, resulting in a degradation of the model's performance in classifying such malicious IP addresses. However, administrators can filter out false positives by examining traffic content and domain registration information to identify the corresponding IP addresses.

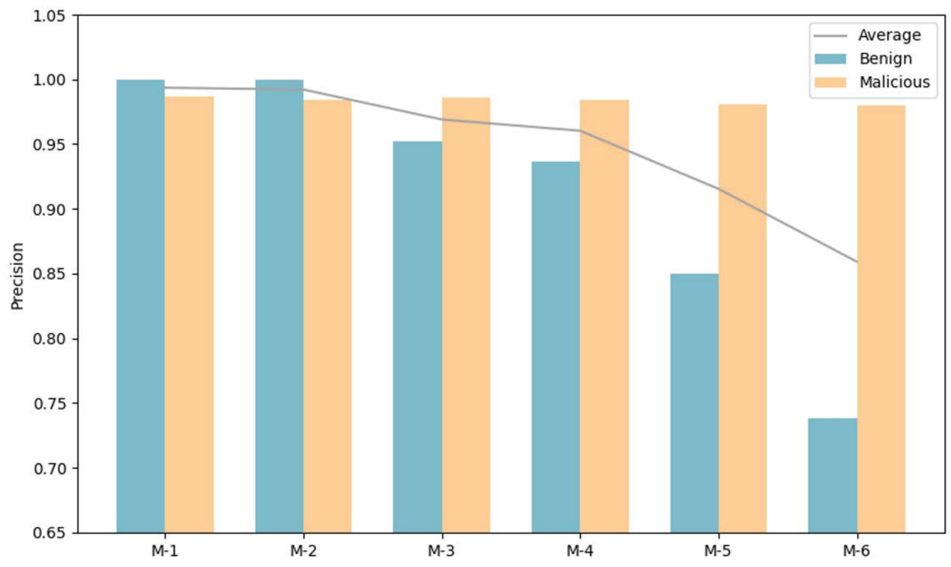


Figure 6. Precision of benign and malicious sequences in scenarios M1 to M6.

5.3. Ablation Experiment

In this section, we explore the effectiveness of the proposed process context for APT attack detection. To demonstrate that process–context information provides critical attack semantic information in sequence model classification, the collected data is re-embedded and the model is trained in a multi-host scenario after removing the process context.

Table 5 illustrates the results of our ablation experiments, and the bold data are the best results for the corresponding scenarios. The baseline in the table indicates that the process context has been removed from ConGraph. As can be summarized from the table, in the M-1 multi-host attack scenario, our method improves the precision rate by 5.13%, the recall rate by 6.74%, and the F-1 score by 6.74%. It implies that the introduction of process context provides important contextual information to the sequence model, which enables it to classify APT attack entities more accurately.

Table 5. Results of ablation experiment.

Scenarios	Metrics	Precision	Recall	F1-Score
M-1	baseline	0.9323	0.9261	0.9261
	+process context	**0.9836**	**0.9935**	**0.9935**
M-2	baseline	0.9892	0.9889	0.9889
	+process context	**0.9922**	**0.992**	**0.992**
M-3	baseline	0.9466	0.9432	0.9431
	+process context	**0.969**	**0.9684**	**0.9684**
M-4	baseline	0.9347	0.9282	0.9279
	+process context	**0.9603**	**0.9591**	**0.9591**
M-5	baseline	0.8941	0.874	0.8724
	+process context	**0.9155**	**0.9051**	**0.9045**
M-6	baseline	0.8415	0.7835	0.7739
	+process context	**0.8592**	**0.8184**	**0.8131**

The M-6 scenario involves exploiting the CVE-2017-11882 vulnerability to launch an attack on the target machine. The victim host downloads an email with a malicious payload, which allows the attacker to gain privileges on the target host. The attacker then delivers other payloads and steals files from the target system to gain higher control privileges and maintain persistent control over the target. Upon the ablation experiment results, it was discovered that the M-4 scenario, lacking support from the process context, resulted in high false negatives. The reason for this is that the attacker accessed sensitive files on the target system when stealing files. In this case, the attack sequence without process–context information appears similar to the behavior sequence of a normal process accessing files, making it difficult for the model to differentiate between normal and attack sequences.

6. Conclusions

In this paper, we proposed ConGraph, an APT detection method based on a provenance graph combined with process context in CPS environments. ConGraph collects provenance data from the system to construct the provenance graph. Additionally, we present a module for collecting process context to detect APT attacks. The module collects data on file-access behavior, network-access behavior, and interactive relationships between processes to enhance the provenance-graph structure. This allows for a more granular portrayal of process behavioral patterns. After reducing the provenance graph, the active nodes are extracted, the process context is fused with the reduced graph, and the textualized action sequences of the active nodes are extracted. Then, we detect APT attacks using the sequence-based model after anonymizing and embedding the sequence. By simulating APT attacks to collect process–context information, we constructed a simulated attack dataset. Based on this dataset, we conducted a comparative analysis of ConGraph with other machine learning (DeepLog) and deep learning (LogBert, Deepro) based APT detection methods. In 6 experimental scenarios, ConGraph demonstrated an average improvement of 13.12%, 25.61%, and 24.28% in precision, recall, and F1 score, respectively, compared to the aforementioned methods.

However, introducing a process–context information collector in a CPS environment may add overhead to the system. Conversely, the absence of process context may degrade the model's classification performance for unknown threats. For instance, to identify partial process runtime and file features through non-API calls, it is essential to obtain a list of runtime processes in the system and evaluate them individually. The process context that requires individual evaluation consumes system resources for detection purposes. In the CPS environment, computational resources may be limited, and researchers need to dynamically add or delete process context based on the actual system resource situation to balance system performance and detection accuracy. In future work, researchers can improve the process–context screening method by considering the following aspects: Analyzing APT malware-related attack patterns to identify behavioral characteristics of malware and using genetic algorithms to find combinations of process context that distinguish between attack and benign behaviors.

Author Contributions: Conceptualization, W.C.; methodology, L.L.; software, L.L.; validation, L.L.; formal analysis, L.L.; investigation, L.L.; resources, W.C.; data curation, L.L.; writing—original draft preparation, L.L.; writing—review and editing, W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program of China (020YFB1805405, 2019QY0800) and the National Natural Science Foundation of China (62372313, 62002248, U19A2068).

Data Availability Statement: The datasets used in this paper are publicly available.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Han, S.; Xie, M.; Chen, H.H.; Ling, Y. Intrusion detection in cyber-physical systems: Techniques and challenges. *IEEE Syst. J.* **2014**, *8*, 1052–1062.
2. Langner, R. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* **2011**, *9*, 49–51. [CrossRef]
3. Kumar, R.; Kela, R.; Singh, S.; Trujillo-Rasua, R. APT attacks on industrial control systems: A tale of three incidents. *Int. J. Crit. Infrastruct. Prot.* **2022**, *37*, 100521. [CrossRef]

4. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Invernizzi, L.; Kallitsis, M.; et al. Understanding the mirai botnet. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 1093–1110.
5. Sicato, J.C.S.; Sharma, P.K.; Loia, V.; Park, J.H. VPNFilter malware analysis on cyber threat in smart home network. *Appl. Sci.* **2019**, *9*, 2763. [CrossRef]
6. European Union Agency for Cybersecurity (ENISA). Baseline Security Recommendations for IoT. Available online: https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot/@@download/fullReport (accessed on 8 October 2023).
7. NIST. Cybersecurity for IoT Program. Available online: https://www.nist.gov/itl/applied-cybersecurity/nist-cybersecurity-iot-program/consumer-iot-cybersecurity (accessed on 8 October 2023).
8. Cirne, A.; Sousa, P.R.; Resende, J.S.; Antunes, L. IoT security certifications: Challenges and potential approaches. *Comput. Secur.* **2022**, *116*, 102669. [CrossRef]
9. Anselmi, G.; Mandalari, A.M.; Lazzaro, S.; De Angelis, V. COPSEC: Compliance-Oriented IoT Security and Privacy Evaluation Framework. In Proceedings of the 29th Annual International Conference on Mobile Computing and Networking, Madrid, Spain, 2–6 October 2023; pp. 1–3.
10. Alshamrani, A.; Myneni, S.; Chowdhary, A.; Huang, D. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1851–1877. [CrossRef]
11. Bridges, R.A.; Glass-Vanderlan, T.R.; Iannacone, M.D.; Vincent, M.S.; Chen, Q. A survey of intrusion detection systems leveraging host data. *ACM Comput. Surv. CSUR* **2019**, *52*, 1–35. [CrossRef]
12. Singla, A.; Bertino, E.; Verma, D. Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation. In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, 5–9 October 2020; pp. 127–140.
13. Axelsson, S. *Intrusion Detection Systems: A Survey and Taxonomy*; Chalmers University of Technology: Goteborg, Sweden, 2000.
14. Han, X.; Pasquier, T.; Ranjan, T.; Goldstein, M.; Seltzer, M. {FRAPpuccino}: Fault-detection through Runtime Analysis of Provenance. In Proceedings of the 9th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 17), Santa Clara, CA, USA, 10–11 July 2017.
15. Han, X.; Pasquier, T.; Bates, A.; Mickens, J.; Seltzer, M. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv* **2020**, arXiv:2001.01525.
16. Zengy, J.; Wang, X.; Liu, J.; Chen, Y.; Liang, Z.; Chua, T.S.; Chua, Z.L. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22–26 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 489–506.
17. Hossain, M.N.; Milajerdi, S.M.; Wang, J.; Eshete, B.; Gjomemo, R.; Sekar, R.; Stoller, S.; Venkatakrishnan, V. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 487–504.
18. Yang, J.; Zhang, Q.; Jiang, X.; Chen, S.; Yang, F. Poirot: Causal correlation aided semantic analysis for advanced persistent threat detection. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3546–3563. [CrossRef]
19. Milajerdi, S.M.; Gjomemo, R.; Eshete, B.; Sekar, R.; Venkatakrishnan, V. Holmes: Real-time apt detection through correlation of suspicious information flows. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1137–1152.
20. Xiong, C.; Zhu, T.; Dong, W.; Ruan, L.; Yang, R.; Cheng, Y.; Chen, Y.; Cheng, S.; Chen, X. CONAN: A practical real-time APT detection system with high accuracy and efficiency. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 551–565. [CrossRef]
21. Liu, Y.; Zhang, M.; Li, D.; Jee, K.; Li, Z.; Wu, Z.; Rhee, J.; Mittal, P. Towards a Timely Causality Analysis for Enterprise Security. In Proceedings of the Network and Distributed Systems Security (NDSS), San Diego, CA, USA, 18–21 February 2018.
22. Hassan, W.U.; Guo, S.; Li, D.; Chen, Z.; Jee, K.; Li, Z.; Bates, A. Nodoze: Combatting threat alert fatigue with automated provenance triage. In Proceedings of the Network and Distributed Systems Security Symposium, San Diego, CA, USA, 24–27 February . 2019.
23. Wang, S.; Wang, Z.; Zhou, T.; Sun, H.; Yin, X.; Han, D.; Zhang, H.; Shi, X.; Yang, J. Threatrace: Detecting and tracing host based threats in node level through provenance graph learning. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3972–3987. [CrossRef]
24. Liu, F.; Wen, Y.; Zhang, D.; Jiang, X.; Xing, X.; Meng, D. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 1777–1794.
25. Du, M.; Li, F.; Zheng, G.; Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1285–1298.
26. Li, T.; Jiang, Y.; Lin, C.; Obaidat, M.S.; Shen, Y.; Ma, J. Deepag: Attack graph construction and threats prediction with bi-directional deep learning. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 740–757. [CrossRef]
27. Ramos, C. Spam Campaigns with Malware Exploiting CVE-2017-11882 Spread in Australia and Japan. Available online: https://www.trendmicro.com/vinfo/us/threat-encyclopedia/spam/3655/spam-campaigns-with-malware-exploiting-cve201711882-spread-in-australia-and-japan (accessed on 6 June 2020).

28. Chen, T.; Dong, C.; Lv, M.; Song, Q.; Liu, H.; Zhu, T.; Xu, K.; Chen, L.; Ji, S.; Fan, Y. APT-KGL: An Intelligent APT Detection System Based on Threat Knowledge and Heterogeneous Provenance Graph Learning. *IEEE Trans. Dependable Secur. Comput.* **2022**, 1–15. [CrossRef]
29. Xu, Z.; Wu, Z.; Li, Z.; Jee, K.; Rhee, J.; Xiao, X.; Xu, F.; Wang, H.; Jiang, G. High fidelity data reduction for big data security dependency analyses. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 504–516.
30. Beijering, K.; Gooskens, C.; Heeringa, W. Predicting intelligibility and perceived linguistic distance by means of the Levenshtein algorithm. *Linguist. Neth.* **2008**, *25*, 13–24. [CrossRef]
31. Alsaheel, A.; Nan, Y.; Ma, S.; Yu, L.; Walkup, G.; Celik, Z.B.; Zhang, X.; Xu, D. {ATLAS}: A sequence-based learning approach for attack investigation. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Vancouver, BC, Canada, 11–13 August 2021; pp. 3005–3022.
32. FireEye Threat Intelligence. Second Adobe Flash Zeroday CVE-2015-5122 from Hackingteam Exploited in Strategic Web Compromise Targeting Japanese Victims. Available online: https://www.fireeye.com/blog/threat-research/2015/07/second_adobe_flashz0.html (accessed on 6 June 2020).
33. Paganini, P. Phishing Campaigns Target US Government Agencies Exploiting Hacking Team Flaw CVE-2015-5119. Available online: https://securityaffairs.co/wordpress/38707/cyber-crime/phishing-cve-2015-5119.html (accessed on 6 June 2020).
34. Li, B.; Chen, J.C. Exploit Kits in 2015: Flash Bugs, Compromised Sites, Malvertising Dominate. Available online: https://blog.trendmicro.com/trendlabs-security-intelligence/exploit-kits-2015-flash-bugs-compromised-sites-malvertising-dominate/ (accessed on 6 June 2020).
35. Trend Micro. Rig Exploit Kit Now Using CVE-2018-8174 to Deliver Monero Miner. Available online: https://blog.trendmicro.com/trendlabs-security-intelligence/rig-exploit-kit-now-using-cve-2018-8174-to-deliver-monerominer/ (accessed on 6 June 2020).
36. Jiang, G.; Mohandas, R.; Leathery, J.; Berry, A.; Galang, L. CVE-2017-0199: In the Wild Attacks Leveraging HTA Handler. Available online: https://www.fireeye.com/blog/threat-research/2017/04/cve-2017-0199-hta-handler.html (accessed on 6 June 2020).
37. Yan, N.; Wen, Y.; Chen, L.; Wu, Y.; Zhang, B.; Wang, Z.; Meng, D. Deepro: Provenance-based APT Campaigns Detection via GNN. In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 9–11 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 747–758.
38. Guo, H.; Yuan, S.; Wu, X. Logbert: Log anomaly detection via bert. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18-22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.