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

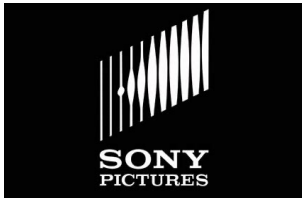
Yuhan Meng†, **Shaofei Li**†, Jiaping Gui‡, Peng Jiang§, and Ding Li†*

† Key Laboratory of High-Confidence Software Technologies (MOE), School of Computer Science, Peking University ,

‡ School of Computer Science, Shanghai Jiao Tong University, § Southeast University

Advanced Persistent Threats

- **APT attacks** pose a critical challenge to the modern world.
  - **Advanced**: Use cutting-edge techniques → Zero-day exploit, well-designed malware.
  - **Stealthy:** Evade detection by operating covertly → Living-off-the-land (LotL) attack, fileless attack.
  - **Persistent**: Maintain long-term access. → The SolarWinds attackers remained undetected for over 14 months.
- This underscores the critical need for effective detection.

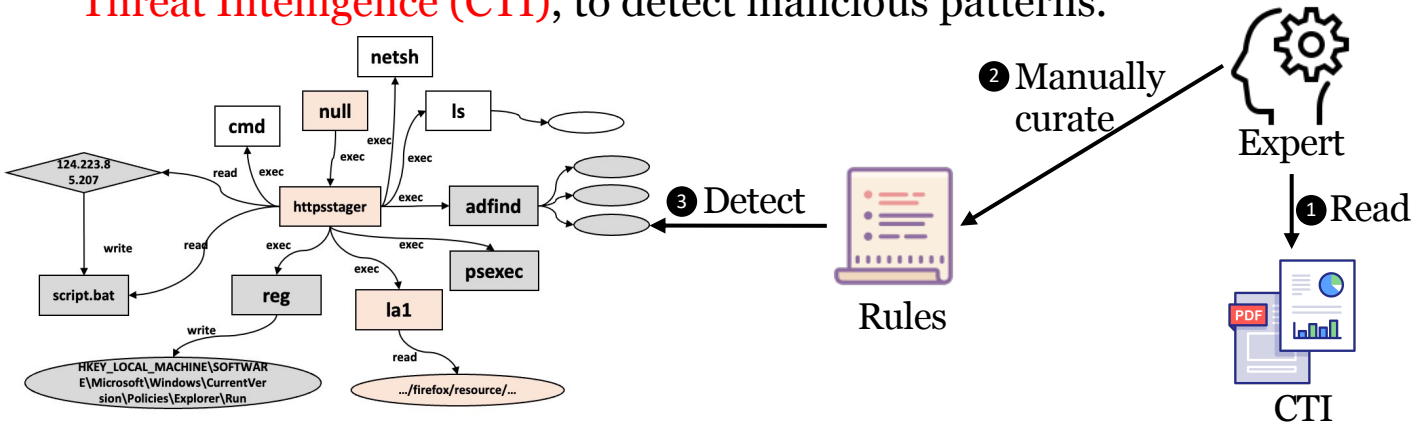


Provenance-Based Detection Systems

- Provenance-based detection systems build a directed, attributed provenance graph that models system behaviors over time.
  - **Node:** system entities (processes, files, sockets, etc.)
  - **Edge:** system events (read, write, fork, execve, sendto, recvfrom, etc.)

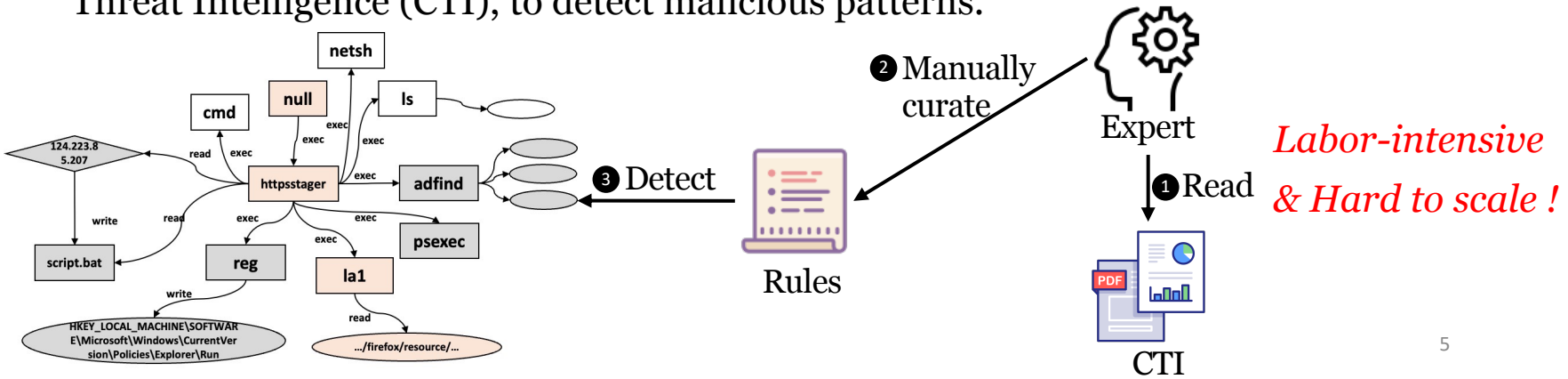
Provenance-Based Detection Systems

- Provenance-based detection systems build a directed, attributed provenance graph that models system behaviors over time.
  - **Node:** system entities (processes, files, sockets, etc.)
  - **Edge:** system events (read, write, fork, execve, sendto, recvfrom, etc.)
- One effective class of methods is **knowledge-driven approaches**
  - Encode attack knowledge, manually curated by security experts from Cyber Threat Intelligence (CTI), to detect malicious patterns.



Provenance-Based Detection Systems

- Provenance-based detection systems build a directed, attributed provenance graph that models system behaviors over time.
  - **Node:** system entities (processes, files, sockets, etc.)
  - **Edge:** system events (read, write, fork, execve, sendto, recvfrom, etc.)
- One effective class of methods is **knowledge-driven approaches**
  - Encode attack knowledge, manually curated by security experts from Cyber Threat Intelligence (CTI), to detect malicious patterns.



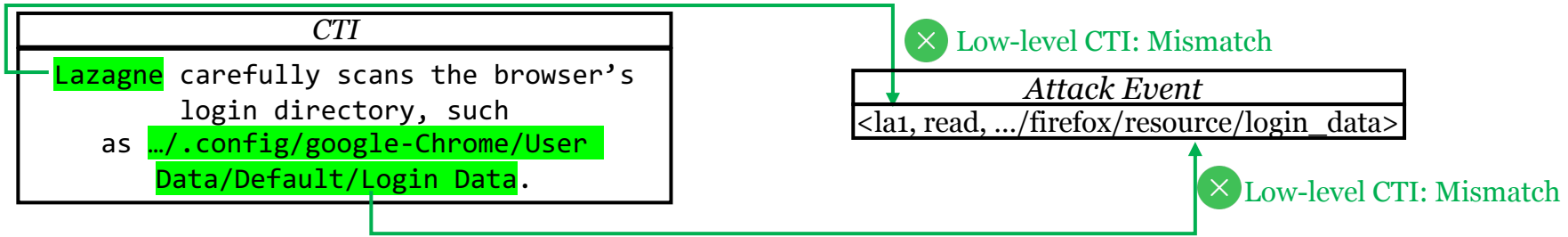
Motivation

北京大学 PEKING UNIVERSITY

- What prevents knowledge-driven approaches from being automatic and extensible?

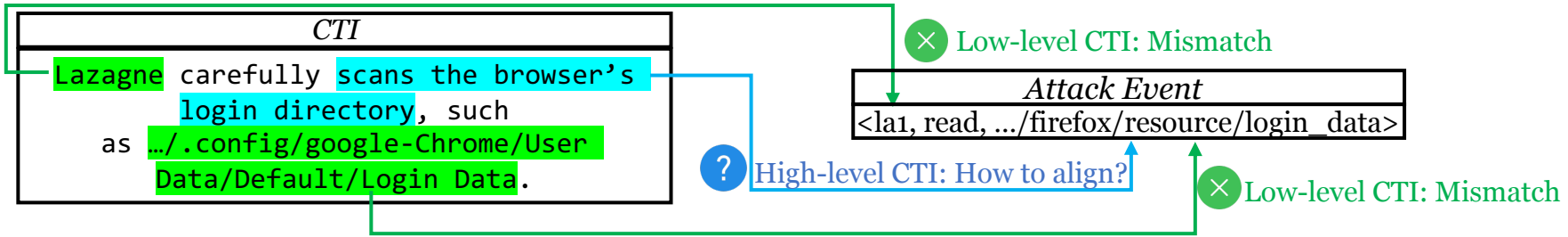
Motivation

- What prevents knowledge-driven approaches from being automatic and extensible?
- Two types of CTI knowledge:
  - **Low-level** (instantiated) : Indicator of Compromise (IoC) in CTI → Automatically extracted and matched by IoC-based methods (e.g., EXTRACTOR, TTPDrill, AttacKG, LADDER).



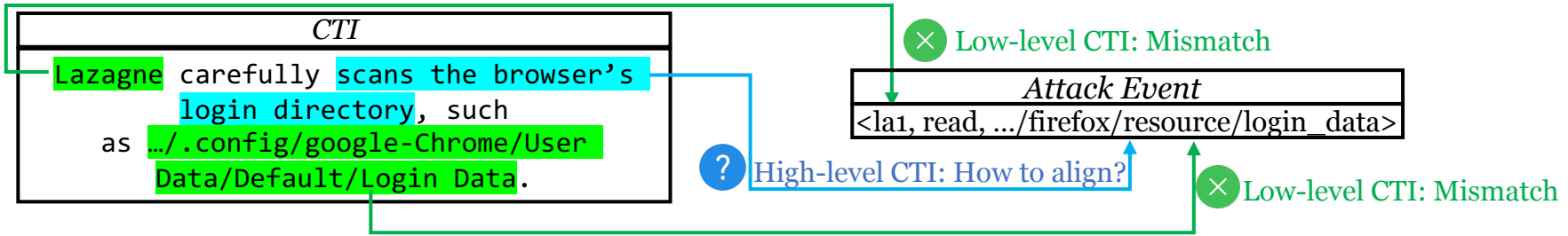
Motivation

- What prevents knowledge-driven approaches from being automatic and extensible?
- Two types of CTI knowledge:
  - **Low-level** (instantiated) : Indicator of Compromise (IoC) in CTI → Automatically extracted and matched by IoC-based methods (e.g., EXTRACTOR, TTPDrill, AttacKG, LADDER).
  - **High-level** (non-instantiated) : General behavioral descriptions in CTI → Cannot be matched automatically due to the **lack of linguistic similarity** with provenance events. (e.g., LADDER fails to correlate CTI sentences with system events).



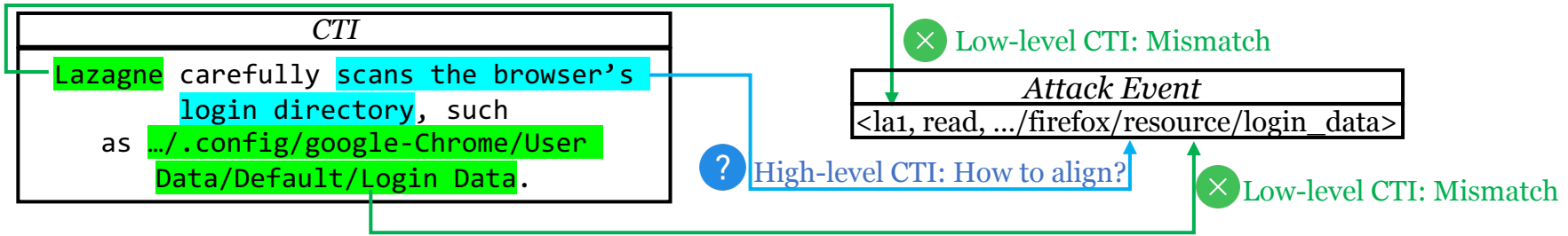
Motivation

- What prevents knowledge-driven approaches from being automatic and extensible?
- Two types of CTI knowledge:
  - **Low-level** (instantiated) : Indicator of Compromise (IoC) in CTI → Automatically extracted and matched by IoC-based methods (e.g., EXTRACTOR, TTPDrill, AttacKG, LADDER).
  - **High-level** (non-instantiated) : General behavioral descriptions in CTI → Cannot be matched automatically due to the **lack of linguistic similarity** with provenance events. (e.g., LADDER fails to correlate CTI sentences with system events).
- Consequently, this rich high-level knowledge can only be leveraged through manual rule writing by experts, limiting the automation and extensibility of detection systems.



Motivation

- What prevents knowledge-driven approaches from being automatic and extensible?
- Two types of CTI knowledge:
  - **Low-level** (instantiated) : Indicator of Compromise (IoC) in CTI → Automatically extracted and matched by IoC-based methods (e.g., EXTRACTOR, TTPDrill, AttacKG, LADDER).
  - **High-level** (non-instantiated) : General behavioral descriptions in CTI → Cannot be matched automatically due to the **lack of linguistic similarity** with provenance events. (e.g., LADDER fails to correlate CTI sentences with system events).
- Consequently, this rich high-level knowledge can only be leveraged through manual rule writing by experts, limiting the automation and extensibility of detection systems.



- **Observation: The Semantic Gap Prevents Automatic Use of High-Level CTI in Knowledge-based Approaches**

Key Insight

北京大学 PEKING UNIVERSITY

- To overcome this semantic gap, our key insight is that high-level CTI can be automatically mapped to provenance events by aligning them into a subject-verb-object (SVO) semantic structure.

Key Insight

- To overcome this semantic gap, our key insight is that high-level CTI can be automatically mapped to provenance events by aligning them into a subject-verb-object (SVO) semantic structure.
- CTI descriptions can be extracted into three semantic roles, which can map to the key elements in provenance events:
  - Attack conductors (who) → processes
  - Attack actions (what operation) → syscall
  - Attack targets (on what) → targeted files
- Unlike static and context-free IoCs, this structure captures both entities and their contextual descriptions, enabling more accurate, context-aware detection.

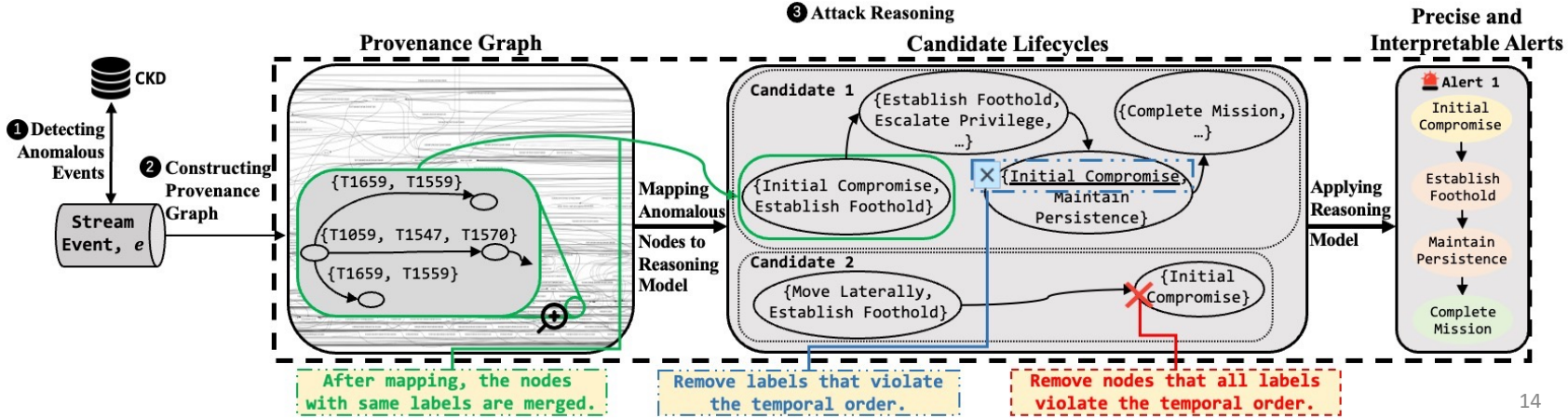
Sentences in CTI	Provenance Events
Lazagne scanned browser resources to steal credentials.	{<left out> "evt.type": "read", "fd.name": "…/firefox/resource…", "proc.cmdline": "la1 -s firefox", "proc.name": "la1 ", <left out>}
Attackers may send the stolen data to the external server hold held by them, to achieve the goal of exfiltration.	{<left out> "evt.type": "sendto", "fd.name": "17.x.x.x->12.5.8.10", "proc.cmdline": "scp ./collect /12.5.8.10", "proc.name": "scp", <left out>}
Some malicious js scripts may be executed and injected into proc filesystem, which can evade process-based defenses as well as possibly elevate privileges.	{<left out> "evt.type": "write", "fd.name": "/proc/filesystem", "proc.cmdline": " sh -r ./exp.js", "proc.name": " sh", <left out>}

Key Challenges

- **C1: How to extract complete and context-preserving SVO triples from CTI reports.**
  - **Problem:** Real-world CTI uses complex, passive, or fragmented sentences, while standard NLP parsers may miss critical modifiers of the entities.
  - **Solution1:** Propose general Indicator of Compromise (gIoC) representation with context-aware SVO extraction.
- **C2: How to efficiently map gIoCs to system events.**
  - **Problem:** Terms like “steal” or “exfiltrate” in gIoCs have no lexical overlap with system entities like “read” or “sendto”, making direct matching impossible.
  - **Solution2:** Design CTI Knowledge Database (CKD) to enable semantic lifting and accelerated mapping against system events.
- **C3: How to detect multi-stage APT behaviors from isolated event-level mapping.**
  - **Problem:** Event-level detection yields fragmented alerts, while APTs require reconstructing cross-process, time-spanning attack chains.
  - **Solution3**: Novel attack reasoning method based on the stages of the APT Lifecycle.

Our Solution: KnowHow

- KnowHow bridges the semantic gap between high-level CTI and low-level provenance events and achieves attack detection through a three-stage pipeline.
  - Preprocessing: Extract gIoCs from CTI reports, and store them in the CKD.
  - Stage 1: Detect anomalous system events by querying gIoC in CKD.
  - Stage 2: Construct the provenance graph starting from the detected anomalous events.
  - Stage 3: Apply reasoning model to analyze candidate alert lifecycles, to eliminate false positives.



C1: How to Extract High-level Knowledge

北京大学 PEKING UNIVERSITY

- **Solution1:** We propose general Indicator of Compromise (gIoC) representation with context-aware SVO extraction.
- **Definition of gIoC**
  - We define gIoC as a structured, semantic triple that captures the core behavioral intent of an attack described in CTI, in the form of:

$$gIoC = \langle Conductor, Action, Target \rangle$$

  - The key difference between gIoCs and conventional SVO triples is that the subjects and objects of gIoCs are attack-relevant concepts and their associated information, such as modifiers describing these concepts.
  - gIoCs extract attack information from a behavioral perspective, instead of the instance-level details, enabling it to reflect higher-level information.

C1: How to Extract High-level Knowledge

北京大学 PEKING UNIVERSITY

- **Solution1:** We propose gIoC (general Indicator of Compromise) representation with context-aware SVO extraction.
- **Extraction of gIoC**
  - Step 1: Identify attack-relevant concepts (*N*), if *N* fulfills one of the following:
    - *N* is an IoC, such as a file name, an IP address, a file hash, etc.
    - *N* is a domain name.
    - *N* is the name of an application or malware.
    - *N* is a command (e.g., *cp*) or its full name (e.g., *copy*)
    - *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.

C1: How to Extract High-level Knowledge

北京大学 PEKING UNIVERSITY

- **Solution1:** We propose gIoC (general Indicator of Compromise) representation with context-aware SVO extraction.
- **Extraction of gIoC**
  - Step 1: Identify attack-relevant concepts (*N*), if *N* fulfills one of the following:
    - *N* is an IoC, such as a file name, an IP address, a file hash, etc.
    - *N* is a domain name.
    - *N* is the name of an application or malware.
    - *N* is a command (e.g., *cp*) or its full name (e.g., *copy*)
    - *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.
  - Step 2: Identify the associated information of the attack-relevant concepts
    - We extract the modifiers (e.g., "browser data") and subclauses (e.g., "steal the credential files of system users") that describe the concept and serve as its associated information.

C2: How to Efficiently Use gIoCs

北京大学 PEKING UNIVERSITY

- **Solution2:** We design CTI Knowledge Database (CKD) to enable semantic lifting and accelerated mapping when using gIoCs to detect system events.
- **Structure of CKD**
  - CKD is defined as a set of ATT&CK Technique Information Entries (ATIEs), each representing an attack technique as defined by MITRE ATT&CK.
  - 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 ($list_{gioc}$).
  - $list_{gioc}$ contains gIoCs, which capture high-level attack knowledge reported in the CTI reports in $list_{cti}$.

C2: How to Efficiently Use gIoCs

北京大学 PEKING UNIVERSITY

- **Solution2:** We design CTI Knowledge Database (CKD) to enable semantic lifting and accelerated mapping when using gIoCs to *detect system events*.
- **Operation of CKD: *ProvQ***
  - $ProvQ$ matches the low-level system events to ATIEs in CKD, which are then flagged as **anomalous**.
  - For input system event $e = (source, destination, syscalltype, commandline)$, $ProvQ$ returns a list of ATIEs that **match** the given event.
  - An event e matches an ATIE t only if its similarity score $Sim(e,t) > \theta_q$, where θ_q is the given query threshold. And $Sim(e,t)$ is defined as

$$Sim(e,t) = \sum S(e.y,t) \ , where\ y \in \{source, destination, syscalltype, commandline\},$$

$and\ S(e.y,t)\ is\ the\ occurrences\ of\ gIoCs\ in\ t\ that\ appear\ in\ e.y\ field.$

C2: How to Efficiently Use gIoCs

北京大学 PEKING UNIVERSITY

- **Solution2:** We design CTI Knowledge Database (CKD) to enable semantic lifting and accelerated mapping when using gIoCs to *detect system events*.
- **Operation of CKD: *ProvQ***
  - $ProvQ$ matches the low-level system events to ATIEs in CKD, which further is flagged as **anomalous**.
  - For input system event $e = (source, destination, syscalltype, commandline)$, $ProvQ$ returns a list of ATIEs that **match** the given event.
  - An event e matches an ATIE t only if its similarity score $Sim(e,t) > \theta_q$, where θ_q is the given query threshold. And $Sim(e,t)$ is defined as

$$Sim(e,t) = \sum S(e.y,t) \ , where\ y \in \{source, destination, syscalltype, commandline\},$$

$and\ S(e.y,t)\ is\ the\ occurrences\ of\ gIoCs\ in\ t\ that\ appear\ in\ e.y\ field.$

How to define 'appear in' ?

C2: How to Efficiently Use gIoCs

北京大学 PEKING UNIVERSITY

- **Solution2:** We design CTI Knowledge Database (CKD) to enable *semantic lifting* and accelerated mapping when using gIoCs to detect system events.
- **Operation of CKD: *ProvQ***
  - Calculate $S(e.y, t)$: Terms like "steal" or "exfiltrate" in gIoC in t have no lexical overlap with system entities like " read" or " sendto" in $e.y$, making direct matching impossible.

C2: How to Efficiently Use gIoCs

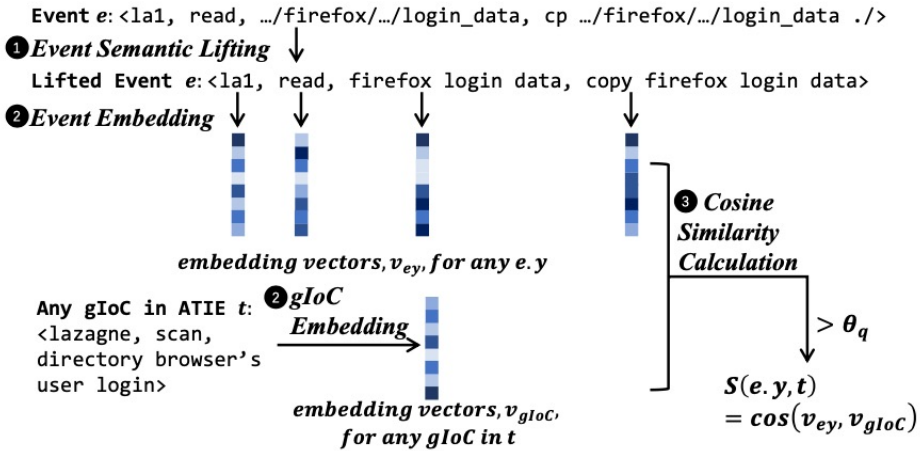
- **Solution2:** We design CTI Knowledge Database (CKD) to enable *semantic lifting* and accelerated mapping when using gIoCs to detect system events.

- **Operation of CKD: *ProvQ***
  - Calculate $S(e.y, t)$: Terms like "steal" or "exfiltrate" in gIoC in t have no lexical overlap with system entities like " read" or " sendto" in $e.y$, making direct matching impossible.
  - We propose semantic lifting to elevate low-level system events to natural language sentences that can be matched with gIoCs, which further helps to calculate $S(e.y, t)$ precisely.

	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

C2: How to Efficiently Use gIoCs

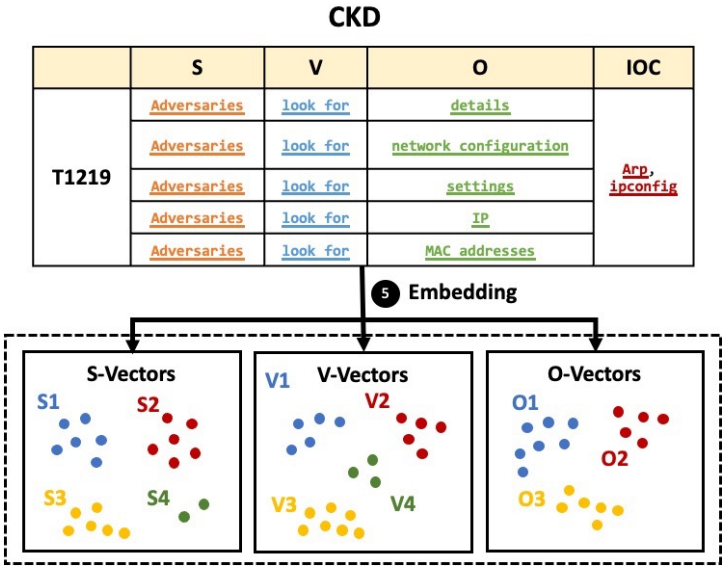
- **Solution2:** We design CTI Knowledge Database (CKD) to enable *semantic lifting* and accelerated mapping when using gIoCs to detect system events.
- **Operation of CKD: *ProvQ***
  - Calculate $S(e.y, t)$: Terms like "steal" or "exfiltrate" in gIoC in t have no lexical overlap with system entities like " read" or " sendto" in $e.y$, making direct matching impossible.
  - We propose semantic lifting to elevate low-level system events to natural language sentences that can be matched with gIoCs, which further helps to calculate $S(e.y, t)$ precisely.



C2: How to Efficiently Use gIoCs

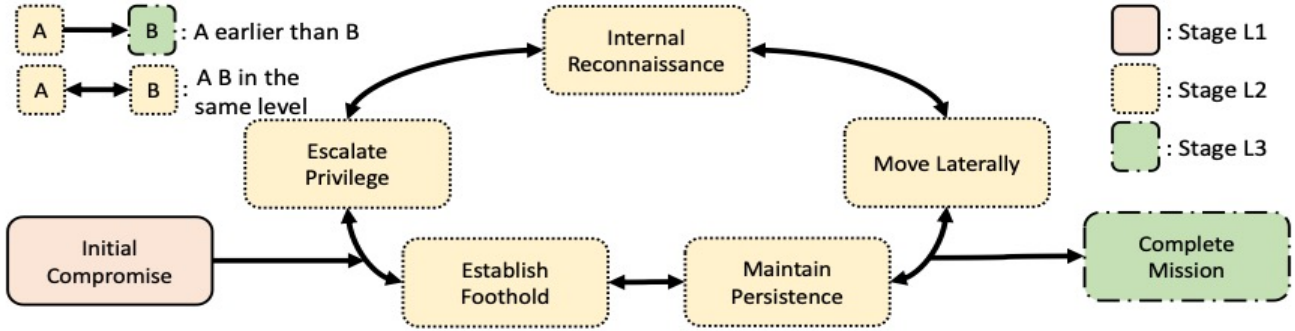
- **Solution2:** We design CTI Knowledge Database (CKD) to enable semantic lifting and *accelerated* mapping when using gIoCs to detect system events.

- **Accelerating *ProvQ***:
  - We propose a two-stage search method to accelerate the speed of ***ProvQ***
  - **Offline stage**: *Cluster* all gIoCs using Mean-Shift algorithm on their embedding vectors
  - **Online stage**: For event field $e.y$,
    - we first find the closest gIoC cluster
    - then compute similarity only within that cluster
  - **Result**: Avoids over 70% unnecessary comparisons while preserving recall.



C3: How to Detect Multi-stage APTs

- **Solution3:** Novel attack reasoning method based on the stages of the APT Lifecycle
- **Reasoning Model: A relaxed APT Lifecycle model**
  - **Lifecycle Completeness**: The attack must have initial compromise step and one of the steps in stage L2.
  - **Lifecycle Temporal Order**: The attack step in later stage must happen later than the step in previous stage.
  - We design a **one-to-limited mapping** optimization to map anomalous nodes to reasoning model.



Evaluation: Protocol

- **Close-world Datasets**
  - DARPA TC dataset
  - In-lab Arena dataset
  - NewlySim dataset[1] : contains attack behaviors that happened after CTI reports are released.
- **Open-world Dataset**
  - An open-world dataset across over 180 endpoints
- **Metrics**
  - graph-level precision/recall, node-level precision/recall
- **Baselines**
  - Detection baselines: HOLMES, AIRTAG, NodLink, KAIROS, EXTRACTOR+POIROT
  - CTI extraction baselines: TTPDrill, LADDER, EXTRACTOR

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

[1]https://github.com/myh0301/KNOWHOW

Evaluation: Effectiveness

北京大学 PEKING UNIVERSITY

- **Graph-level accuracy:**
  - Detects **all the attacks** and only reports **0 false positives**.
- **Node-level accuracy**:
  - Node-level precision: KnowHow reduces up to 90% more node-level false positives compared to existing detection baselines.
  - Node-level recall: KnowHow achieves almost perfect recall.

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

Detection Baselines	Dataset									
	THEIA		TRACE		In-Lab Arena		NewlySim		Open-World	
	P	R	P	R	P	R	P	R	P	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	Dataset									
	THEIA		TRACE		In-Lab Arena		NewlySim		Open-World	
	P	R	P	R	P	R	P	R	P	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**

Evaluation: Effectiveness

- **Effectiveness of gIoC:**
  - KnowHow can identify attack behaviors in different datasets using gIoC, while failing when using results from the other SOTA CTI extraction.
    - Because, using the event semantic lifting method, 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.

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**

Evaluation: Technique Label Accuracy

北京大学 PEKING UNIVERSITY

- Our datasets cover 643 attack actions spanning 12 tactics and 65 techniques.
- KnowHow accurately labels 559 of all 643 attack actions, accounting for 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, showing the effectiveness of gIoCs.

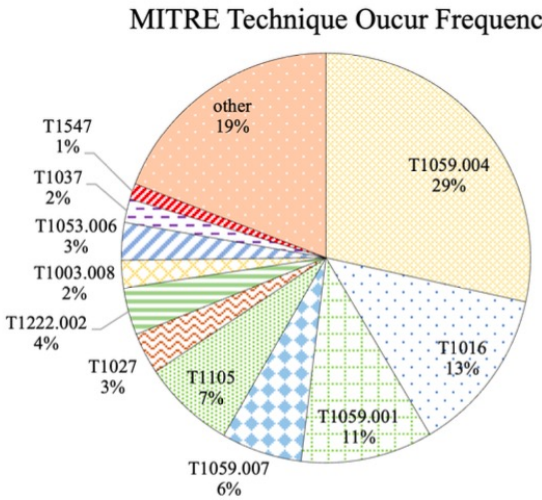


Fig. 5: Top 10 frequent techniques in our experiments.

Evaluation: Efficiency

- **Throughput:** how many system events can be processed per second (eps)
  - KnowHow is comparable to SOTA online detection systems.
  - KnowHow is capable of monitoring over **180** hosts online in our real-world Open-World Datatset.

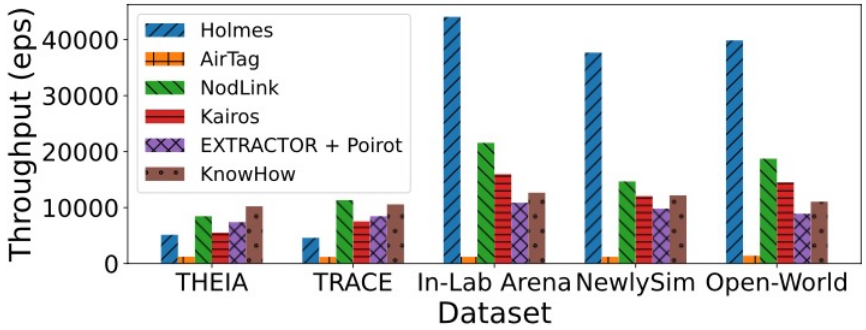


Fig. 6: Working throughput among different frameworks. TTPDRILL and LADDER have the same throughput as KNOWHOW.

Evaluation: Robustness

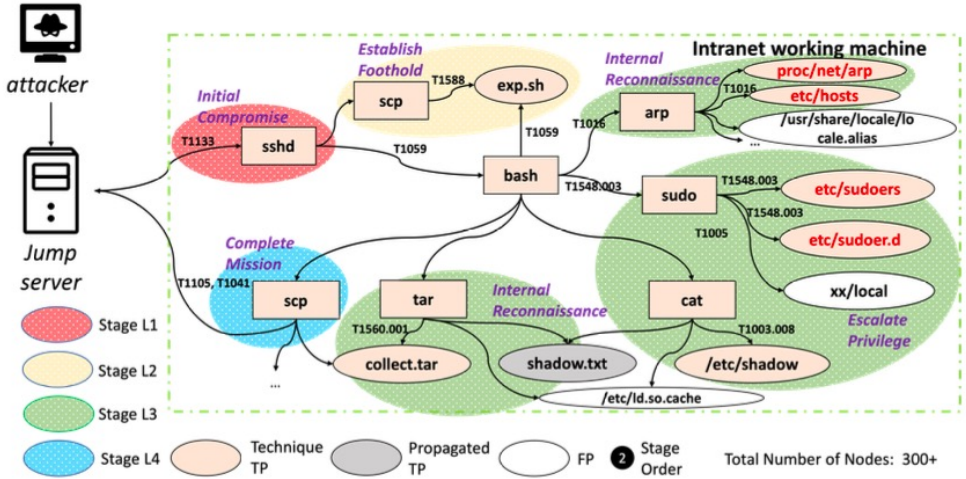
北京大学 PEKING UNIVERSITY

- **Mimicry Attacks**
  - We constructed a mimicry dataset, Mimic-Prov, following the steps described in the previous work, and evaluated KnowHow on it.
  - KnowHow can successfully detect all attack events at the graph level across all these mimicry insertion ratios within the Mimic-Prov dataset.
- **Incomplete Attacks**
  - We simulate incomplete attacks by removing attack steps from the original attack sequences in the In-lab Arena dataset.
  - Both graph- and node-level detection performance of KnowHow remain stable and effective, except for a limited increase in node-level false positives, when the attack integrity ratio is 0.80 or lower.

Evaluation: Case Study on Unseen Attacks

北京大学 PEKING UNIVERSITY

- KnowHow successfully detects two unseen multi-stage APT attacks in NewlySim dataset.
- These attacks exploited newly disclosed CVEs that were not yet included in any CTI report used to build KnowHow's CKD, and occurred after those CTIs, ensuring they were truly unseen.



KNOWHOW APT Attack Report

Incident Overview

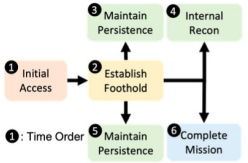
Date: 2024-05-10
Reported By: KNOWHOW
Incident ID: th-008

Executive Summary

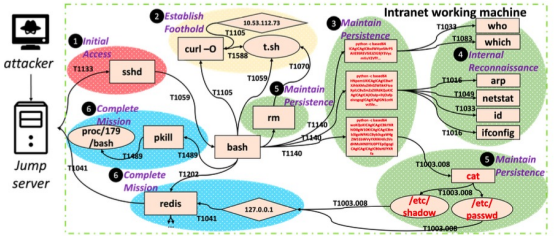
On 2024-05-10, KNOWHOW identified a sophisticated cyberattack originating from an internal IP address (192.0.1.15). The attack involved multiple stages including initial access, establishing a foothold, maintaining persistence, internal reconnaissance, and completing the mission. The attacker utilized various techniques and tools to achieve their objectives.

Attack Overview

1. **Attack Lifecycle Alert**



2. **Attack Provenance Graph**



Conclusion

北京大学 PEKING UNIVERSITY

- 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.
- Our experiments show that KnowHow outperforms existing baselines in terms of both **accuracy** and **interpretability** while maintaining a **comparable throughput**. Moreover, KnowHow can achieve great **robustness** to attacks stemming from unknown vulnerabilities and existing mimicry attacks.

https://github.com/myh0301/KNOWHOW

mengyuhan@pku.edu.cn