KnowHow: Automatically Applying High-Level CTI Knowledge for Interpretable and Accurate Provenance Analysis

Yuhan Meng†, Shaofei Li†, Jiaping Gui‡, Peng Jiang§, and Ding Li†*
†Key Laboratory of High-Confidence Software Technologies (MOE), School of Computer Science, Peking University
‡School of Computer Science, Shanghai Jiao Tong University, §Southeast University
†{mengyuhan, lishaofei, ding_li}@pku.edu.cn, ‡jgui@sjtu.edu.cn, §pengjiang@seu.edu.cn

***Abstract*—High-level natural language knowledge in Cyber Threat Intelligence (CTI) reports, such as the ATT&CK framework, is beneficial to counter Advanced Persistent Threat (APT) attacks. However, how to automatically apply the high-level knowledge in CTI reports in realistic attack detection systems, such as provenance analysis systems, is still an open problem. The challenge stems from the semantic gap between the knowledge and the low-level security logs: while the knowledge in CTI reports is written in natural language, attack detection systems can only process low-level system events like file accesses or network IP manipulations. Manual approaches can be labor-intensive and error-prone.**

In this paper, we propose KnowHow, a CTI-knowledge-driven online provenance analysis approach that can automatically apply high-level attack knowledge from CTI reports written in natural language to detect low-level system events. The core of KnowHow is a novel attack knowledge representation, General Indicator of Compromise (gIoC), that represents the subjects, objects, and actions of attacks. By lifting system identifiers, such as file paths, in system events to natural language terms, KnowHow can match system events to gIoCs and further match them to techniques described in natural language. Finally, based on the techniques matched to system events, KnowHow reasons about the temporal logic of attack steps and detects potential APT attacks in system events. Our evaluation shows that KnowHow can accurately detect all 16 APT campaigns in the open-source and industrial datasets, while existing approaches all introduce large numbers of false positives. Meanwhile, our evaluation also shows that KnowHow reduces at most 90% of node-level false positives while having a higher node-level recall and is robust against several unknown attacks and mimicry attacks.

I. Introduction

Advanced Persistent Threat (APT) attacks are serious attacks in the modern world. Recently, multiple APT attacks have caused massive losses [39], [77]. To counter APT attacks, researchers have proposed provenance analysis techniques that mine system events to detect APT attacks [6], [87], [88], [34], [4], [20], [35], [86]. However, most existing provenance analysis techniques are data-driven, which inevitably introduce many false positives and make the detection result hard to interpret [87], [86], [34], [21]. For instance, when an attacker executes a "Word" document containing a macro virus [40], data-driven approaches may mistakenly report other Word processes as malicious and cannot explain why each "msword" process is considered malicious, which can be confusing for a security analyst encountering such malware for the first time. This makes it challenging for security personnel to process all alerts in a timely manner, potentially leaving some unaddressed, which ultimately may pose risks to the system [21], [34], [66].

Besides data-driven approaches, researchers also propose Cyber Threat Intelligence (CTI) knowledge driven approaches [54], [89], [53], [43], [44]. These approaches analyze CTI reports (e.g., MITRE ATT&CK framework [55], APT organization reports [48], [26] and research blogs [23], [70]), extract knowledge from these documents, and map the extracted knowledge to system events to detect attacks. Existing CTI-knowledge-driven approaches fall into two categories: Indicator of Compromise (IoC) based approaches [72], [28], [43], [36], [69] and high-level knowledge-based approaches [89], [44], [2]. The former one extracts low-level IoCs, such as concrete attack-related file names and IP addresses, and detects whether system events contain the reported IoCs. The latter one leverages human experts to curate detection rules or log patterns to detect attacks in system events.

Unfortunately, existing CTI-knowledge-driven approaches are limited due to the lack of extensibility. IoC-based approaches ignore general information in CTI documents but only focus on specific IoCs, which quickly become outdated [41] and cannot detect the varients of known attacks. High-level knowledge-based approaches rely on manual rules, which are expensive to build and imprecise. Holistically, both types of approaches can hardly be extended to keep up with the rapidly evolving attack techniques.

The key challenge of mapping the knowledge contained in high-level CTI reports to low-level provenance data is the semantic gap between natural language descriptions and system events. For instance, the CTI description "Lazagne carefully scanned the browser's resource directory to exten-

* is the corresponding author.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.230199
www.ndss-symposium.org

sively steal the credential files of users" should correlate with system events like "la1 read ../firefox/resource..", but the lack of linguistic similarity makes direct matching difficult. IoC-based approaches cannot handle this case since the description contains no IoC. For high-level knowledge-based approaches, building a mapping rule requires significant efforts from advanced human experts.

Our insight to make CTI-knowledge-driven approaches more extensible is that the mapping from high-level CTI knowledge can be largely automated by focusing on core elements in CTI reports. Specifically, we focus on three core elements in CTI reports: *attack conductors*, *attack actions*, and *attack targets*. These three core elements can be precisely mapped to key elements in system events: *processes*, *system calls*, and *operation files*. Therefore, by leveraging minimal lifting rules, system events can be automatically mapped to the three core elements in CTI reports using NLP techniques.

Based on our insight, we propose KNOWHOW, a CTI-knowledge-driven provenance analysis approach that automatically applies high-level attack knowledge from CTI reports to detect APT attacks in low-level system events. The core of our approach is CTI Knowledge Database (CKD) that features a new intermediate representation, the General Indicator of Compromise (gIoC), to represent the three core components in CTI reports. To effectively match gIoCs and system events, we propose a novel approach that lifts the semantics of system events to the natural language level (e.g., lifting "firefox" to "browser"), thereby bridging the gap between them. Besides, we introduce a novel attack reasoning approach that automatically identifies APT attacks in system events based on CKD. Our attack reasoning approach aligns attack techniques with attack tactics and APT Lifecycle stages, thereby reducing false positives by eliminating attack steps that do not adhere to the temporal logic of APT attacks [54], [15], [32], [19], [50], [79].

We conduct a comprehensive evaluation of KNOWHOW using the most recent and widely adopted open-source datasets, along with an industrial dataset. KNOWHOW precisely identifies all attack campaigns within our datasets, whereas all existing baselines exhibit much more false positives. Specifically, KNOWHOW reduces false positives at the node level by 81% and 90% for the two recent baselines, NODLINK [42] and KAIROS [13], respectively. Regarding the effect of the extraction on APT detection, our evaluation confirms that knowledge extracted using KNOWHOW can help reduce false positives in downstream APT attack detection tasks by 68% and 79%, respectively, compared to utilizing knowledge from LADDER and EXTRACTOR —two of the latest CTI information extraction frameworks. Our evaluation further shows that KNOWHOW's novel CTI knowledge condensation representation, gIoC, is the primary reason for its superiority over existing approaches. Moreover, it has been deployed in open-world scenarios, with the experimental results collectively validating the practical usability of KNOWHOW.

We summarize our contributions as follows:

- We propose KNOWHOW, a CTI-knowledge-driven online provenance analysis solution that can automatically apply high-level attack knowledge from CTI reports to detect APT attacks in low-level system events.
- We introduce a novel compact representation, gIoC, for high-level attack knowledge and manage it within a novel knowledge base, CKD.
- We design a novel attack reasoning method based on the stages of the APT Lifecycle identified by querying CKD.
- We thoroughly evaluate KNOWHOW on widely used datasets, as well as an industrial dataset. The results confirm that KNOWHOW meets the requirements for accurate, efficient, and interpretable detection of APT attacks.
- We have deployed KNOWHOW within the OpenEuler ecosystem, which is a production-grade, community-driven Linux distribution, to verify its validity in real-world security operations[1].

Open Science. We release the core code and datasets in https://github.com/myh0301/KNOWHOW to facilitate further research.

II. THREAT MODEL AND ASSUMPTIONS

We follow the same threat model used in previous works on system monitoring [25], [87], [88], [53], [54], [6]. Specifically, we assume the kernel and kernel-layer auditing frameworks [80], [51], [68] are not compromised. Any kernel-level attack campaign is outside the scope. We assume that there is an external attacker, who attacks the victim system remotely. Based on the above assumptions, the attacker can only achieve system intrusion by inducing the victim to download and execute a malicious payload or by exploiting a vulnerability.

III. BACKGROUND AND MOTIVATION

A CTI-knowledge-driven approach has two stages [72], [28], [36], [57], [2], [89]: (1) extracting knowledge from CTI reports and (2) mapping the extracted knowledge to low-level system events. In practice, the extracted knowledge may also be organized in standard formats, such as Structured Threat Information eXpression (STIX) [57]. Based on the types of extracted knowledge, we categorize existing CTI-knowledge-driven approaches into IoC-based approaches and high-level knowledge-based approaches. We show the differences between these two types of CTI knowledge in Figure 1.

A. IoC-Based Approaches

IoC-based approaches, such as EXTRACTOR [72], TTPDRILL [36], and THREATRAPTOR [28], extract concrete IoCs from CTI reports and match them against entities observed in system events. For instance, given the CTI text shown in Figure 1, these methods extract specific IoCs such as the malicious process name *Lazagne* and the suspicious file path *.../.config/google-chrome/User Data/Default/Login Data*. These IoCs are then used to match entities in system events for attack detection.

[1]We have extended KNOWHOW with an automated APT report generation module that transforms low-level alerts into structured, analyst-centric APT reports by using LLMs. This extended version is provided in http://arxiv.org/abs/2509.05698.

CTI Extraction for Detection		
CTI Text	Lazagne carefully scans the browser’s login directory, such as …/.config/google-Chrome/User Data/Default/Login Data, to extensively steal the credential files of system users.	
Corresponding System Event	E1 = <la1, **read**, …/firefox/resource/login_data…> E2 = <la1, **cp**, **/home/usr/**.config/firefox/xxx.default/key4.db>	
Method	***Extracted Knowledge***	***Detection Result and Explanation***
LADDER	• carefully scan the browser’s resource directory to extensively steal the credential files of the system users	<u>Cannot Be Used</u>: It is impossible to correlate complex natural language sentences with basic system events.
EXTRACTOR / TTPDrill / AttackKG /ThreatRaptor	• Lazagne • …/.config/google-Chrome/User Data/Default/Login Data	<u>Causes False Negative</u>: The specific IoCs do not align with the actual Firefox path and malicious process in system event E1. <u>Missing Knowledge</u>: It fails to acquire knowledge within the sentence “steal the credential files of system users”, which is necessary to match E2.
KnowHow	• gIoC.1: <Lazagne, scan, directory browser’s user login > • gIoC.2: <lazagne, steal, files credential system user >	<u>Full Knowledge</u>: KnowHow captures attack knowledge in both sentences of “..scan..” and “..steal..”. <u>Fuzzy Event Matching</u>: KnowHow matches “read” to “scan”, “steal” to “copy(cp)” and “browser’s login directory” and “credential files” to their paths.

Fig. 1: A comparison of various CTI-knowledge driven methods with the detection results using the extracted knowledge on the Lazagne case[3], [11].

IoC-based approaches suffer from high false-negative rates due to their reliance on static, time-sensitive indicators that quickly become obsolete and fail to generalize across environments (e.g., “Lazagne” vs “la1” in Figure 1), while also exhibiting knowledge omission by ignoring valuable behavioral descriptions in favor of concrete artifacts. For example, IoC only includes the Chrome browser path as Lazagne’s target, ignoring the description “scan the browser’s login directory,” thus failing to detect attacks on other browsers like Firefox in Figure 1.

B. High-Level Knowledge-Based Approaches

High-level knowledge-based approaches recognize that CTI reports often contain rich, descriptive content about attack behaviors and patterns beyond simple IoCs. These methods aim to extract such high-level knowledge from natural language CTI reports [2] and convert it into actionable detection rules through expert interpretation [89], [54]. An example of extracted high-level knowledge from one of these approaches, LADDER, is shown in Figure 1.

The key challenge is how to convert high-level knowledge into actionable rules. To this end, researchers have proposed many interesting methods that rely on expert-curated rules. Specifically, HOLMES [54] and APTSHIELD [89] perform pattern matching by checking whether observed system behaviors conform to their predefined attack lifecycle rules and sensitivity labels, while RAPSHEET [33] uses the manually curated rules to flag suspicious activities based on known attack indicators. POIROT [53] and ATTACKG [43] use expert-curated query graphs and attack technique templates derived from CTI reports, which are then applied to system events for detection. Likewise, TREC [44] performs graph-level feature matching between selected procedures from the Atomic Red Team [67] and system events to detect potential threats.

The limitation of existing high-level knowledge-based approaches is that they rely on static, concrete rules, which are often too strictly defined and specific to individual systems. This limitation hinders their ability to generalize to new systems and adapt to evolving attack patterns. For example, HOLMES, RAPSHEET, and APTSHIELD rely on manually defined whitelists, such as trusted IP addresses, benign command lists, sensitive files, and critical system files. These whitelists are often tailored to specific environments and require continuous updates to remain effective. Moreover, POIROT, ATTACKG, and TREC focus on concrete attack procedures extracted from CTI reports, which are often instance-specific and lack sufficient abstraction, limiting their ability to handle diverse and evolving attack variants.

C. Structured Threat Information eXpression

STIX provides a standardized way to represent both concrete IoCs and high-level attack descriptions (e.g., TTPs, Attack Patterns). STIX, by standardizing the representation of CTI knowledge, facilitates knowledge sharing across organizations. However, how to effectively use the full knowledge in STIX is still an open problem.

Moreover, although STIX supports representing high-level knowledge in its design, the community uaually uses STIX to share IoCs instead of high-level attack knowledge. A study shows that over 94.93% of STIX objects contain only IoC attributes, while less than 0.1% include high-level behavioral descriptions [38].

D. Motivation and Insight

Our key insight for an extensible high-level knowledge-based approach is that subject-verb-object phrases in CTI reports encapsulate richer attack knowledge that can be structured into more meaningful and extensible representations. These phrases decompose into three fundamental components: *attack conductors* (who), *attack actions* (what operation), and *attack targets* (on what), reflecting the core semantic structure of attack behaviors, exemplified by “la1 reads login data files” in Figure 1.

Unlike static and context-free IoCs, this structured representation preserves richer semantic information by including not only the core entities but also their relationships and contextual descriptions. As shown in Figure 1, *“steal the credential files of system user”* contains both the conductor (*lazagne*), the action (*steal*), and the target (*credential* *files of system user*), which provides a more comprehensive understanding of the attack behavior. The extracted knowledge can then be used to match with the system events `<la1, read, .../firefox/resource/login_data...>` in a proper way to detect the attack, which can address the limitation of existing high-level knowledge extraction methods. Compared to STIX, our representation contains more structured and actionable contextual information, which is essential for understanding the attack behaviors. Therefore, it can be used as a general structured knowledge representation that bridges the gap between high-level CTI knowledge and low-level system events, enabling more effective and automated attack detection.

IV. CTI Knowledge Database

To bridge the semantic gap between high-level knowledge in CTI reports and low-level system events, we propose CKD, a knowledge database dedicated to high-level attack knowledge derived from CTI reports. CKD is defined as a set of ATT&CK Technique Information Entries (ATIEs), each representing an attack technique as defined by MITRE ATT&CK [55]. An ATIE comprises four fields: a unique ID for a technique in ATT&CK (uid), a description of the technique (des), a CTI list ($list_{cti}$), and a gIoC list ($list_{gioc}$). Specifically, $list_{cti}$ contains CTI reports in which the technique can be identified. $list_{gioc}$ contains gIoCs, which encapsulate high-level attack knowledge reported in the corresponding CTI reports in $list_{cti}$ and will be elaborated on in the following subsections. To support high performance APT detection, CKD also provides a provenance query, $ProvQ$, that enables users to identify ATIEs related to a given system event.

A. General Indicators of Compromises

gIoC is the key component in CKD that connects high-level attack knowledge in CTI reports with low-level system events. Conceptually, a gIoC is a compact representation of attack descriptive sentences in CTI reports and can be automatically learned from CTI reports using Natural Language Processing (NLP) techniques. Formally, a gIoC is a Subject-Verb-Object (SVO) triplet structure, *(subject, verb, object)*, which captures the core information of ”who performs which operation on what” in attack descriptions. A typical gIoC is shown in Figure 1.

Specifically, the *subject* is the conductor of an attack. It can be the name of an attacker, malware, or a hijacked application that initiates attacks. For example, in the CTI sentence, “APT41 used built-in commands *net* to enumerate local administrator groups”[70], “APT41” is the attacker subject. In contrast, in another CTI sentence, “Keydnap adds the *setuid* flag to a binary to easily elevate in the future”[23], “Keydnap” is the malware subject. The $verb$ represents the action of attacks, such as “use” and “enumerate” in the APT41 example, and “add” and “elevate” in the Keydnap example. Finally, *object* is the target of an attack, such as “built-in commands *net*” and “local administrator groups” in the APT41 example, and “the *setuid* flag” and “a binary” in the Keydnap example.

The key difference between gIoCs and conventional SVO pairs is that the *subjects* and *objects* of gIoCs are attack-relevant concepts and their associated information, such as modifiers describing these concepts. On a high level, we define a noun as an attack-relevant concept if it constitutes an IoC or the name of a system object (e.g., an application name, domain names, file names, etc.). Our emphasis on attack-relevant concepts stems from our objective to correlate high-level CTI reports with low-level system events. These concepts provide clues about tools, IP addresses, and files that attackers might utilize during an attack, which can facilitate the matching of low-level events. By recognizing attack-relevant concepts, we can avoid overly broad statements like “the attack originated from state-owned groups,” which are less useful for automated low-level event matching. Note that attack-relevant nouns are more versatile than conventional IoCs, allowing our approach to match low-level events with corresponding gIoC even when the textual content does not match precisely.

Formally, we define a noun N as an attack-relevant concept if it fulfills one of the following five conditions:

1) N is an IoC, such as a file name, an IP address, a file hash, etc.
2) N is a domain name.
3) N is the name of an application or malware.
4) N is a command (e.g., cp) or its full name (e.g., copy)
5) N represents a general concept of system objects, including but not limited to terms like “file,” “directory,” “IP address,” “process,” “application,” “registry,” and their synonyms.

We propose conditions 2-5 to enhance the extensibility of gIoCs. For example, condition 5 allows our approach to capture expressions like “browser’s folder” in Figure 1, thereby facilitating the matching of events involving Firefox with attack knowledge pertinent to Chrome.

Given an attack-relevant concept, we also consider modifiers (e.g., “browser data”) and subclauses (e.g., “steal the credential files of system users”) that describe the concept and serve as its related information. This is due to the fact that nouns in CTI reports are often devoid of context and meaning. For example, in Figure 1, the identified attack-relevant concept in the sentence “..scans the browser’s resource directory.. ” is “directory”, which carries limited significance. Instead, we must identify the related information, such as “browser data” and “steal the credential files of system users” to align with low-level events involving access to Firefox’s user data and system credential files.

Compared with IoCs, the key advantage of gIoCs lies in their ability to reflect higher-level information. gIoCs extract attack information from a behavioral perspective, whereas IoCs focus on an instance perspective. Therefore, gIoCs can be better extended across different environments. For example, a typical type of IoC is the process name of malware appearing in system events. Consequently, IoC knowledge can only identify malware with the same process name in the event, ignoring other pertinent information. In contrast, gIoCs summarize attack behavioral information and match it across various parts of the event, such as the target file of the malware

TABLE I: Excerpts of the Extended IoC Parsing Rules.

<table>
<tr><th></th><th>Example</th><th>Regular Expression</th></tr>
<tr><td>Registry Run Key</td><td>HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon</td><td>^(HKCU|HKLM|HKCR|HKCC|HKEY_LOCAL_MACHINE|HKEY_CURRENT_USER|HKEY_CLASSES_ROOT|HKEY_CURRENT_CONFIG|HKEY_USERS)(\\[^\\?/*|<>:”]+)*$</td></tr>
<tr><td>File Name, File Path</td><td>C:\\Windows\Microsoft.NET \Framework64\, /root/.vscode-server/</td><td>^(?:(?:[a-zA-Z]:|\.{1,2})? [\\/](?:[^\\?/*|<>:”]+ [\\/])*)(?:(?:[^\\?/*|<>:”]+?)(?:\.[^.\\?/*|<>:”]+)?)?$</td></tr>
<tr><td>Typical Linux Command</td><td>scp, ssh, sftp, tftp, curl, sshd, certutil, wget, ls, rm, sh, mv, stat, cat, reg add, reg del, kill, pkill, grep, find, cat, ifconfig</td><td>Same as IoCParser</td></tr>
<tr><td>Typical Windows Command</td><td>Get-Process, Get-Service, Get-ChildItem, New-Item, Remove-Item, Set-Location, Clear-Host</td><td>Same as IoCParser</td></tr>
</table>

or executed commands. Hence, they are more effective in detecting different variants of the malware.

B. Building CKD from CTI Reports

We leverage NLP techniques to automatically identify concepts relevant to attacks. For IoCs, we use the IoCParser [63] to extract IoCs more effectively. However, we observe that the original IoCParser lacks adequate support for identifying file paths, registry run keys, and command line operations. Therefore, KNOWHOW extends the regular expressions in IoCParser to improve the identification of file paths and registry run keys. Meanwhile, KNOWHOW maintains a list of commonly used command lines to recognize command line entries in CTI. Table I illustrates the regular expressions and command line samples that we have extended. For conditions 2 and 3 mentioned in Section IV-A, we rely on large language models to identify domain and application names. Specifically, given a noun, we submit it to a large language model and inquire whether it is a domain name or an application/malware name. For condition 4, we maintain a list of common Linux and Windows system commands, along with their corresponding full names, based on their documentation. For condition 5, we maintain a list of general concepts of system objects and detect their synonyms using large language models. In our approach, we can leverage existing NLP toolkits, such as Stanford CoreNLP [78], to identify the modifiers and subclauses associated with an attack-relevant concept.

gIoCs Extraction. After identifying and complementing the attack-relevant concepts, our approach extracts gIoCs from CTI reports. It first extracts sentences that contain these concepts identified in the first step. Then, for each of these sentences, we employ SVO toolkits to extract *(subject, verb, object)* tuples, which serve as the backbones of gIoCs. For example, for the sentence shown in Figure 1, the extracted backbone is *(Lazagne, scan, directory)*.

Then, we enhance the semantics of the extracted backbone by appending the modifier text to its corresponding nouns. For example, as shown in Figure 1, the enriched gIoC would be *(Lazagne, scan, directory browser's user login)*. Here, we do not take into account the order or grammatical correctness of the text since these elements are adequate for modern NLP techniques to generate sentence embeddings that can fuzzily match low-level events, as will be discussed in Section IV-C. Specifically, we treat subclauses as regular sentences and process them in the same way as mentioned above. Therefore, the subclauses will also generate gIoCs.

There are two challenges in generating gIoC. The first challenge is the potential absence of certain components of *(subject, verb, object)* in the subclauses. For example, in the sentence depicted in Figure 1, the subclause “to steal the credential files of system users” lacks a subject. To address this, KNOWHOW directly borrows the missing component from the main clause to form complete *(subject, verb, object)* tuples. The second challenge is that many attack-relevant concepts are specific IoCs, such as concrete file names and IP addresses, which cannot be extended for broad attack detection. To address this issue, our approach adopts a rule-based method to extend common patterns of IoCs into gIoCs. Our high-level strategy involves extracting components from IoCs that carry semantic meaning. For example, for the file name “*/usr/bin/long_file_name/wallet.db*”, our approach converts it into a gIoC by using the first-level folder name as “the user folder” and its file name as “wallet db file”. This conversion allows a more extensible matching of files with similar names within BOTH Linux and Windows user folders. The excerpt of the conversion rules is outlined in Table II.

ATIEs Generation. The final step of our approach is converting the extracted gIoCs to ATIEs. This step needs first adding the ATT&CK technique IDs and descriptions to the corresponding fields within the ATIE.

For the ATT&CK technique IDs and descriptions, our approach first scans to determine whether the CTI report references MITRE ATT&CK technique descriptions [55]. Specifically, the content is extracted using the PyMuPDF [65], and regular expressions are employed to scan the technique IDs and descriptions. Note that numerous popular threat intelligence sources include ATT&CK technique labels, which are explicitly summarized in their CTI reports [47], [74], [46]. If the report explicitly cites a MITRE ATT&CK technique, our approach generates an ATIE for that technique.

If a CTI report does not explicitly reference an ATT&CK Technique, our approach attempts to associate it with an existing ATIE. To accomplish this, we have developed a function akin to $ProvQ$, which will be elaborated on in Section IV-C, to query the CKD for the most similar list of ATIEs. The

TABLE II: Excerpts of the simplification table to turn IoCs into gIoCs , where D means the name of this level dictionary, F means the name of the files, E means the extension name of the file, and Dom means the DNS domain name of the corresponding IP. The simplification of directories is similar to that of files, except that there is no extension name.

	System Identifier	Lifted Sentences
Linux File	/etc/D/*/$F.E$	etc D E file
	/var/D/*/$F.E$	var D E file
	/proc/[PID]/D/*/$F.E$	proc D E file
	/bin/D/*/$F.E$, /sbin/D/*/$F.E$, /usr/bin/D/*/$F.E$, /usr/sbin/D/*/$F.E$, /usr/local/bin/D/*/$F.E$, /usr/local/sbin/D/*/$F.E$	F E file
	/home/aa/$D/*/$F.E\$	user D F E file
	/root/D/*/$F.E$	root user D F E file
	/lib/D/*/$F.E$, /lib32/D/*/$F.E$, /lib64/D/*/$F.E$, /usr/local/lib/D/*/$F.E$, /xx/lib/D/*/$F.E$	D library file
	other: */$F.E$	E file
Windows File	HKEY_*, HKCU*, HKCR*, HKLM*, HKU*, HKCC*	registry run key
	c:\\windows\system32\$D$*\$F.E$	windows system D $F.E$ file
	c:\\windows\$D$*\$F.E$	windows system D $F.E$ file
	c:\\ProgramFiles\$D$*\$F.E$, c:\\ProgramFiles(x86))\$D$*\$F.E$	D F E file
	other: *\$F.E$	F E file
IP	10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16	internal network
	other	external network Dom
Command Operation	cp	copy
	scp, ssh, sftp, tftp, curl, sshd, certutil	transfer
	wget	download
	ls,dir	list
	rm,del,rmdir	remove
	sh	shell
	stat, cat	show
	schtask	schdule
	rundll	run, dll file
	reg add	add
	reg del	del
	kill, pkill, taskkill	stop
	grep, find	search
	cat	read
	powershell command with capital and lower-case letter (e.g., Invoke-Command, Get-ChildItem)	Divided by "-" and capital letters. (e.g., Invoke, Command; Get, Child Item)
System Call	execve	execute
	recvmsg, recvfrom	receive
	sendmsg, sendto	send
	chmod	change, file mode

sole difference is that, rather than utilizing a system event, we query the CKD using gIoCs derived from the CTI reports. We also employ a similar method to compute similarity scores, as outlined in Section IV-C.

C. Querying CKD with System Events

CKD provides the provenance query, $ProvQ$, that matches the low-level system events to ATIEs. The key advantage of $ProvQ$ is that it allows fuzzy matching to gIoCs, providing a more general detection even when IoCs fail to match system events.

$ProvQ$ takes a system event $e = (source, destination, syscalltype, commandline)$ and returns a list of ATIEs that match the given event. We say that e matches an ATIE, t, if their similarity score is greater than the given query threshold θ_q. The similarity score $Sim(e,t)$ between e and t is defined as $score(e,t) = \sum S(e.y,t)$, where y is one of "source", "destination", "syscalltype", and "commandline", and $S(e.y,t)$ is the occurrences of gIoCs in t that appear in $e.y$, respectively.

We calculate $S(e.y,t)$ by matching the semantics of $e.y$ to the subject-verb-object triplet of the gIoC. The high-level process is shown in Figure 2. The challenge is how to determine the occurrence of t in $e.y$ since the natural language words in gIoC cannot directly match the system identifiers, such as file paths, in system events. To address this problem, KNOWHOW first lifts the specific system events to extensible semantic representations and then uses semantic aware embedding to convert the system events and gIoCs into numeric vectors. Finally, our approach calculates the cosine similarity of the embedding vectors of the system events and gIoCs. If the cosine similarity is above a threshold, our approach considers the event "hits" a gIoC.

Event Semantic Lifting. The key step in constructing $ProvQ(e,\theta_q)$ involves elevating low-level system events to natural language sentences that can be matched with gIoCs. The high-level idea is the same as the conversion method mentioned in gIoC extraction in Section IV-B. For file paths, we extract the file type name, application name, and the necessary path information as the semantic representation. The file type name can be inferred from the file's extension, while the application name is derived from its installation location within specific folders (e.g. the name of the folder directly under "/bin" is typically the application name). For IP addresses, we utilize their domain name (when available) in reverse DNS as the semantic representation. For example, "64.233.160.0" is represented as "Google". For IPs whose DNS domain cannot be resolved, we label them as "unknown network". Finally, for command lines and system call types, we expand abbreviations to their full forms for semantic representation. For example, we expand the command "cp" to "copy" and the system call "execve" to "execute".

Event and gIoC Embedding. After lifting the system events into natural language representations, KNOWHOW converts system events and gIoCs into numerical vectors so that we can calculate their cosine similarity. To this end, we leverage FastText [24] to embed the lifted event entries and fields in gIoCs into numerical vectors. We chose FastText due to its ability to preserve semantics and be efficient. First, embedding vectors of FastText can preserve the semantics of

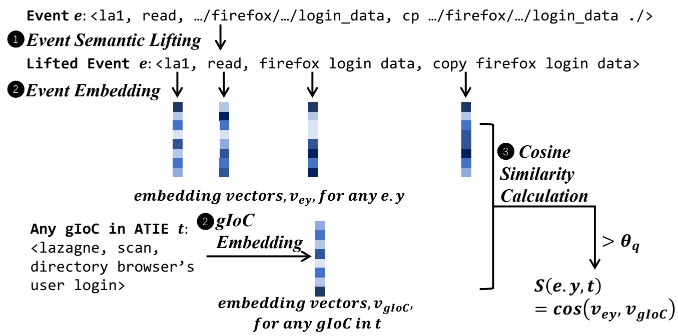

Fig. 2: The calculation process of $S(e.y, t)$.

words. For example, FastText ensures the distances between "run" and "execute" to be small so that we can match the system call "execute" to text "run" in CTI reports. Second, compared to other techniques like large language models, FastText is significantly faster. Such efficiency is critical for attack detection systems like KNOWHOW as they are resource-constrained [21]. Noting that we use large language models for offline gIoC extraction in Section IV-B due to their superior ability in identifying diverse and nuanced natural language entities. In contrast, we avoid using LLMs to directly analyze system events for efficiency.

Query Acceleration. A naive approach to realizing $ProvQ(e.y, \theta_q)$ is to compare e to each ATIE, t, in CKD and then further enumerate each gIoC within t to calculate the score. However, such a native method is time-consuming and unsuitable for online APT attack detection. To accelerate the queries, we devised a two-stage searching method to eliminate the need to compare every gIoC in CKD. The key idea is driven by the observation that gIoCs can be very different semantically. Thus, we do not need to compare system events to gIoCs that are semantically far away. For example, we do not need to calculate the similarity between "remove" and "add a file". Our two-stage searching method first clusters gIoCs using the Mean-Shift algorithm [10], [64] based on their embedding vectors. We chose Mean-Shift because it is a non-parametric algorithm that does not require setting hyper-parameters, such as the number of clusters manually. Additionally, it is suitable for distributions with irregular shapes and varying densities, making it capable of handling clusters of arbitrary shapes, which aligns with our scenario where clustering involves a large number of security-specific terms. Thus, during the search, given a field $e.y$ of a system event e, KNOWHOW first identifies the cluster closest to $e.y$ and then finds the most similar gIoCs within that cluster. This approach enables KNOWHOW to avoid comparing $e.y$ with gIoCs in other clusters, which are semantically far away.

Threshold Setting. We employ Grubbs' Test [14], a standard statistical method for outlier detection, to automatically determine the value of θ_q based on benign data. We opt for this method due to its elimination of the need for manually specified parameters. Specifically, for a given benign dataset comprising system events, we compute the similarity score of each system event within the benign dataset against each ATIE in the CKD. Subsequently, we apply the one-sided Grubbs' Test to determine the outlier detection threshold for these similarity scores, which serves as θ_q.

V. DESIGN OF KNOWHOW

KNOWHOW offers online, accurate, and interpretable detection of APT attacks based on CKD. The key idea is to utilize a similarity score between system events and ATIEs as an indicator of attacks, rather than relying on statistical anomalies detected through data-driven approaches. This design presents two key advantages over data-driven methods. Firstly, the alerts generated by KNOWHOW are more interpretable. Instead of simply presenting statistical anomaly scores, KNOWHOW can explain why a specific event is deemed malicious by identifying the attack technique used at each stage. Secondly, KNOWHOW offers higher precision by reducing false positives based on attack logic, which is a significant limitation in data-driven approaches.

A. Detection Workflow

The detection workflow of KNOWHOW is illustrated in Figure 3. It consists of three steps: ❶Detecting Anomalous Events: Given a stream of system events, KNOWHOW first matches each incoming event e with CKD to identify anomalous events. It detects anomalies by querying a system event in CKD using $ProvQ$, determining whether it corresponds to an attack technique documented in CKD. If an event matches at least one technique, it is classified as anomalous. ❷Constructing Provenance Graph: KNOWHOW constructs a provenance graph based on the detected anomalous events and treats it as a potential seed alert for attacks in an online manner, adopting the same online graph construction approach as NODLINK [42], which we chose for its high efficiency. During the graph construction, the event-level anomalies are treated as seed nodes and expand the subgraph by propagating anomaly scores. Based on the attack aggregation assumption [42], this expansion introduces lower-scored (benign in event-level detection) nodes, reducing false negatives. ❸Applying Reasoning Model: KNOWHOW analyzes provenance graph with our reasoning model, extended from APT Lifecycle model [54], [45]. This step helps eliminate false positives that do not contain sufficient steps to ensure a successful attack or that do not follow the logical sequence of attacks.

B. Attack Reasoning

KNOWHOW utilizes the knowledge of the APT Lifecycle to analyze attack graphs and reason about concise, APT Lifecycle-structured attack alerts. The core idea is that a successful attack must include sufficient steps that follow a logical sequence, for example,"Initial Access" must precede data theft. Missing critical stages (e.g., "Command and Control") suggests an incomplete attack, while steps violating the expected order are less likely to represent genuine threats, helping prioritize high-confidence, logically consistent alerts.

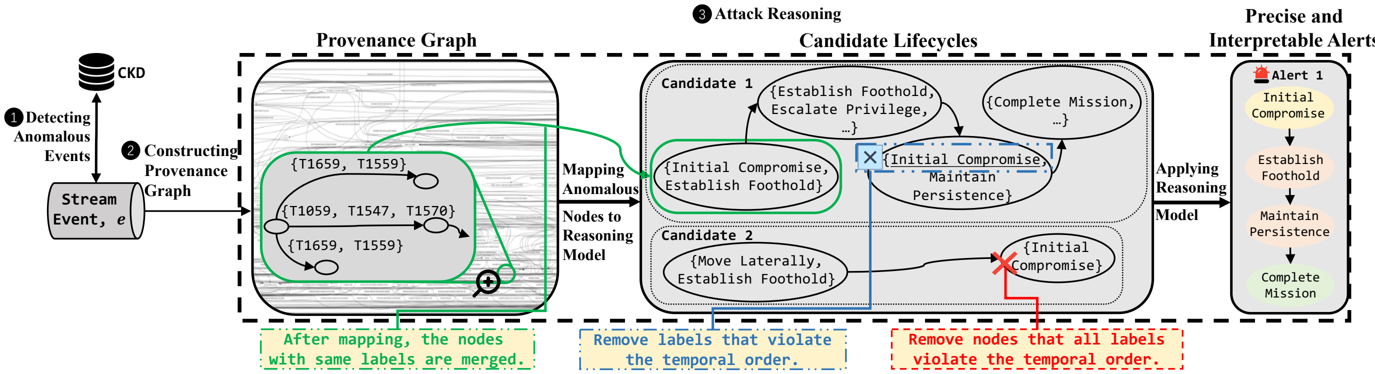

Fig. 3: The workflow of KNOWHOW. Given an event stream, KNOWHOW first queries CKD to detect anomalous events by labeling attack techniques associated with them. Each event may correspond to multiple techniques. Next, KNOWHOW constructs the provenance graph, with a zoomed-in view provided in the figure, highlighting the detected technique labels on the edges. Finally, KNOWHOW performs attack reasoning on the provenance graph to generate alerts. Specifically, it maps the anomalous nodes to reasoning model based on the technique labels of their outgoing edges, generating candidate lifecycles. And then, KNOWHOW applies the reasoning model to eliminate false positives and produce precise, interpretable alerts.

The attack reasoning consists of two steps. First, KNOWHOW maps nodes to the reasoning model based on the industry-standard APT Lifecycle stage models [56], [49], widely adopted in production environments [66], [21]. Second, KNOWHOW uses the reasoning model to verify the temporal order of stages within candidate lifecycles, remove invalid stages, check the completeness of attack stages, and generate alerts.

Reasoning Model. KNOWHOW's reasoning model is designed with two goals: (1) to provide strong guidance that improves attack detection accuracy, and (2) to ensure high interpretability, enabling security experts to understand detection results without extensive training.

To achieve our design goals, we propose a relaxed APT Lifecycle model based on the widely adopted APT Lifecycle model within the industry [15], [32], [19], [50], shown in Figure 4, where $A \rightarrow B$ indicates that stage A precedes stage B. This model fulfills the first design goal by encapsulating the general understanding of APT attacks. It captures the key steps and the temporal relationships among them for typical APT attacks in practice, enabling the identification of incomplete or chronologically inconsistent alerts as potential false positives. Regarding the second design goal, this model is derived from the standard industrial models, enabling security experts to grasp its terminology without specialized training.

Note that our reasoning model defines and uses a relaxed APT Lifecycle that only requires the "*Initial Compromise*" stage to precede all others and "*Complete Mission*" to occur last. No strict temporal order is imposed on intermediate stages such as "*Escalate Privilege*". This design reflects practical observations: attackers must first compromise a system before performing follow-up actions. We relax ordering constraints in the original Lifecycle models for other stages because they often share mutiple techniques—e.g., a C2 mechanism may support both "*Establish Foothold*" and "*Escalate Privilege*". Relaxing these constraints helps mitigate ambiguity and improves robustness in real-world scenarios.

Usage of the Reasoning Model. KNOWHOW uses the reasoning model to reason the candidate lifecycles and infer accurate attack alerts. It first streamlines alerts by removing stage labels that violate the model's temporal order. Then, by evaluating lifecycle completeness, it generates alerts for valid sequences and discards incomplete ones—likely false positives from benign behavior.

Streamlining Alerts. As a highly automated system, KNOWHOW may generate false positives in flagging anomalous events, for instance, misclassifying a benign port scan as "*Initial Compromise*". To address this, KNOWHOW removes events that violate the temporal order shown in Figure 4. For example, as illustrated in Figure 3, if *Initial Compromise*" appears after "*Establish Foothold*" in the provenance graph, the former is pruned, as re-compromising the system after already gaining access is logically inconsistent

Raising Alerts. Our insight to raise alerts is that failed or incomplete attack candidates lacking essential attack behaviors can be mistaken for benign behaviors. For example, an IT maintainer might also exhibit behavior resembling port scanning during the "*Initial Compromise*" stage. Simply detecting these incomplete attacks can lead to false positives. Therefore, KNOWHOW necessitates that an alert encompasses at least the "*Initial Compromise*" and "*Establish Foothold*" stages, as they are essential for an attacker to penetrate the victim system. Additionally, KNOWHOW demands that the alert include at least one stage from "*Escalate Privilege*", "*Internal Reconnaissance*", "*Move Laterally*", and "*Maintain Persistence*" because these represent the essential behaviors in an attack. Notice that the presence of "*Complete Mission*" is optional, as not all attackers will erase their tracks or harm the victim's system.

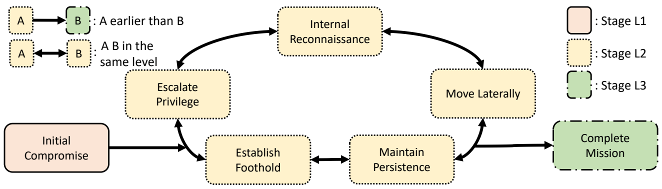

Fig. 4: Stages are grouped into three levels based on the temporal order. Double arrows indicate stages within the same level, with no fixed order; single arrows denote sequential progression between stages.

C. Mapping Anomalous Nodes to the Reasoning Model

To enable reasoning, KNOWHOW needs to map anomalous nodes in alerts to stages within our reasoning model to generate the candidate lifecycles. To this end, we first map the technique labels of each anomalous node to a tactic label. This step is straightforward, as the ATT&CK framework already provides the mapping relations [55]. Then, we employ the mapping rules in Table III, derived from our comprehension of the APT Lifecycle model, to assign the tactic labels to the respective APT Lifecycle stages.

TABLE III: Tactic-Lifecycle Stage Mapping Table.

Tactic	APT Lifecycle Stage
Reconnaissance, Initial Access	Initial Compromise
Execution, Resource Development, Command and Control	Establish Foothold
Privilege Escalation, Credential Access	Escalate Privilege
Discovery, Collection	Internal Reconnaissance
Lateral Movement	Move Laterally
Persistence, Defense Evasion	Maintain Persistence
Exfiltration, Impact	Complete Mission

The key challenge in mapping anomalous nodes to stage labels lies in the one-to-many relationships: a node may have multiple edges, while each single edge may correspond to multiple ATIEs in CKD, linked to different tactics and thus multiple APT stages. For instance, an alarm edge involving "T1053 Scheduled Task" and "T1546 Event Triggered Execution" may relate to tactics like "Execution", "Persistence" and "Privilege Escalation", mapping to stages including "Maintain Persistence" and "Escalate Privilege". As nodes typically have many outgoing edges, aggregating edge-level stage labels directly would assign multiple stages per node, causing a path explosion during reasoning due to the need to evaluate all stage label combinations for temporal order.

To address the one-to-many mapping problem, we transform it into a one-to-limited mapping via label merging, assigning edges, and subsequently nodes, a limited set of stage labels. Our core concept is to retains stage labels with "relatively high" scores, computed as the cumulative similarity ($Sim(e,t)$) between event e and matching ATIEs. Using Grubbs' Test, we identify high outliers as "relatively high" scores and keep their corresponding stages; if no high outliers exist, we remove low outliers and retain the rest. This strategy avoids over-reliance on the single highest-scoring stage, which may be a false positive. Since node stage labels are derived from their outgoing edges, which also yields a one-to-many mapping, we apply the same one-to-limited method to assign the node a refined set of stage labels with "relatively high" scores.

Specifically, We deduce the correct stage labels for a node by performing one-to-limited mapping over its outgoing edges. Nodes without successors, which means they have no generated events or outgoing edges, are skipped, and stage selection begins from their predecessors. For node n, the stage labels of its outgoing edges serve as candidate options, each weighted by its corresponding anomaly score. The labels with "relatively high" weights (determined via Grubbs' Test, as before) are selected for n. We then backtrack to n's predecessor to vote on its stage labels, repeating this process iteratively until all nodes in the attack graph are assigned APT lifecycle stages. Finally, nodes sharing the same stage label are merged into single nodes in the reasoning model to form candidate lifecycles, as illustrated in Figure 3.

VI. EVALUATION

We implemented a prototype of KNOWHOW and evaluated it with realistic attack scenarios. In this section, we first detail the implementation and the experimental environment. Then, we introduce our evaluation protocol, including metrics, knowledge sources, datasets, and baselines. To evaluate KNOWHOW, we focus on answering the following research questions:

RQ 1: Can KNOWHOW detect APT attacks accurately and precisely with the help of gIoC?
RQ 2: Does KNOWHOW correctly label the ATT&CK techniques to system events?
RQ 3: Is KNOWHOW efficient enough for online detection?
RQ 4: Is KNOWHOW robust against mimicry attacks, incomplete attack lifecycles, new and unseen attacks?
RQ 5: How each component in KNOWHOW contributes to the overall detection process?

A. Implementation and Experiment Environment

We implemented KNOWHOW on Python3.8.8 with around 2,000 lines of code (LoC) across all components. In our deployment, KNOWHOW utilizes the FastText model [24] to encode gIoCs and key information from input system events. The embedding space of FastText is pre-trained through CBOW model [52] using over 10,000 attack-related statements extracted from 1,500 paragraphs in the MITRE ATT&CK technique description [55] and 80 pieces of attack descriptions sourced from public CTI reports [1]. Meanwhile, KNOWHOW employs Sklearn [73] to implement the Mean-Shift algorithm in the query acceleration component (Section IV-C). For CKD and dataset management, KNOWHOW leverages the ElasticSearch database [22] for data management and querying. All experiments were conducted on a Ubuntu 22.04 machine with a GTX 3090 GPU, a 40-core 2.40GHz CPU, and 128GB of main memory.

B. Evaluation Protocol

Metrics. NodLink [42] has proposed to use graph-level and node-level accuracy to evaluate the effectiveness of attack detection more comprehensively. Graph-level accuracy comprises two parts: graph-level precision and graph-level recall, which are defined as $\frac{GTP}{GTP+GFP}$ and $\frac{GTP}{GTP+GFN}$, respectively. GTP, GFP, and GFN stand for graph-level true positives, false positives, and false negatives, respectively. The graph-level positive is the number of reported graphs that contain attack steps, while the negative represents the number of reported graphs that do not contain attack steps. Node-level accuracy also comprises two parts: node-level precision and node-level recall, which are defined as $\frac{NTP}{NTP+NFP}$ and $\frac{NTP}{NTP+NFN}$, respectively. NTP, NFP, and NFN are the numbers of node-level true positives, false positives, and false negatives, respectively. Given a reported provenance graph, we consider a node in the graph as an NTP if it represents an attack step.

Knowledge Sources. To build CKD, KNOWHOW crawled the technique descriptions from ATT&CK v10.0, which encompassed 567 techniques. Additionally, 80 CTI reports containing attack-related sentences were randomly selected from ThreatMiner [1] and Mandiant [49]. Following the construction of CKD, KNOWHOW generated one ATIE for each technique, amounting to a total of 567 entries. Among these, there were 27,652 gIoCs and 1,795 IoCs. All our CTI reports are dated before December 2022.

Datasets. The general information of our datasets is shown in Table IV. We first evaluate KNOWHOW on three public datasets: THEIA, TRACE, and In-lab Arena. THEIA and TRACE originate from the DARPA Transparent Computing Engagement #3 (E3) database [17], which has been extensively utilized in recent researches [42], [13], [88], [81]. And we labeled the ground truth nodes following the E3 document [18]. We did not incorporate the CADET dataset because the necessary information, such as full command lines and parameters, is missing in it, hindering fine-grained knowledge discovery. In-lab Arena [62] was simulated by NODLINK [42], replicating real attacks that took place at Sangfor, one of the most prominent Chinese security vendors. We rely on the documents provided by the NODLINK repository [42] for labeling the attack techniques in In-lab Arena. We refrain from using the DARPA Transparent Computing Engagement #5 (E5) dataset because we observed discrepancies between attack reports and actual logs in several cases in it. For example, it lacks the corresponding write event of the "sshdlog" injection operation documented in THEIA.

In addition to these three datasets, we constructed a newly simulated dataset, NewlySim, for our evaluation, with the aim of testing the effectiveness of KNOWHOW in handling unknown new attacks. Specifically, using NewlySim, we exploited two new vulnerabilities, CVE-2023-22809 [59] and CVE-2024-28085 [60]. These vulnerabilities were selected for their high risk and broad impact. For example, CVE-2023-22809 is rated "High" by the NVD [61], affects multiple

TABLE IV: Summary of our evaluation datasets. "Duration" and "Event Rate" denote the duration of data collection and the average number of events generated per second, respectively.

Dataset	# APTs	Duration	# Hosts	Event Rate	# Attack Actions
THEIA	1	247h	1	11.25 eps	97
TRACE	2	264h	1	75.76 eps	93
In-lab Arena	5	144h	5	48.23 eps	202
NewlySim	2	336h	3	168.69 eps	39
Open-World	6	168h	186	28.13eps	212

versions of "sudo", posing a widespread threat. Similarly, CVE-2024-28085 targets util-linux, which is a core component in most Linux distributions, making it highly relevant to Linux users. Most importantly, neither exploit is covered in our CTIs, as our CKD contains only CTIs published before December 2022.

To build NewlySim, we deployed a three-machine scenario comprising a hijacked attacker machine, an extranet interactive jump server machine, and an intranet working machine. The adversary first gained access to the intranet interactive jump server machine through social engineering channels and then utilized the "ssh" service on the springboard to infiltrate the intranet machines. On the intranet working machine, the adversary launched two attack campaigns, leveraging the aforementioned vulnerabilities to achieve privilege escalation. With system-level privileges, the adversary executed various malicious actions that are common in APT attacks, including malicious payload acquisition and execution [5], [83], internal environment reconnaissance [75], [16], OS credential dumping [84], [71], important file scanning and collection [12], [9], information leakage [58], [31], and attack evidence eradication [76], [82], thus completing an APT attack. We also used a trusted machine to log into the intranet machine to generate benign data through regular Microsoft Office operations. The attack lasted for 168 hours, during which we also collected the benign dataset in the same environment for 168 hours.

Finally, we also evaluated KNOWHOW using an Open-World dataset generated from a realistic industrial environment. We deployed KNOWHOW within one of the world's largest cloud providers, Huawei, and participated in their internal penetration test. During our experiment, KNOWHOW was deployed across over 180 endpoints, including diverse servers, workstations, and desktops, all running various operating systems such as Linux, Windows and openEuler. We first collected benign data over a 20-hour period and manually verified it to ensure no attacks were present. Then, a professional red team initiated attacks on the machines monitored by KNOWHOW over a seven-day period. The ground truth for the Open-World dataset was provided by the this red team.

For the DARPA datasets, we follow the same practice of NodLink [42] to split the training and test datasets. We first select 20% of the data that covers the time period of attacks as the test dataset. Then, we use the remaining 80% of the data, which only contains benign events, as the training dataset [42]. For other datasets, we use the provided benign data for training and anomaly data for testing. We first apply event-level detection on the training set to derive the threshold using

Grubbs' Test, and then evaluate KnowHow 's performance on the test set.

Overlap Between CTI Reports and Data. To ensure the fairness of the evaluation, we have verified that the attack techniques utilized in our datasets are NOT documented in our CTI reports directly. For the DARPA dataset, we manually inspected it to confirm that none of the knowledge in our CTI reports were used in the attacks. For the In-lab Arena dataset and the Open-World dataset, all of our CTI reports are dated before December 2022, which is before the generation dates of these datasets. We further examined the attack techniques employed in these two datasets and confirmed that they were not mentioned in the CTI reports. In our NewlySim dataset, we deliberately chose to use techniques that are dated after January 2023.

Baselines. To evaluate the effectiveness of KnowHow in detecting APT attacks, we compare it with five end-to-end provenance-based APT attack detection systems: Holmes [54], AirTag [20], NodLink [42], Kairos [13], and Extractor [72]. The first four baselines are non-CTI driven detection systems and Extractor is a state-of-the-art CTI-driven system that extracts IoC patterns from CTI reports. Extractor is a representative CTI-extraction system that extracts IoC graphs from CTI reports. Following the evaluation methodology in its paper [72], we use Extractor to generate graphs and feed them into Poirot for detection, ensuring a fair and comparable evaluation. To evaluate the effectiveness of gIoC in APT attack detection, we replace the knowledge in CKD with knowledge extracted using two state-of-the-art knowledge extraction methods: TTPDrill [36] and LADDER [2]. We also compare KnowHow with an IoC-only version of KnowHow, which utilizes IoCs extracted by IoCParser in the knowledge base. Table V lists a summary of all the baseline systems. These three CTI extraction baselines are designated as "TTPDrill + KnowHow", "LADDER + KnowHow", and "KnowHow with only IoC".

Setup for Baselines. For AirTag, NodLink, Kairos, Extractor, TTPDrill, and LADDER, we utilize the open-source code released by the respective authors. We apply all the optimizations and recommended hyperparameters outlined in their original papers to train the models using the same benign dataset as KnowHow. For Holmes, we adopt the implementation provided by the authors of NodLink. For AirTag, we retrain the BERT using the benign dataset, which is identical to the one used for the other baselines and KnowHow, following the training parameters specified in their open-source code.

C. RQ 1: Effectiveness in APT Attack Detection

We calculate the graph- and node-level accuracy of KnowHow and the baselines to evaluate the effectiveness of KnowHow in APT detection. The results are shown in Table VI and Table VII. KnowHow performs both the best graph- and node-level accuracies compared to the baselines.

Graph-Level Accuracy. For graph-level accuracy, KnowHow achieves the precision and recall of 1.00, except for Open-World dataset. There is a false positive of KnowHow at the graph level in the Open-World dataset. After an investigation by the red team, it was confirmed that the false positive was caused by an update to the system's security software, which involved serious high-risk behaviors similar to attacks. Nevertheless, all baselines also fail to detect this attack. Data-driven methods (i.e., NodLink, Kairos, and AirTag) have low graph-level precision because they rely on extracting normal behavior characteristics from benign data, which cannot handle rare but benign events. Holmes's graph-level precision is the lowest due to the incompleteness of the knowledge included and the over-generalization in the handling of objects, such as treating all non-trusted IP addresses as a means of initial compromise.

Node-Level Accuracy. For node-level accuracy, KnowHow has higher node-level precision and recall in each dataset than the baselines. Most baselines suffer from low precision and recall, because they cannot identify rare but benign nodes or have complete rules capturing all attack behaviors. For data-driven approaches, the training set significantly impacts detection effectiveness. That said, if normal behaviors are not included in the training set, they are prone to being identified as

TABLE V: Summary of our baselines.

Type	Name	Methodology	Venue	Year
Detection Baselines	HOLMES	Manual Rules	IEEE S&P	2019
	AIRTAG	Date-driven	Usenix Security	2023
	NodLink	Date-driven	NDSS	2024
	KAIROS	Date-driven	IEEE S&P	2024
	EXTRACTOR + POIROT	CTI-IoC-driven	Euro S&P	2021
CTI Extraction Baselines	TTPDrill + KnowHow *	Basic Syntax Analysis	ACSAC	2017
	LADDER + KnowHow *	Language Model	RAID	2023
	KnowHow with only IoC	Regular Expression matching	NA	NA

TABLE VI: The graph-level accuracy results for KnowHow and baselines. P stands for precision, and R stands for recall.

Detection Baselines	THEIA P	THEIA R	TRACE P	TRACE R	In-Lab Arena P	In-Lab Arena R	NewlySim P	NewlySim R	Open-World P	Open-World R
Holmes	**1.00**	**1.00**	0.15	**1.00**	0.04	**1.00**	0.14	**1.00**	0.15	0.40
AirTag	**1.00**	**1.00**	**1.00**	**1.00**	0.83	**1.00**	1.00	**1.00**	0.63	**1.00**
NodLink	**1.00**	**1.00**	0.67	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.71	**1.00**
Kairos	0.91	**1.00**	0.88	0.88	**1.00**	**1.00**	0.50	**1.00**	0.63	**1.00**
Extractor + Poirot	0.33	**1.00**	0.33	**1.00**	0.50	**1.00**	0.25	**1.00**	0.43	0.30
KnowHow	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.91**	**1.00**

TABLE VII: The node-level accuracy results for KnowHow and baselines. P stands for precision, and R stands for recall.

Detection Baselines	THEIA P	THEIA R	TRACE P	TRACE R	In-Lab Arena P	In-Lab Arena R	NewlySim P	NewlySim R	Open-World P	Open-World R
Holmes	0.01	0.98	0.01	0.74	0.01	0.32	0.01	0.40	0.04	0.21
AirTag	0.31	0.84	0.26	0.88	0.18	0.96	0.19	0.86	0.30	0.87
NodLink	0.23	**1.00**	0.25	**0.98**	0.17	0.92	0.28	0.94	0.48	0.90
Kairos	0.13	0.93	0.11	0.94	0.32	0.92	0.17	0.96	0.33	0.94
Extractor + Poirot	0.34	0.77	0.34	0.88	0.56	0.54	0.29	0.25	0.38	0.22
KnowHow	**0.62**	**1.00**	**0.82**	**0.98**	**0.82**	**1.00**	**0.78**	**1.00**	**0.82**	**1.00**

malicious. For example, in NewlySim, a normal user's unique file handling process, including using the *tar* command (absent from benign data), led to false alarms (NFPs) by all data-driven baselines except NODLINK. While NODLINK employs Grubbs' test and limited-step anomaly score propagation to reduce NFPs, it still generates a significant number due to the absence of node behavior and temporal logical order checks, as seen in KNOWHOW. HOLMES performs the worst due to its coarse-grained rule design. For example, HOLMES categorizes read operations originating from untrusted IP addresses as $Untrusted\_Read$. However, in normal scenarios, numerous legitimate network access operations occur, resulting in a large number of IP addresses that have never appeared in the system becoming untrusted, causing false positives.

Effectivenss of gIoC. To further evaluate the effectiveness of gIoC in CKD, we compare KNOWHOW's performance augmented with gIoC knowledge against three CTI extraction baselines. We find that KNOWHOW with gIoC achieves superior accuracy in both graph- and node-level detection, outperforming all baselines. The results are shown in Table VIII and Table IX.

TABLE VIII: The graph-level accuracy results for KNOWHOW and baselines. P stands for precision, and R stands for recall.

Dataset	CTI Extraction Baselines						Ours	
	TTPDRILL + KNOWHOW *		LADDER + KNOWHOW *		KNOWHOW with only IOC		KNOWHOW	
	P	R	P	R	P	R	P	R
THEIA	0.17	**1.00**	0.20	**1.00**	0.50	**1.00**	**1.00**	**1.00**
TRACE	0.50	**1.00**	0.40	**1.00**	0.67	**1.00**	**1.00**	**1.00**
In-Lab Arena	0.56	**1.00**	0.33	0.80	0.83	**1.00**	**1.00**	**1.00**
NewlySim	0.17	**1.00**	0.17	**1.00**	0.17	**1.00**	**1.00**	**1.00**
Open-World	0.67	0.80	0.38	0.30	0.35	0.60	**0.91**	**1.00**

TABLE IX: The node-level accuracy results for KNOWHOW and baselines. P stands for precision, and R stands for recall.

Dataset	CTI Extraction Baselines						Ours	
	TTPDRILL + KNOWHOW *		LADDER + KNOWHOW *		KNOWHOW with only IOC		KNOWHOW	
	P	R	P	R	P	R	P	R
THEIA	0.31	**1.00**	0.15	0.54	0.23	0.88	**0.62**	**1.00**
TRACE	0.67	**0.98**	0.12	0.63	0.18	0.83	**0.82**	**0.98**
In-Lab Arena	0.54	0.92	0.21	0.52	0.32	0.65	**0.82**	**1.00**
NewlySim	0.12	0.84	0.08	0.12	0.19	0.68	**0.78**	**1.00**
Open-World	0.59	0.37	0.11	0.37	0.11	0.27	**0.82**	**1.00**

For the "TTPDRILL + KNOWHOW" and "KNOWHOW with only IoC" baselines, the reliance on IoC alone, coupled with the limited number of corresponding IoCs in CTI reports, is the primary cause of poor performance. This limitation leads to numerous missed attack-related events, resulting in a large number of false negatives. Furthermore, both baselines generate many false positives, as IoCs are tied to specific paths, files, and command lines that don't necessarily indicate malicious behavior across all operations involving these entities. In our evaluation, we observe that both baselines can only detect malicious external IP nodes and nodes with specific command lines or file names (e.g., the *del* command used to erase attack traces, or the */etc/passwd* file accessed by attackers). However, they struggle to detect malicious processes or files disguised as normal nodes. Additionally, certain IoC-related files and folders can also be accessed by legitimate processes. Relying on IoC makes it difficult for KNOWHOW to distinguish between these benign external IPs and legitimate command operations, leading to a significant number of false positives.

"LADDER + KNOWHOW" performs poorly because part of the extracted knowledge contains a significant amount of irrelevant information, which distorts the matching process with system events. More specifically, the incomplete and imprecise extraction by LADDER leads to incorrect query scores and ultimately disrupts the accurate scores derived from true IoCs.

In contrast, with gIoCs, KNOWHOW can identify attack behaviors, such as downloading of a malicious file from an external IP, the execution of the malicious file, and the removal of the malicious file after execution is completed, even if the names of malicious nodes don't appear in the CTI. This is because, using the event semantic lifting method in Section IV-C, KNOWHOW can detect the malicious process and file node from the granularity of behavior and maps the gIoC dealing with the situation when the malicious file is renamed to disguise itself as a normal file.

D. RQ 2: Accuracy of Technique Labeling

A key advantage of KNOWHOW is its ability to link ATT&CK technique labels to low-level system events, yielding interpretable and precise detection. With properly assigned ATT&CK technique labels, KNOWHOW can map system events to lifecycle stages. This capability enables temporal reasoning to filter out likely benign sequences, thereby enhancing detection precision without compromising interpretability.

Therefore, in this section, we evaluates KNOWHOW 's accuracy in assigning technique labels. The main challenge lies in establishing ground truth. While the Open-World dataset is fully labeled with ATT&CK techniques, the In-Lab Arena dataset is partially labeled, and for the remaining unlabeled datasets, we engaged the same red team in Open-World dataset to annotate techniques based on attack documents. To ensure consistency, each attack behavior was assigned at most three techniques during both annotation and expert review. A labeling is considered correct if KNOWHOW matches any ground-truth techniques. Our evaluation covers 643 attack actions spanning 12 tactics and 65 techniques.

In our experiment, KNOWHOW accurately labels 559 of all 643 attack actions, accounting for a 87.0% accuracy. After manual inspection, 507 out of 559 actions can be labeled by gIoCs while only 101 actions can also be identified by IoCs. For example, in the dataset, there is an event accessing the process memory file, (*cp, read, /proc/11793/mem*), which is triggered by the attacker performing an OS credential dumping. For this event, no IoC mentioned in CTI can be matched with the exact path of "/proc/11793/mem". However, there are related sentences in CTI stating "The attackers usually dump the process memory in the system." With these sentences, KNOWHOW can extract gIoC as (attacker, dump, process memory in the system), which can then be matched to the event. Of the 84 mislabeled actions, 69 are those that are not labeled by KNOWHOW as the top 3 accurate labels, and the other 15 are incorrectly labeled by KNOWHOW.

The reason why KNOWHOW does not give accurate label to the 69 attack actions is that these actions correspond to multiple attack techniques, causing ambiguity. For example, in the In-Lab Arena dataset, attackers downloaded malicious payloads by abusing PowerShell commands. This top 3 actions can correspond to *Ingress Tool Transfer (T1105)*, *Scheduled Task/Job (T1053)*, and *Develop Capabilities-Malware (T1587.001)*, while KNOWHOW gives *Abuse PowerShell (T1059.001)*, which results in an incorrect label. Inspecting the candidate label set of this attack action in KNOWHOW, we find that *Ingress Tool Transfer (T1105)* is also a candidate, but with a lower similarity score.

For the other 15 mislabeled attack actions, it is because KNOWHOW did not properly understand the semantics of the attack action, resulting in incorrect labeling. For example, in our datasets, the attackers used the *echo* command to write invalid data to a malicious payload (execution file) to achieve an obfuscated operation, which reflected *Obfuscated Files or Information (T1027)*. The KNOWHOW incorrectly labeled this attack action as *Windows Command Shell (T1059.003)* because it didn't understand the attack correctly and thought it was a simple malicious command execution.

Although labeling inaccuracies may appear concerning, they do not severely hinder attack reasoning, as predicted techniques are often semantically related to the ground truth. This enables KNOWHOW to map them to similar lifecycle stages, preserving reasoning flow and preventing premature pruning of valid paths, thus avoiding graph-level false positives. Meanwhile, Section VI-G explains how attack reasoning further mitigates event-level false positives from technique labeling. In addition to the overall analysis presented above, Section VII provides concrete examples illustrating how technique labels are captured by matching system events with gIoCs. We recognize that our evaluation datasets and covered attack techniques are limited in representing real-world complexity.

To facilitate future research, Figure 5 presents the ten most frequently observed MITRE ATT&CK techniques in our experiments. The most common techniques are primarily associated with four tatics, including *Execution*, *Internal Reconnaissance*, *Command and Control*, and *Credential Access*. Among these, techniques under *Internal Reconnaissance* (e.g., commands such as *ifconfig* and *arp*) are relatively easy to detect using IoCs, as often involving well-known system utilities. In contrast, the remaining techniques are more challenging to identify with IoCs alone, and require gIoCs for accurate mapping. The case-level details are illustrated in Section VII.

E. RQ 3: Efficiency

To evaluate the efficiency, we compare the throughput of KNOWHOW with baselines on different datasets. The throughput is defined as the number of system events processed per second (eps). We omit TTPDRILL and LADDER because they share the detection component of KNOWHOW, leading to the same throughput.

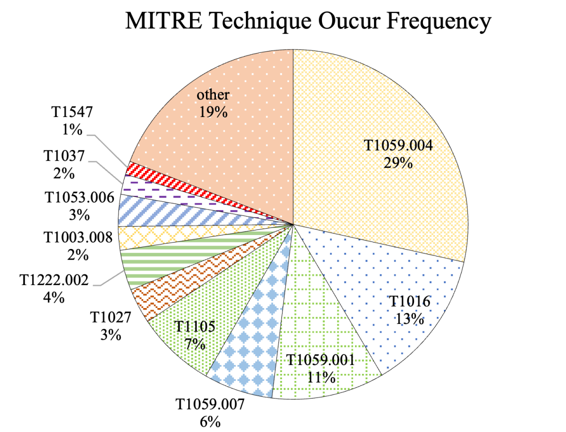

Fig. 5: Top 10 frequent techniques in our experiments.

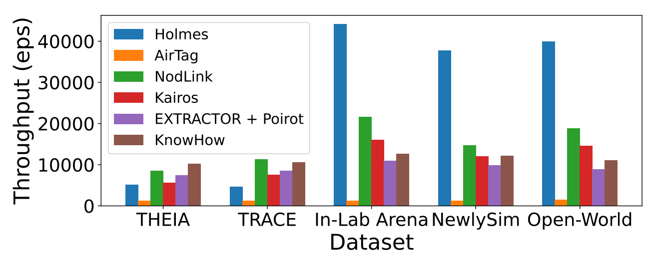

Fig. 6: Working throughput among different frameworks. TTPDRILL and LADDER have the same throughput as KNOWHOW.

Figure 6 depicts the throughput of all the systems across different datasets. KNOWHOW is comparable to the SOTA online detection works. HOLMES achieves exceptionally high throughput in the In-Lab Arena and NewlySim datasets as it only has 16 rules, resulting in the lowest accuracy in Section VI-C. AIRTAG exhibits the lowest throughput because it uses the time-consuming BERT for event encoding.

Moreover, we analysis the impact of CTI scale on detection efficiency. Larger volumes of CTIs may generate more gIoCs within CKD, potentially slowing down $ProvQ$ performance. However, the theoretical time complexity of $ProvQ$ is $O(\log(n))$, where n denotes the number of gIoCs, and that is a complexity that remains acceptable even as the number of gIoCs increases. Specifically, under the low-dimensional manifold hypothesis, which is empirically effective for textual data[7], [85], our Mean-Shift algorithm clusters the n gIoCs into k clusters, where $k = O(\log(n))$. Furthermore, by organizing the k ATIE clusters in CKD using a KD-tree structure, the approximate time complexity of $ProvQ$ becomes $O(k + \log(n/k))$[8], [27]. Substituting $k = O(\log(n))$, this complexity simplifies to $O(\log(n))$. Therefore, the efficiency of KNOWHOW remains largely unaffected by the scale of CTI.

TABLE X: Detection result of KNOWHOW on the Mimic-Prov dataset with different insertion ratios. Columns with an insertion ratio of 0.00 indicate the original dataset and no mimic structure is inserted. P, R are the same as Table VI.

Ratio	0.00		0.20		0.50		1.00		2.00	
Metrics	P	R	P	R	P	R	P	R	P	R
Graph	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Node	0.78	1.00	0.43	1.00	0.51	1.00	0.31	0.90	0.42	1.00

TABLE XI: Detection result of KNOWHOW on the incomplete attacks with different integrity ratios. Columns with the attack integrity ratio of 1.00 indicate the original dataset and no attack behaviors is removed. P, R are the same as Table VI.

Ratio	0.60		0.70		0.80		0.90		1.00	
Metrics	P	R	P	R	P	R	P	R	P	R
Graph	1.00	0.67	1.00	0.67	1.00	1.00	1.00	1.00	1.00	1.00
Node	0.76	0.83	0.76	0.82	0.84	1.00	0.74	1.00	0.82	1.00

TABLE XII: The intermediate detection result for each component of KNOWHOW. P, R are the same as Table VI.

Node-level Result	Event-level Detection		Graph Construction		Attack Reasoning	
	P	R	P	R	P	R
THEIA	0.47	0.74	0.41	1.00	0.62	1.00
TRACE	0.67	0.83	0.60	0.98	0.82	0.98
In-Lab Arena	0.52	0.74	0.46	1.00	0.82	1.00
NewlySim	0.48	0.70	0.46	1.00	0.78	1.00
Open-World	0.63	0.79	0.57	1.00	0.82	1.00

F. RQ 4: Robustness

Mimicry Attacks. Existing research has demonstrated that existing APT detection systems, particularly those based on graph learning, exhibit considerable vulnerability when facing mimicry attacks [29]. They find that attackers can mislead graph-learning-based detection approaches by inserting some benign behaviors into their attacks, causing the attack provenance graph to contain benign structures. The proportion of the size of the above-mentioned benign subgraph to the average size of normal benign structures in the benign data is called the insertion ratio. It shows that when the insertion ratio exceeds 2.00%, the attacker can escape the detection algorithm.

To evaluate the robustness of KNOWHOW against mimicry attacks, we constructed a mimicry dataset, Mimic-Prov, following the steps described in [29], and evaluate KNOWHOW on it. We include this dataset in our code repository. Note that we did not utilize th dataset openly sourced by [29] because it removes contextual information (e.g., process names and command-line arguments) which is crucial for KNOWHOW's detection. To accommodate the impact of insertion ratios of benign substructures on the system's performance, we constructed multiple datasets with four different ratios: 0.20%, 0.50%, 1.00% and 2.00%. Table X shows statistic details.

Our experimental results show that KNOWHOW successfully detected ALL attack events at the graph level across these insertion ratios within the Mimic-Prov dataset. Notably, at a 1.00% insertion ratio, node-level recall drops below 1, indicating successful evasion of some attack actions by mimic structures. Upon further analysis, we find that the missed nodes correspond to simple, shallow reconnaissance activities (e.g., ifconfig, arp), which lack complex event sequences and derived successor nodes, hence are more vulnerable to mimicry attacks. However, more complex and deep attacks, which dominate our datasets, are less susceptible to such interference.

Incomplete Attacks. We also evaluate how incomplete attack lifecycles impact the detection effectiveness and robustness of KNOWHOW. We conduct a controlled experiment based on the In-lab Arena dataset. The In-lab Arena dataset was chosen for its unparalleled complexity of attack steps, enabling reliable incomplete-scenario construction. We simulate incomplete attacks by removing attack steps from the original attack sequences in the dataset. We define the attack integrity ratio as the ratio of remaining to original attack steps, and evaluate KNOWHOW across the modified datasets with integrity ratios of 0.60, 0.70, 0.80, 0.90, and 1.00. Detection performance is measured in terms of precision and recall as in Section VI-B.

The results are in Table XI. We observe that when the attack integrity ratio is 0.80 or higher, both graph- and node-level detection performance remain stable and effective, except for a limited increase in node-level false positives. It means removing a small portion of attack behaviors does not compromise the overall integrity of the attack lifecycle, thereby allowing normal alerting through attack reasoning.

When the attack integrity ratio falls below 0.80, we notice an increase in false negatives at the graph-level detection. For ratios of 0.60 and 0.70, two of three attack graphs are detected, while a simpler graph (less than 50 nodes) is missed. We further find that the missed attack graph only contains *the Initial Compromise stage and Establish Foothold stage*, which is at the earlier stage of the attack. These stages have minimal behavioral distinction from benign activity, thus challenging for all knowledge-based methods.

New and Unseen Attacks. To further evaluate KNOWHOW's capability in detecting unseen attacks, besides the statistic analysis on NewlySim dataset in Section VI-C, we conducted a case study on the attack in NewlySim, which exploits CVE-2023-22809, a privilege escalation vulnerability absent from CKD, as one of attack actions, in Section VII.

G. RQ 5: Component-wise Analysis

We conduct a component-wise analysis to illustrate the impact of each component of KNOWHOW by measuring node-level precision and recall after each step. We follow the same evalution protocal mentioned in Section VI-B. The steps includes event-level detection with gIoC (Step 1), constructing provenance graphs based on anomalous events (Step 2) and attack reasoning (Step 3). This breakdown reveals how successive stages refine results, balancing early-stage trade-offs. The results are in Table XII and we analyze it as follows.

Step 1 initially identifies high-confidence anomalies (e.g., Lua script execution, sudo privilege escalation). However, events with benign-like features (e.g., ambiguous network connections) may fall below the detection threshold, limiting recall (In-Lab Arena: 74% recall). Step 2 improves recall (up

26%) by propagating scores to connected nodes. For example, undetected attacker-controlled sshd nodes inherit scores from detected downstream actions and these scores then exceed the detection threshold, enabling the previously missed nodes to be identified. However, over-propagation to benign nodes (e.g., hostguard spawning malicious sh) introduces false positives, reducing precision (down 6%). Step 3 restores precision (up 36%) by enforcing lifecycle constraints. Temporally inconsistent nodes (e.g., hostguard process incorrectly labeled in Establish Foothold) are filtered, while fragmented attacks are reconstructed through subgraph alignment which improves the final node-level precision.

In conclusion, our three-stage design systematically addresses earlier limitations: propagation recovers missed signals (recall), while knowledge-guided reasoning suppresses contextual false positives (precision). The interplay demonstrates KNOWHOW's robustness against sophisticated APT attacks.

VII. CASE STUDY

In this section, we analyze how KNOWHOW detects unseen attacks and Living off the Land (LotL) techniques through case studies on the NewlySim and Open-World datasets.

NewlySim Dataset. Figure 7 shows the detection result of KNOWHOW on an APT campaign from NewlySim dataset, which exploits CVE-2023-22809—a privilege escalation vulnerability that not documented in CKD. Attackers first uploaded the exploit script *exp.sh* via *scp*, then executed it to gain *sudo* privileges through *sudoedit*, accessing critical system files. Finally, attackers collected and compressed password files and transferred them to the jump server using LotL processes (*cp*, *tar*, *scp*).

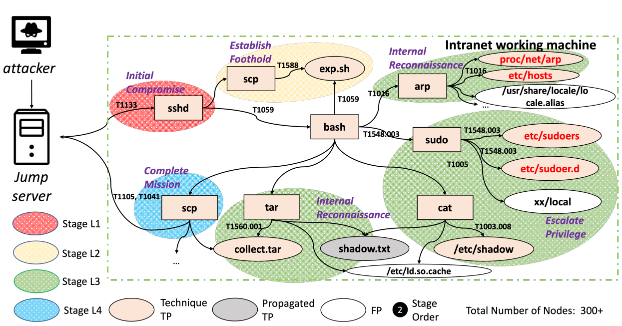

Fig. 7: Detection result of the APT campaingn using CVE-2023-22809 from the NewlySim dataset.

Despite CVE-2023-22809 being undocumented in CKD and the LotL actions, KNOWHOW successfully detected all attack actions by attackers through gIoCs with few false positives, whereas the SOTA baselines failed. This capability stems from three key factors: ① Although CVE-2023-22809 is not documented, CKD contains similar attack behaviors. For instance, the ATIE of T1548.003 includes gIoCs like (Adversary, read, sudoers file), (Adversary, perform, sudoers file), which align with system events generated when accessing the *sudoers* file. KNOWHOW uses this semantic similarity to detect novel vulnerabilities and infer attack intent. ② KNOWHOW can identifies LotL behaviors involving legitimate tools like *cat*, *scp*, and *tar*. For example, the ATIE of T1548.003 contains the gIoC, (Adversary, read, etc shadow file), matching events where the attacker used *cp* to read the *shadow* file. Event Semantic Lifting and Event and gIoC Embedding first lift the "cat" command to "show", then reduce the distance between "read" and "show" through embedding, allowing KNOWHOW to realize the LotL actions. ③ Attack reasoning enables KNOWHOW to reduce false positives. For instance, other benign *tar* operations of normal users accessing shared files (e.g., `/etc/ld.so.cache` in Figure 7) that are often misclassified by data-driven baselines, are correctly identified as non-malicious through contextual analysis in attack reasoning.

Open-World Dataset. Figure 8 presents the detection result of KNOWHOW on an APT campaign from the Open-World dataset that exemplifies the extensive use of LotL techniques, leaving minimal forensic footprints. Specifically, attackers first accessed to the target host from a compromised server, and leveraged the LotL command, *curl*, to download a malicious script, *t.sh*. Subsequently, attackers executed *t.sh*, which invoked three Base64-encoded, obfuscated Python commands to perform internal network reconnaissance and sensitive data exfiltration. To evade detection, attackers used the LotL command, *rm*, to delete the script after execution. Finally, the stolen data is exfiltrated via the built-in *redis* service, which is then terminated by the LotL command, *pkill*, to eliminate evidence of the attack.

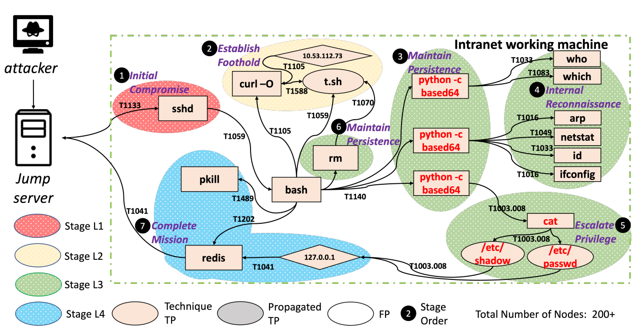

Fig. 8: Detection result of the APT case in the Open-World dataset.

Although this attack campaign employed several LotL techniques, KNOWHOW achieved full detection with minimal false positives, while the other baselines failed. This capability also stems from the same three key factors: ① CKD contains the attack knowledge that that closely reflects attacker behaviors. For example, the ATIE of T1140 includes gIoCs such as (Adversary, use, base64 encoded file) and (Attacker, run, base64 obfuscated scripts), which aligns with the attacker's use of obfuscated Python commands in the event-level detection. Similarly, other malicious activities, such as downloading the *t.sh* script via *curl* from a compromised host, removing the script, exfiltrating sensitive data, and performing internal reconnaissance, were also successfully detected at the event level. ② KNOWHOW can effectively identify LotL behaviors. For example, the LotL behaviors, such as invoking *redis*, *who*, and *id*, are assigned high scores with the existing CKD knowledge are subsequently filtered out during event-level detection. ③ Attack reasoning enables KNOWHOW to reduce false positives especially on the LotL behaviors. For example, data-driven approaches generates numerous false positives in this case, because the *redis* service is widely used legitimate system users, and its benign usage can easily be misclassified as malicious. KNOWHOW significantly reduces such false positives during the attack reasoning step. This is because this false positives occurred either before the Initial Access stage or after the Complete Mission stage and were therefore filtered out by KNOWHOW, significantly reducing false alarms.

VIII. Other Related Work

Data-Driven Provenance Analysis. Data-driven methods [42], [88], [86], [4], [13], [87], [20], [37], [81], [30] are widely studied in APT detection. These methods typically employ deep learning models to extract features from provenance graphs and then apply various algorithms to classify nodes or subgraphs as benign or malicious. Different from CTI-based methods, data-driven methods do not rely on static knowledge of attack techniques and vulnerabilities but instead learn from historical data to identify patterns and anomalies. However, they often face challenges such as high resource consumption, high false positive rates, and difficulties in interpreting results [21].

IX. Discussion

Mimicry Attacks and Evasion. A potential evasion involves an adaptive attack where the attacker, aware of KNOWHOW's design, blends attack steps with crafted benign steps to mislead the one-to-limited mapping of reasoning module, as happened in Section VI-F. For this to succeed, attackers must: ① effectively misclassify most attack steps using plausible benign actions, otherwise KNOWHOW can infer the attack lifecycle from remaining steps; ② have full knowledge of CKD and KNOWHOW 's training data, requiring extensive local testing to refine attack sequences. These conditions make such attacks technically challenging and require nontrivial work. Hence, while theoretically possible, the likelihood of a successful adaptive attack evading KNOWHOW appears very low.

Limitations: We acknowledge three limitations in the design and evaluation of KNOWHOW. First, while KNOWHOW demonstrates scalability to previously unseen and LotL attacks by capturing behavioral mechanisms (as demonstrated in Section VII), it cannot detect entirely novel attacks where every step is completely unprecedented and exhibits no similarity to known behaviors in the CKD. Although such cases are rare in practice, this limitation is inherent to all knowledge-driven detection approaches. Second, like prior systems (e.g., NODLINK, HOLMES, KAIROS), KNOWHOW requires a complete attack graph for accurate alerts, which introduces detection delays and makes early-stage detection challenging due to limited information. However, the experimental results in Section VI-F demonstrate that KNOWHOW can still achieve accurate detection with relatively low false positive rates, even before the attack has fully completed. Third, our current evaluation is based on a limited set of CTI reports and datasets. These datasets cover only a subset of known attack techniques, as illustrated in Section VI-D, and the types and distributions of these techniques may not fully reflect real-world attack scenarios. Therefore, whether KNOWHOW can maintain strong performance in more complex, real-world environments remains to be validated through future deployment efforts.

X. Conclusion

In this paper, we propose KNOWHOW, a CTI-knowledge-driven online provenance analysis solution that can automatically learn high-level attack knowledge from CTI reports and apply this knowledge to detect APT attacks in low-level system events. Our evaluation shows that KNOWHOW outperforms existing baselines in terms of both accuracy and interpretability. For interpretability, KNOWHOW can automatically map system events to high-level technique descriptions and summarize them into APT Lifecycle stages, while none of the baselines can achieve this goal. Furthermore, KNOWHOW maintains the same level of efficiency as the baselines and is robust to attacks stemming from unknown vulnerabilities and existing mimicry attacks.

Acknowledgments

This work was partly supported by the National Science and Technology Major Project of China (2022ZD0119103), the National Natural Science Foundation of China (62172009), and Huawai Research Fund.

References

[1] "ThreatMiner," 2024, accessed on April 25, 2024. [Online]. Available: https://threatminer.org/

[2] M. T. Alam, D. Bhusal, Y. Park, and N. Rastogi, "Looking Beyond IoCs: Automatically Extracting Attack Patterns from External CTI," in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, ser. RAID '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 92–108. [Online]. Available: https://doi.org/10.1145/3607199.3607208

[3] AlessandroZ, "Lazagne," 2017, accessed on July 25, 2025. [Online]. Available: https://github.com/AlessandroZ/LaZagne/

[4] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "ATLAS: A Sequence-based Learning Approach for Attack Investigation," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3005–3022. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/alsaheel

[5] "SANDWORM INTRUSION SET CAMPAIGN TARGETING CENTREON SYSTEMS," ANSSI, 2021, accessed on April 25, 2024. [Online]. Available: https://www.cert.ssi.gouv.fr/uploads/CERTFR-2021-CTI-005.pdf/

[6] H. Z. Azadeh Tabiban, Y. J. Lingyu Wang, and M. Z. Makan Pourzandi, "ProvTalk: Towards Interpretable Multi-level Provenance Analysis in Networking Functions Virtualization (NFV)." in *NDSS*, 2022.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[8] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, p. 509–517, Sep. 1975. [Online]. Available: https://doi.org/10.1145/361002.361007

[9] Bermejo, “BLACKENERGY & QUEDAGH,” 2020, accessed on April 25, 2024. [Online]. Available: https://blog-assets.f-secure.com/wp-content/uploads/2019/10/15163408/BlackEnergy_Quedagh.pdf/

[10] M. A. Carreira-Perpiñán, “Fast nonparametric clustering with gaussian blurring mean-shift,” in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML ’06. New York, NY, USA: Association for Computing Machinery, 2006, p. 153–160. [Online]. Available: https://doi.org/10.1145/1143844.1143864

[11] O. Caspi, “Teamtnt with new campaign aka ”chimaera”,” 2021, accessed on July 25, 2025. [Online]. Available: https://levelblue.com/blogs/labs-research/teamtnt-with-new-campaign-aka-chimaera/

[12] “ROCKET KIT TEN: A CAMPAIGN WITH 9 LIVES,” CHECK POINT, 2015, accessed on April 25, 2024. [Online]. Available: https://blog.checkpoint.com/wp-content/uploads/2015/11/rocket-kitten-report.pdf/

[13] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, “Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance,” 2023.

[14] N. Couderc, “GRUBBS: Stata module to perform Grubbs’ test for outliers,” 2007.

[15] “ADVANCED PERSISTENT THREAT (APT),” CrowdStrike, 2023, accessed on April 25, 2024. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/advanced-persistent-threat-apt/

[16] “A BAZAR OF TRICKS: FOLLOWING TEAM9’S DEVELOPMENT CYCLES,” Cyberreason Nocturnus, 2020, accessed on April 25, 2024. [Online]. Available: https://www.cybereason.com/blog/a-bazar-of-tricks-following-team9s-development-cycles/

[17] “DARPA Transparent Computing Engagement #3,” DARPA, 2018, accessed on December 19, 2023. [Online]. Available: https://github.com/darpa-i2o/Transparent-Computing/

[18] DARPA, “Transparent computing engagement 3 data release,” 2018, accessd on March 5, 2023. [Online]. Available: https://drive.google.com/drive/folders/1QlbUFWAGq3Hpl8wVdzOdIoZLFxkII4EK/

[19] “Lifecycle of the Advanced Persistent Threat,” DELL SecureWorks, accessed on April 25, 2024. [Online]. Available: https://docs.media.bitpipe.com/io_10x/io_105022/item_550605/Lifecycle_of_the_Advanced_Persistent_Threat%5B1%5D.pdf/

[20] H. Ding, J. Zhai, Y. Nan, and S. Ma, “AIRTAG: Towards Automated Attack Investigation by Unsupervised Learning with Log Texts,” in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 373–390. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/ding-hailun-airtag

[21] F. Dong, S. Li, P. Jiang, D. Li, H. Wang, L. Huang, X. Xiao, J. Chen, X. Luo, Y. Guo, and X. Chen, “Are we there yet? An Industrial Viewpoint on Provenance-based Endpoint Detection and Response Tools,” in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2396–2410. [Online]. Available: https://doi.org/10.1145/3576915.3616580

[22] Elasticsearch, “Elasticsearch,” *software, version*.

[23] ESET, “New osx/keydnap malware is hungry for credentials,” 2016, accessed on July 25, 2025. [Online]. Available: https://www.welivesecurity.com/2016/07/06/new-osxkeydnap-malware-hungry-credentials/

[24] “FastText. ,” Facebook., 2017, accessed on October 1, 2023. [Online]. Available: https://fasttext.cc/

[25] P. Fang, P. Gao, C. Liu, E. Ayday, K. Jee, T. Wang, Y. F. Ye, Z. Liu, and X. Xiao, “Back-Propagating system dependency impact for attack investigation,” in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 2461–2478. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/fang

[26] “APT37(reaper): The overlooked north korean actor,” FireEye, 2021, accessed on April 25, 2024. [Online]. Available: https://services.google.com/fh/files/misc/apt37-reaper-the-overlooked-north-korean-actor.pdf

[27] J. H. Friedman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Trans. Math. Softw.*, vol. 3, no. 3, p. 209–226, Sep. 1977. [Online]. Available: https://doi.org/10.1145/355744.355745

[28] P. Gao, F. Shao, X. Liu, X. Xiao, Z. Qin, F. Xu, P. Mittal, S. R. Kulkarni, and D. Song, “Enabling Efficient Cyber Threat Hunting With Cyber Threat Intelligence,” in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021, pp. 193–204.

[29] A. Goyal, X. Han, G. Wang, and A. Bates, “Sometimes, you aren’t what you do: Mimicry attacks against provenance graph host intrusion detection systems,” in *30th Network and Distributed System Security Symposium*, 2023.

[30] A. Goyal, G. Wang, and A. Bates, “R-CAID: Embedding Root Cause Analysis within Provenance-based Intrusion Detection,” in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 3515–3532.

[31] “Project TajMahal – a sophisticated new APT framework,” GReAT, 2019, accessed on April 25, 2024. [Online]. Available: https://securelist.com/project-tajmahal/90240/

[32] A. F. Gregg Lindemulder, “What are advanced persistent threats?” 2024, accessed on April 25, 2024. [Online]. Available: https://www.ibm.com/topics/advanced-persistent-threats/

[33] W. U. Hassan, A. Bates, and D. Marino, “Tactical Provenance Analysis for Endpoint Detection and Response Systems,” in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1172–1189.

[34] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, “Nodoze: Combatting threat alert fatigue with automated provenance triage,” in *Network and Distributed Systems Security Symposium*, 2019.

[35] W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, D. Wang, Z. Chen, Z. Li, J. Rhee, J. Gui, and A. Bates, “This is why we can’t cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage,” in *Annual Computer Security Applications Conference*, ser. ACSAC ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 165–178. [Online]. Available: https://doi.org/10.1145/3427228.3427255

[36] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, “TTPDrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources,” in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 103–115. [Online]. Available: https://doi.org/10.1145/3134600.3134646

[37] Z. Jia, Y. Xiong, Y. Nan, Y. Zhang, J. Zhao, and M. Wen, “MAGIC: Detecting advanced persistent threats via masked graph representation learning,” in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 5197–5214. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/jia-zian

[38] B. Jin, E. Kim, H. Lee, E. Bertino, D. Kim, and H. Kim, “Sharing cyber threat intelligence: Does it really help?” in *NDSS*, 2024.

[39] “Cyberwar in Ukraine leads to all-time-high levels of DDoS attacks,” Kaspersky, 2022, accessed on April 25, 2022. [Online]. Available: https://www.kaspersky.com/about/press-releases/2022_cyberwar-in-ukraine-leads-to-all-time-high-levels-of-ddos-attacks/

[40] “Virus.MSWord.Aos,” Kaspersky, 2024, accessed on January 1, 2025. [Online]. Available: https://threats.kaspersky.com/en/threat/Virus.MSWord.Aos/

[41] H. I. Kure, S. Islam, and H. Mouratidis, “An integrated cyber security risk management framework and risk predication for the critical infrastructure protection,” *Neural Computing and Applications*, vol. 34, no. 18, pp. 15 241–15 271, 2022.

[42] S. Li, F. Dong, X. Xiao, H. Wang, F. Shao, J. Chen, Y. Guo, X. Chen, and D. Li, “NODLINK: An Online System for Fine-Grained APT Attack Detection and Investigation.” in *NDSS*, 2024.

[43] Z. Li, J. Zeng, Y. Chen, and Z. Liang, “AttacKG: Constructing Technique Knowledge Graph from Cyber Threat Intelligence Reports,” in *Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 589–609. [Online]. Available: https://doi.org/10.1007/978-3-031-17140-6_29

[44] M. Lv, H. Gao, X. Qiu, T. Chen, T. Zhu, J. Chen, and S. Ji, “TREC: APT Tactic / Technique Recognition via Few-Shot Provenance Subgraph Learning,” in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 139–152. [Online]. Available: https://doi.org/10.1145/3658644.3690221

[45] “MANDIANT Threat Intelligence,” Mandiant, accessed on April 30, 2024. [Online]. Available: https://www.mandiant.com/advantage/threat-intelligence/

[46] “This Is Not a Test: APT41 Initiates Global Intrusion Campaign Using Multiple Exploits,” Mandiant, 2020, accessed on December 19, 2023. [Online]. Available: https://www.mandiant.com/resources/blog/apt41-initiates-global-intrusion-campaign-using-multiple-exploits/

[47] "APT41, A DUAL ESPIONAGE AND CYBER CRIME OPERATION," Mandiant, 2022, accessed on May 19, 2024. [Online]. Available: https://www.mandiant.com/sites/default/files/2022-02/rt-apt41-dual-operation.pdf/
[48] "Unc3524: Eye spy on your email," Mandiant, 2022, accessed on December 19, 2023. [Online]. Available: https://www.mandiant.com/resources/blog/unc3524-eye-spy-email/
[49] "Cyber Security & Threat Intelligence Webinars," Mandiant, 2023, accessed on April 25, 2024. [Online]. Available: https://www.mandiant.com/resources/webinars/
[50] "Targeted attack lifecycle," Mandiant, 2023, accessed on April 25, 2024. [Online]. Available: https://www.mandiant.com/resources/insights/targeted-attack-lifecycle/
[51] "ETW events in the common language runtime," Microsoft, 2017, accessed on October 1, 2023. [Online]. Available: https://msdn.microsoft.com/en-us/library/ff357719(v=vs.110).aspx/
[52] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computer Science*, 2013.
[53] S. M. Milajerdi, B. Eshete, R. Gjomemo, and V. 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*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1795–1812. [Online]. Available: https://doi.org/10.1145/3319535.3363217
[54] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, "HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1137–1152.
[55] "MITRE ATT&CK," The MITRE Corporation, 2023, accessed on October 10, 2023. [Online]. Available: https://attack.mitre.org/
[56] "MITRE ATT&CK Tactic," The MITRE Corporation, 2023, accessed on October 10, 2023. [Online]. Available: https://attack.mitre.org/tactics/enterprise/
[57] "Structured threat information expression," The MITRE Corporation, 2024. [Online]. Available: https://stixproject.github.io/
[58] "NICKEL targeting government organizations across Latin America and Europe," MSTIC, 2021, accessed on April 25, 2024. [Online]. Available: https://www.microsoft.com/security/blog/2021/12/06/nickel-targeting-government-organizations-across-latin-america-and-europe/
[59] "CVE-2023-22809 Detail," NATIONAL VULNERABILITY DATABASE, 2023, accessd on March 3, 2024. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2023-22809/
[60] "CVE-2024-28085 Detail," NATIONAL VULNERABILITY DATABASE, 2024, accessd on May 3, 2024. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2024-28085/
[61] "National vulnerability database," NIST, 2024. [Online]. Available: https://nvd.nist.gov/
[62] Nodlink, "Simulated-data," 2024, accessed on April 30, 2024. [Online]. Available: https://github.com/Nodlink/Simulated-Data/
[63] "ioc-parser," PaloAlto, 2014, accessed on May 10, 2024. [Online]. Available: https://github.com/PaloAltoNetworks/ioc-parser/
[64] S. Paris and F. Durand, "A Topological Approach to Hierarchical Segmentation using Mean Shift," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.
[65] PyMuPDF, "Pymupdf: Python bindings for mupdf," 2021, accessed on January 1, 2025. [Online]. Available: https://pypi.org/project/PyMuPDF/
[66] M. R. Rahman, R. M. Hezaveh, and L. Williams, "What Are the Attackers Doing Now? Automating Cyberthreat Intelligence Extraction from Text on Pace with the Changing Threat Landscape: A Survey," *ACM Comput. Surv.*, vol. 55, no. 12, Mar. 2023. [Online]. Available: https://doi.org/10.1145/3571726
[67] RedCanary, "Atomic red team," 2021, accessd on March 5, 2023. [Online]. Available: https://github.com/redcanaryco/atomic-red-team/
[68] "The linux audit framework." Redhat., accessed on October 1, 2023. [Online]. Available: https://github.com/linux-audit/
[69] D. Rhoades, "Machine actionable indicators of compromise," in *2014 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2014, pp. 1–5.
[70] N. Rostovcev, "Apt41 world tour 2021 on a tight schedule," 2022, accessed on July 25, 2025. [Online]. Available: https://www.group-ib.com/blog/apt41-world-tour-2021/
[71] Rusu, "Iranian Chafer APT Targeted Air Transportation and Government in Kuwait and Saudi Arabia," 2020, accessed on April 25, 2024. [Online]. Available: https://www.bitdefender.com/blog/labs/iranian-chafer-apt-targeted-air-transportation-and-government-in-kuwait-and-saudi-arabia/
[72] K. Satvat, R. Gjomemo, and V. Venkatakrishnan, "Extractor: Extracting attack behavior from threat reports," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021, pp. 598–615.
[73] "scikit-learn," scikit-learn, 2023, accessed on December 19, 2023. [Online]. Available: https://scikit-learn.org/stable/index.html/
[74] "Executable installers are vulnerableŴEVIL (case 7): 7z*.exe allows remote code execution with escalation of privilege," SECLIST, 2015, accessed on December 19, 2023. [Online]. Available: https://seclists.org/fulldisclosure/2015/Dec/34/
[75] Slowok, "The baffling berserk bear: A decade's activity targeting critical infrastructure," 2021, accessed on April 25, 2024. [Online]. Available: https://vblocalhost.com/uploads/VB2021-Slowik.pdf/
[76] S. Smith, "Darkwatchman: A new evolution in fileless techniques," 2021, accessed on April 25, 2024. [Online]. Available: https://www.prevailion.com/darkwatchman-new-fileless-techniques/
[77] "Survey of the state of security 2022," Splunk Inc., 2022, accessed on April 13, 2022. [Online]. Available: https://www.splunk.com/zh_cn/pdfs/resources/e-book/state-of-security-2022.pdf/
[78] "Stanford CoreNLP," Stanford NLP Group, accessed on October 10, 2023. [Online]. Available: https://stanfordnlp.github.io/CoreNLP/
[79] "3 Advanced Persistent Threat (APT) Examples You Should Know About," Swiss Cyber Institute, 2024, accessed on April 25, 2024. [Online]. Available: https://swisscyberinstitute.com/blog/guide-of-advanced-persistent-threat-apt/
[80] "Sysdig." Sysdig., 2013, ccessed on October 1, 2023. [Online]. Available: https://sysdig.com
[81] M. Ur Rehman, H. Ahmadi, and W. Ul Hassan, "Flash: A comprehensive approach to intrusion detection via provenance graph representation learning," in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 3552–3570.
[82] "Malware Analysis Report (MAR) - 10135536-B," USCERT, 2017, accessed on April 25, 2024. [Online]. Available: https://www.us-cert.gov/sites/default/files/publications/MAR-10135536-B_WHITE.PDF/
[83] "AppleJeus: Analysis of North Korea's Cryptocurrency Malware," USCERT, 2021, accessed on April 25, 2024. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/aa21-048a/
[84] T. Wadhwa-Brown, "Where 2 worlds collide Bringing Mimikatz et al to UNIX," 2018, accessed on April 25, 2024. [Online]. Available: https://labs.portcullis.co.uk/download/eu-18-Wadhwa-Brown-Where-2-worlds-collide-Bringing-Mimikatz-et-al-to-UNIX.pdf/
[85] B. Wang, S. Wang, Y. Cheng, Z. Gan, R. Jia, B. Li, and J. J. Liu, "Infobert: Improving robustness of language models from an information theoretic perspective," in *International Conference on Learning Representations (ICLR 2021)*, March 2021. [Online]. Available: https://www.microsoft.com/en-us/research/publication/infobert-improving-robustness-of-language-models-from-an-information-theoretic-perspective/
[86] Q. Wang, W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, J. Rhee, Z. Chen, W. Cheng, C. A. Gunter *et al.*, "You are what you do: Hunting stealthy malware via data provenance analysis." in *NDSS*, 2020.
[87] Z. Xu, P. Fang, C. Liu, X. Xiao, Y. Wen, and D. Meng, "DEPCOMM: Graph Summarization on System Audit Logs for Attack Investigation," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 540–557.
[88] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "SHADEWATCHER: Recommendation-guided Cyber Threat Analysis using System Audit Records," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 489–506.
[89] T. Zhu, J. Yu, C. Xiong, W. Cheng, Q. Yuan, J. Ying, T. Chen, J. Zhang, M. Lv, Y. Chen, T. Wang, and Y. Fan, "APTSHIELD: A Stable, Efficient and Real-Time APT Detection System for Linux Hosts," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 5247–5264, 2023.