Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Commander: A robust cross-machine multi-phase Advanced Persistent Threat detector via provenance analytics

Qi Liu *, Kaibin Bao, Veit Hagenmeyer

Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, 76344, Germany

ARTICLE INFO

Keywords:
Advanced Persistent Threat detection
Data provenance analysis
Digital forensics
Active Directory
Industrial-sector organization security

ABSTRACT

Intrusion detection systems (IDS) have traditionally focused on identifying malicious behaviors caused by malware undertaking a series of suspicious activities within a short time. Facing Advanced Persistent Threat (APT) actors employing the so-called low-and-slow strategy, defenders are often blindsided by the poor performance of these IDS. Provenance-based IDS (PIDS) emerged as a promising solution for reducing false alerts, detecting true attacks, and facilitating attack investigation, by causally linking and contextualizing indicative system activities in provenance graphs. However, most existing PIDS can detect neither multi-phase nor cross-machine APT attacks, enabled by persistence and lateral movement techniques, respectively. In the present work, we propose a new PIDS called Commander, which is, to our knowledge, the first system capable of detecting cross-machine multi-phase APT attacks. Further, Commander targets several evasion attacks that can bypass existing PIDS, making it more robust. In addition, Commander can perform whole network tracing for cross-machine multi-phase APT attacks across an industrial-sector organization, for which we additionally develop parsers for system logs of popular industrial controllers. We also develop detection rules with a reference to MITRE's knowledge base for industrial control systems. Our evaluations show that Commander accurately detects attacks, outperforms existing detection systems, and delivers succinct and insightful attack graphs.

1. Introduction

Advanced Persistent Threat (APT) actors are well-resourced attackers specialized in launching covert cyberattacks on computer systems to which they maintain unauthorized access for an extended period of time. The main objectives of APT actors are cyber espionage and impairing physical processes when targeting industrial-sector organizations. Once achieved initial access on a machine in a target organization, attackers typically find ways to pivot to other domain-joined machines for more sensitive data, or means to cause destructive damages to cyber–physical systems. Stolen credentials are the most popular way for attackers to access further machines in the network: Recent CrowdStrike studies [1–4] show that 80% of modern cyberattacks are identity-driven, and Dragos's findings [5] suggest that nearly all attackers leverage stolen credentials for lateral movement.

To remain undetected, APT actors routinely employ the so-called low-and-slow strategy. Relying on Living-Off-the-Land Binaries (LOLBins) [6,7], i.e., system pre-installed programs, helps attackers stay *low*, as legacy intrusion detection systems (IDS) often consider system activities associated with these programs benign, due to their constant presence during normal operations. In response to this, current IDS incorporate TTP (Tactics, Techniques, Procedures) detection rules with a reference to the MITRE ATT&CK Matrix [8]. This matrix provides a high-level summary of attack tactics and techniques based on real-world observations, including execution of LOLBins. It is maintained with contributions from leading security vendors.

Yet, simply applying TTP detection rules results in a deluge of security alerts, and hence leads to the alert fatigue problem, as the majority of these alerts are in fact false positives [9]. Provenance-based IDS (PIDS) [10–24] emerged as a promising approach for reducing false alerts and accelerating attack investigation, by parsing system audit logs into provenance graphs and thus presenting historical context of system activities. Starting from an *indicative* system activity associated with a security alert, backward & forward tracing in the provenance graph can uncover more indicative, causally related system activities. By doing so, TTP security alerts can be linked, contextualized, and then prioritized for further investigation. The underlying assumption of PIDS is that, when a series of TTP alerts can be causally linked into an attack graph reassembling a Cyber Kill Chain, they are more likely caused by attackers than those alerts unrelated to each other.

* Corresponding author.
E-mail address: qi.liu@kit.edu (Q. Liu).

https://doi.org/10.1016/j.jisa.2025.104057

Available online 29 April 2025

However, most existing PIDS can detect neither multi-phase nor cross-machine APT attacks, enabled by persistence and lateral movement techniques, respectively. Multi-phase attack detection necessitates an understanding of persistence attacks' nature, and cross-machine tracing faces significant challenges due to the notorious dependency explosion problem [25,26]. In order to evade PIDS, APT actors resort to persistence techniques, which enable them to disassemble an attack chain into multiple phases and stay *slow*. This makes it infeasible for PIDS without persistence awareness [10–24] to causally link those system activities (and the TTP alerts) from the same attacker, as they are located in fragmented, seemingly unrelated graphs. Cpd [25] is the first specialized persistence detector, which can accurately identify attackers' use of persistence techniques from system logs, and hence recouple those disassembled, but actually related graphs.

Most PIDS are restricted to intra-machine provenance tracing, even though cross-machine activities are omnipresent in real-world APT attacks [27]. This, again, limits PIDS' ability to link attack steps of the same APT actor, and thus reveal the scale of the attacker's traversal inside an organization, crucial for accurate attack detection and remediation. To overcome this, Hades [26] presents a concept called *logon session-based execution partitioning and tracing*, which tackles the cross-machine dependency explosion problem, and hence enables accurate and efficient causality-based cross-machine tracing in enterprise networks.

However, our analysis of APT threat reports linked in the MITRE ATT&CK knowledge base [28] yields the finding that Hades, as the first PIDS capable of accurate cross-machine tracing, is susceptible to several evasive attack tactics/techniques routinely employed by APT actors: persistence [29], session hijacking [30], and port forwarding [31]. Whereas port forwarding attacks can directly interrupt Hades's cross-machine tracing, both persistence and session hijacking attacks hinder accurate intra-machine tracing, on which cross-machine tracing hinges. The impacts of these evasion techniques on Hades are discussed in Section 4.2 in detail.

Recognizing the weaknesses of Hades, we designed Commander (**Cr**O**ss-M**achine **M**ulti-phase **A**dva**N**ced persistent threat **DE**tecto**R**). Commander integrates Cpd with Hades, and incorporates another two specialized detectors: one for session hijacking and one for port forwarding. Commander features a three-stage approach for efficient and accurate attack detection and graph reconstruction. The first stage of Commander serves as preliminary detection, and accommodates four specialized detectors running in parallel, i.e., an authentication anomaly detector, a cyber persistence detector, a session hijacking detector and a port forwarding detector. The authentication anomaly detector identifies anomalies indicating identity-driven attacks, and produces an initial high-level attack graph for each anomaly, which is passed to Commander's second stage, i.e., whole network tracing. The other three specialized detectors make complementary contributions to ensuring *robust and correct whole network tracing*, by delivering critical information to guide and adjust the tracing process.

The objective of whole network tracing is to separate system activities from different identities, and causally link system activities of the same identity, be it the attacker or a genuine user, on each involved machine across the entire organization. *Correct and complete tracing* in this stage is vitally important for the attack graph ranking in Commander's final stage, in which graphs are ranked by threat score. The threat score is determined by the set of indicative attack steps/security alerts found in an attack graph. Thus, incorrect or incomplete tracing could let true attack graphs be ranked lower than false-positive ones.

Further, Commander is designed as a PIDS for detecting cross-machineattacks in industrial-sector organizations consisting of an enterprise network and an industrial control system (ICS), which are typically in two separate domains. That is, Commander is able to trace across domains, given that the enterprise network and the ICS network are often connected via a gateway server. Besides, we developed parsers for system logs from two popular industrial controllers in energy systems, i.e., Siemens SIPROTEC [32] and Hitachi Energy RTU500 [33]. We also developed TTP detection rules with a reference to the MITRE ATT&CK Matrix for ICS [34] for these controllers. Commander is capable of accurately attributing the activities on these controllers to the true identity behind those actions, even if the access originates from the enterprise network.

We evaluate Commander on the AVIATOR dataset [27], which is based on the MITRE attack emulation plans [35] and consists of the most complex and authentic APT attacks comparing to other datasets. The AVIATOR dataset includes two extended MITRE emulation plan: one for Sandworm [36] and one for Oilrig [37]. These extended emulation plans incorporate an ICS infrastructure and attack steps on it, making AVIATOR the only APT dataset for investigating attacks spanning from an enterprise network to an ICS network. Our evaluations on the AVIATOR dataset show that Commander accurately detects attacks, outperforms open-source detection rules and a commercial detection system, and produces succinct and informative attack graphs.

The main contributions of this paper are as follows:

- We introduce several attack techniques for evading Hades, the first PIDS capable of accurate cross-machine tracing.
- We present two specialized detectors: one for session hijacking attacks and one for port forwarding attacks.
- We propose Commander, a modularized provenance-based detection system that integrates Cpd and the above two specialized detectors with Hades to counter evasion attacks.
- We develop parsers and TTP detection rules for two popular industrial controllers.
- We demonstrate Commander's ability to perform robust whole network tracing, including attribution of malicious activities conducted on industrial controllers.

2. Background

An industrial-sector organization typically has an enterprise/IT network and an ICS/OT (Operational Technology) network that are separated from each other [38]. Communications between these networks are often enabled by a gateway/jump server. To reduce the risk, the enterprise network and the ICS network have separate identity and access management (IAM) systems [5,39]. IAM systems are ubiquitous in networks of large organizations. (Microsoft) Active Directory is the most popular IAM system, with more than two-thirds of the market share and 90% of Fortune 1000 companies relying on it [4,40,41]. Active Directory is popular not only in IT networks, but also in OT networks. In fact, OT vendors, e.g., Schneider Electric, shifted their OT products' OS from Unix-based ones to Windows NT long time ago in order to better make use of Active Directory functionalities like Kerberos Authentication and Group Managed Service Accounts [42]. Although our implementation and evaluation of Commander rest on Active Directory, the concept for robust whole network tracing and efficient attack detection introduced in Section 6 is transferable to networks employing another IAM system.

As shown in [43], during an attack against an industrial-sector organization, a complete attack path includes *multiple cross-machine movements* necessary for APT attackers to achieve their final objectives. Often, attackers first gain an initial foothold on a machine in the IT network through phishing or binary exploitation. From the initially accessed machine, attackers routinely perform Active Directory discovery to locate more machines in the network. In comparison to network scanning using third-party tools like `nmap` [44], Active Directory discovery techniques leverage native Windows programs like `setspn` [45] to avoid noisy network activities and hence bypass network IDS. Moreover, attackers commonly steal credentials, e.g., cached authentication tickets, password hashes, or even clear text passwords, on each accessed machine. Leveraging stolen credentials is the most popular way for attackers to *move across a target network* [1–5]. The pivotal access on a gateway server enables attackers to reach machines

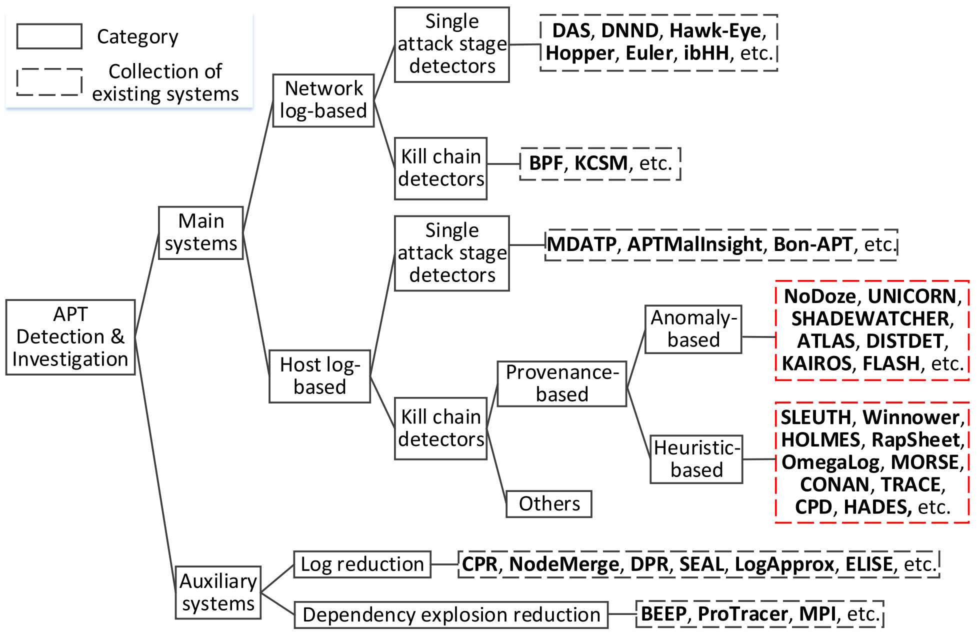
Category
Collection of existing systems
APT Detection & Investigation
Main systems
Network log-based
Single attack stage detectors
DAS, DNND, Hawk-Eye, Hopper, Euler, ibHH, etc.
Kill chain detectors
BPF, KCSM, etc.
Host log-based
Single attack stage detectors
MDATP, APTMalInsight, Bon-APT, etc.
Kill chain detectors
Provenance-based
Anomaly-based
NoDoze, UNICORN, SHADEWATCHER, ATLAS, DISTDET, KAIROS, FLASH, etc.
Heuristic-based
SLEUTH, Winnower, HOLMES, RapSheet, OmegaLog, MORSE, CONAN, TRACE, CPD, HADES, etc.
Others
Auxiliary systems
Log reduction
CPR, NodeMerge, DPR, SEAL, LogApprox, ELISE, etc.
Dependency explosion reduction
BEEP, ProTracer, MPI, etc.

Fig. 1. Taxonomy of APT detection and investigation systems.

in the OT network. As the objective of attacks against industrial-sector organizations is impairing physical processes, the final target of the attackers is typically Engineering Workstations, through which field controllers are programmed and configurated, and field controllers, like PLC (Programmable Logic Controller), RTU (Remote Terminal Unit) and IED (Intelligent Electronic Device), through which physical processes are controlled.

3. Related work

In Fig. 1, we present a taxonomy of APT detection and investigation systems, which we categorize into main systems and auxiliary systems. Systems designed directly for APT detection and investigation are considered as main systems, while systems that serve to assist the detection or investigation process, but do not perform detection themselves are considered as auxiliary systems. We further categorize main systems into network log-based and host log-based systems, depending on the data source. In addition, we classify a detection system as single attack stage detector, if it is designed for detecting a specific attack stage, e.g., data exfiltration. In contrast, kill chain detectors have a broader detection scope: a Cyber Kill Chain consisting of several attack stages.

We find that network log-based APT detection and investigation systems largely fall into the category of single attack stage detector, whereas host log-based APT detection and investigation systems mostly aim to detect an entire Cyber Kill Chain. For instance, Hopper [46], Euler [47,48] target lateral movement detection using network logs. There are also network log-based single attack stage detectors for detecting phishing emails/initial access, e.g., DAS [49], command and control (C2) activities, e.g., DNND [50], Hawk-Eye [51], and data exfiltration, e.g., ibHH [52].

It is worth noting, that although our system Commander's cross-machine APT detection involves detection of lateral movement, it is more than a lateral movement detector. That is, like Hades [26], Commander performs whole-network provenance tracing to accurately and comprehensively unravel attackers' traversal inside a network, and hence to reveal detailed malicious system activities of an attacker on *each* involved machine (see Fig. 7 and 12). On the contrary, existing lateral movement detectors like Hopper [46], Euler [47,48] leverage network logs to infer simply the presence of the lateral movement attack stage. They can neither perform accurate provenance tracing, nor assist attack investigation by providing detailed attack graphs, in which the lateral movement attack stage is associated with other attack stages of a Cyber Kill Chain.

There are only a few network log-based kill chain detectors: BPF [53], KCSM [54]. By construction, network log-based kill chain detectors can only detect a partial Cyber Kill Chain, as some attack stages, e.g., persistence, credential access, and privilege escalation, can be detected only with host logs. Further, attack graphs output by such detectors are very coarse granular, in which nodes denote individual machines and edges denote suspected attack stages. To find the root cause for an attack or thoroughly assess the attack consequences, security analysts would still need to manually inspect host logs from each involved machine.

We find that the majority of host log-based APT detection systems leverage data provenance analysis. Only a few detectors do not fall into the category of provenance-based intrusion detection systems (PIDS), e.g., MDATP [55] for detecting credential access attack stage, and APTMalInsight [56] and Bon-APT [57] for detecting (APT malware) execution attack stage. Comparing to network log-based kill chain detectors, host log-based kill chain detectors operate at a finer granularity. Attack graphs produced by PIDS expose malicious behaviors at the system activity level, e.g, malicious process creation or suspicious file access, rather than at machine level, e.g., suspicious network interaction.

A recent study [58] shows that PIDS are considered more effective than other detection systems, mostly due to better interpretability of detection results. Context-rich provenance graphs are leveraged in PIDS to accelerate attack detection & investigation process. PIDS are often categorized into anomaly-based systems and heuristic-based systems. Anomaly/learning-based PIDS model/learn benign behavior from provenance graphs, and detect deviations from it as anomalies. Heuristic-based PIDS embed expert knowledge and developed heuristics into detection algorithms. Heuristics are often derived from key insights identified during analysis of past real-world APT attack campaigns. PIDS are included in the red, dashed bounding boxes in Fig. 1, and further discussed in Section 3.1 in detail due to their relevance to our system Commander.

In order to enhance PIDS' practicality, a number of auxiliary systems have been proposed. For instance, various dedicated log reduction schemes are developed: CPR [59], NodeMerge [60], DPR [61], LogApprox [62], SEAL [63], ELISE [64]. They serve to reduce the volume of

Table 1
Comparison of PIDS.

Name	Methods	Innovative techniques	Tracing optim.	Attack scope	Target envir.	Detect. result	Investigation assistance	Resp. time
SLEUTH [10], 2017	Heuristics	Node Tagging, Tag Propagation	✓	Simp. APT	Indiv. hosts	Alerts	Succinct attack graphs	short
Winnower [11], 2018	Heuristics	Graph Grammar Models	✗	Non-APT	Distr. appli.	Alerts	Succinct attack graphs	short
NoDoze [13], 2019	Rules, Anomaly	Behavioral Execution Partitioning	✓	Simp. APT	Indiv. hosts	Ranked alerts	Succinct attack graphs	short
HOLMES [12], 2019	Rules, Heuristics	Information Flow Correlation	✗	Simp. APT	Indiv. hosts	Ranked alerts	High-level scen. graphs	short
RapSheet [14], 2020	Rules, Heuristics	Graph-Based Scoring Scheme	✓	Simp. APT	Indiv. hosts	Ranked alerts	Succinct attack graphs	short
OmegaLog [16], 2020	Heuristics	Log Unification, Peephole Concretization	✗	Non-APT	Indiv. hosts	Alerts	Succinct attack graphs	short
UNICORN [19], 2020	ML, Anomaly	Sketch-based Provenance Encoding	✗	Simp. APT	Indiv. hosts	Alerts	None	long
MORSE [15], 2020	Heuristics	Tag Attenuation, Tag Decay	✓	Simp. APT	Indiv. hosts	Alerts	Succinct attack graphs	short
CONAN [17], 2020	Heuristics	State-based Detection Framework	✓	Simp. APT	Indiv. hosts	Alerts	Succinct attack graphs	short
TRACE [18], 2021	Heuristics	Unit-based Selective Instrumentation	✓	Simp. APT	Enter. netw.	Alerts	Succinct attack graphs	short
ATLAS [21], 2021	ML, Anomaly	Sequence-based Model Learning	✗	Simp. APT	Small netw.	Alerts	Succinct attack graphs	long
SHADEWA. [20], 2022	ML, Anomaly	Recomm.-guided Log Analysis	✗	Simp. APT	Indiv. hosts	Alerts	None	N/A
DISTDET [22], 2023	Anomaly	Alarm Deduplication & Semantic Aggregation	✗	Simp. APT	Small netw.	Ranked alerts	Succinct attack graphs	N/A
KAIROS [23], 2024	ML, Anomaly	Encoder–Decoder-based Anomaly Quantification	✗	Simp. APT	Indiv. hosts	Alerts	Verbose attack graphs	long
FLASH [24], 2024	ML, Anomaly	Embedding Recycling Database	✗	Simp. APT	Indiv. hosts	Ranked alerts	Verbose attack graphs	long
CPD [25], 2024	Rules, Heuristics	Pseudo Edges, Expert-guided Edges	✓	MP APT	Indiv. hosts	Ranked alerts	Succinct attack graphs	short
HADES [26], 2024	Rules, Heuristics	Logon Session-based Execution Partitioning	✓	CM APT	Enter. netw.	Ranked alerts	Succinct attack graphs	short
COMMANDER	Rules, Heuristics	Anti-evasion techniques, Robust Tracing	✓	CM MP APT	IS organ.	Ranked alerts	Succinct attack graphs	short

Note: appli. = applications, APT = Advanced Persistent Threat, CM = Cross-Machine, Detect. = Detection, Distr. = Distributed, Enter. = Enterprise, envir. = Environment, Indiv. = Individual, IS = Industrial-sector, MP = Multi-Phase, ML = Machine Learning, N/A = not applicable/available, netw. = networks, optim. = optimization, organ. = organizations, Recomm. = Recommendation, Resp. = Response, scen. = scenario, SHADEW. = SHADEWATCHER, Simp. = Simplified.

logs required for APT detection and investigation, and hence alleviate the long-term log storage problem.

The large volume of host logs also deteriorate the dependency explosion problem. Dependency explosion happens for long-running processes, in which each input event is conservatively considered causally responsible for all subsequent output events, and vice versa, resulting in excessive amounts of false dependencies and formidably dense provenance graphs [65,66]. Program execution partitioning techniques introduced in BEEP [65] and ProTracer [67] are the first proposed approaches for combating (intra-machine) dependency explosion. MPI [68] improves these techniques by introducing a semantic-aware program annotation and instrumentation technique.

3.1. Comparison of PIDS

Table 1 provides a comparison of PIDS. SLEUTH [10] is a PIDS that introduces node tagging and tag propagation based on heuristics for attack graph reconstruction. It is designed for detecting and investigating simplified APT attacks, and focuses on attacks on individual hosts in isolation. *We consider an attack scenario as simplified APT attack, if the entire scenario is restricted to a single host and does not contain an effective persistence technique, as apposed to real-world APT attacks.* SLEUTH produces alerts as detection result, and succint attack graphs to support investigation by security analysts. Evaluation on a DARPA dataset [69] shows that SLEUTH has a short response time, i.e., within a few seconds or minutes. Winnower [11] is a heuristic-based PIDS that adapts graph grammar techniques to detect attacks on distributed applications, instead of whole systems. It is evaluated on a unpublished, self-developed dataset.

NoDoze [13] is an anomaly-based PIDS that leverages detection rules from commercial security products to generate initial alerts. Then it performs tracing on these alerts with system logs, and employs behavioral execution partitioning for enhanced tracing efficiency. With a path-based anomaly scoring algorithm, NoDoze is the first PIDS producing *ranked* alerts. Ranked alerts are more useful in practice, as security analysts can prioritize the highest ranked alerts for investigation due to limited time.

HOLMES [12], as a heuristic-based PIDS, proposes the notion high-level scenario graph (HSG), which is a summarized description of an attack history. It first produces initial alerts through self-developed detection rules, then performs tracing via correlation between suspicious information flows, and outputs HSG at the end. RapSheet [14] also combines detection rules with heuristics for APT detection and investigation. Instead of relying on self-developed detection rules, RapSheet employs a commercial EDR (Endpoint Detection and Response) product for the initial alerts. Unlike NoDoze, RapSheet adopts a graph-based scoring scheme for ranking final alerts. RapSheet also proposes techniques to perform graph reduction, which makes produced attack graphs more succinct. OmegaLog [16] presents the first approach realizing log unification, in which system logs and application logs are combined to enhance tracing efficiency, and to produce succinct attack graphs with enriched context.

UNICORN [19] is the first Machine Learning (ML)-based anomaly detector via runtime provenance analytics. It uses a sketch-based, time-weighted provenance encoding scheme to summarize provenance graphs for training and detection. However, unlike aforementioned PIDS, UNICORN does not assist attack investigation by producing attack graphs. Rather, it classifies an entire provenance graph as benign or malicious. UNICORN is evaluated on StreamSpot [70] and DARPA [69] datasets.

MORSE [15] improves on SLEUTH [10] by introducing two key techniques called tag attenuation and tag decay, which help to combat the dependency explosion problem for long-running attacks. Note that SLEUTH [10] can create succinct attack graphs only for short-lived attacks. For long-running attacks, SLEUTH would produce dense attack graphs containing many benign system activities. CONAN [17] proposes a state-based detection framework, in which each system entity, e.g., process and file, is represented as an finite state automata (FSA)-like structure. Combined with suspicious code execution tracking, system entities are assigned with benign or malicious states.

TRACE [18] presents a network connection-based cross-machine tracing approach, and proposes a technique called unit-based selective instrumentation to reduce dependence explosion during tracing. TRACE's cross-machine tracing is, however, restricted to system activities of individual instrumented programs, rather than an entire machine's system activities. In addition, such invasive program instrumentation is often too costly to be adopted in practice [16]. Although TRACE's target environment is enterprise networks, it is evaluated only on DARPA-E3 [69] and DARPA-E5 [71] datasets, which contain simplified APT attack scenarios not in an enterprise setting [27]. Besides, TRACE's design is oblivious to the domain context of an enterprise environment, and insufficient for combating cross-machine dependency explosion [26].

ATLAS [21] is a ML-based PIDS that leverages sequence-based model learning for APT attack graph reconstruction. It is applied to detect attacks on individual hosts or small networks consisting of only two hosts. SHADEWATCHER [20] is another ML-based PIDS, which combines recommendation principles with provenance analysis. Like UNICORN, SHADEWATCHER does not support security analysts in attack investigation. DISTDET [22] is an anomaly-based PIDS focusing on cost-effective deployment of PIDS in practice. It introduces alarm deduplication & semantic aggregation techniques to filter false alarms.

KAIROS [23] is an anomaly-based PIDS that adopts a Graph Neural Network (GNN)-based encoder–decoder architecture for quantifying the degree of anomaly of each system event. It improves on UNICORN, and performs attack detection at a finer granularity. That is, instead of taking an entire provenance graph as input, it takes system events as input, splits the entire time line into many time windows, and classifies each time window as benign or malicious. Yet, this approach produces comparatively dense attack graphs mixing malicious and benign system activities. FLASH [24], as another GNN-based PIDS, performs classification at an even finer granularity level: the node level. In addition, FLASH presents an embedding recycling database to store the node embeddings from the training phase for improved computational efficiency. Nevertheless, attack graphs produced by FLASH still contain a high percentage of benign system activities, complicating attack investigation [25]. Further ML-based PIDS, e.g., APT-KGL [72], WATSON [73], THREATRACE [74], DEPIMPACT [75], PROGRAPHER [76], MAGIC [77], are not discussed, as they all, like the state-of-the-art ML-based PIDS KAIROS and FLASH, have a limited attack scope: simplified APT attacks.

We argue that, most prior PIDS, both anomaly-based [13,19–24] and heuristics-based ones [10–12,14–18], would fail at detecting not only multi-phase APT attacks but also cross-machine APT attacks. First, prior PIDS fall short of detecting persistence, due to failure to recognize that persistence attack is always a two-part act [25]. By leveraging persistence, APT attackers break down an entire attack chain into multiple phases, which in turn makes these PIDS create rather disintegrated, seemingly unrelated attack graphs. Second, prior PIDS are, *by construction*, limited to intra-machine tracing, and unable to causally associate an attacker's activities across machines in an organization's network(s), as a result of an obliviousness to the network context [26].

Unlike these systems, Cpd excels in detecting persistence techniques by combining advanced detection rules with pseudo-edges and expert-guided edges. This unique approach enables Cpd to effectively identify and investigate persistence threats, bridging the gaps that other PIDS fail to address. Hades is the first PIDS capable of performing truly causality-based cross-machine tracing, capitalizing on a critical yet previously undiscovered information: logon session ID. However, we find that Hades is vulnerable to several evasion attack techniques, including persistence. By integrating several preliminary detectors, targeting each evasion attack technique individually, with Hades, our system Commander's whole network tracing achieves a considerably higher level of robustness. Moreover, Commander is the first PIDS that incorporates a module for provenance tracing to industrial controllers.

Like HOLMES [12] and RapSheet [14], Commander leverages the synergy of detection rules and heuristics for APT detection and investigation. These detection rules match indicative system activities, and are used to initiate the tracing process or calculate a threat score, which in turn serves to discover related indicative system activities, or rank an attack graph, respectively. But, unlike RapSheet [14], we do not rely on any commercial security product. Comparing to self-developed detection rules in HOLMES [12], self-developed rules in Commander are readily integrable into open-source rule repositories like Sigma [78].

We consider that a system offers tracing optimization, if specific techniques are proposed to improve tracing efficiency, completeness, or correctness. While most prior systems [10,13–16,18] optimize the tracing process all by enhancing the efficiency, e.g., through optimized path selection or instrumentation, Cpd and Hades optimize the tracing process by improving efficiency and completeness, and Commander also improves tracing correctness.

We stress that each compared study in Table 1 has its own strengths, in which it solves a specific problem with the proposed innovative techniques. For instance, SLEUTH [10] and MORSE [15] present tag-based key techniques to combat the dependency explosion problem [65,66]. Winnower [11] specifically targets attacks on distributed applications with its graph grammar models. NoDoze's behavior execution partitioning [13] can enhance tracing efficiency. HOLMES's information flow correlation produces simplistic high-level scenario graphs, which are easily understandable without extensive security background knowledge. RapSheet's graph-based scoring scheme aims to produce more reasonably ranked alerts and help security analysts focus on more likely serious incidents. OmegaLog [16] presents a log unification technique to enhance tracing efficiency, and more properly leverage existing logs. CONAN [17] demonstrates yet another technique to increase tracing efficiency, i.e., a state-based detection framework. TRACE [18] addresses the dependency explosion problem with another key technique called unit-based selective instrumentation. DISTDET's alarm deduplication technique [22] can reduce deployment cost of PIDS in practice.

The main strength of ML-based systems UNICORN [19], ATLAS [21], THREATRACE [74], KAIROS [23] and FLASH [24] is that they require less security domain-specific expert knowledge from researchers

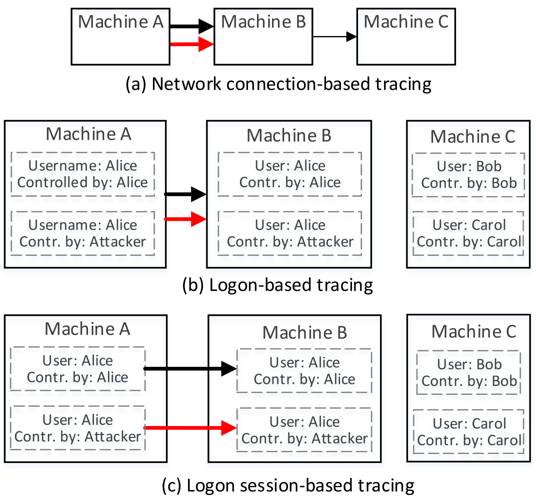

Fig. 2. Comparison of cross-machine tracing approaches. Whereas a thin black edge denotes a network connection, a thick black edge denotes a logon by a genuine user and a thick red edge denotes a logon by an attacker. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

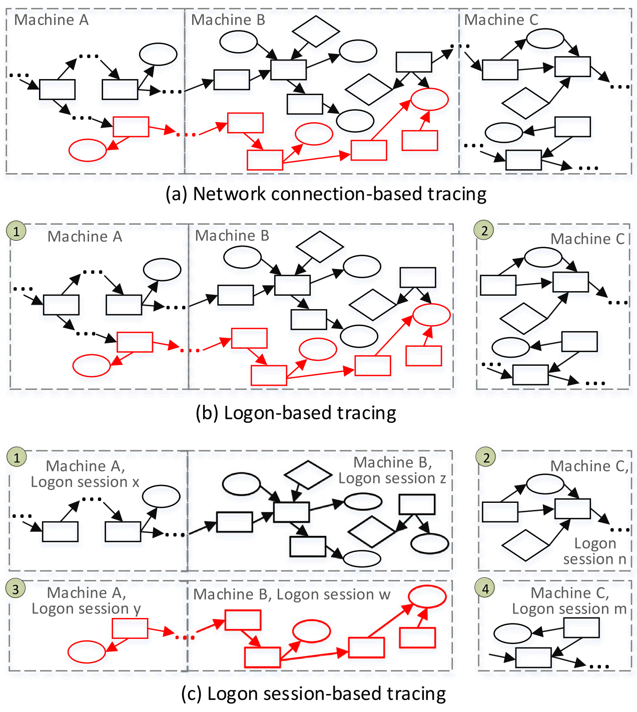
Machine A
Machine B
Machine C
(a) Network connection-based tracing
(b) Logon-based tracing
Machine A, Logon session x
Machine B, Logon session z
Machine C, Logon session n
Machine A, Logon session y
Machine B, Logon session w
Machine C, Logon session m
(c) Logon session-based tracing

Fig. 3. Comparison of results from different cross-machine tracing approaches. Whereas black denotes system activities originated from genuine users, red denotes system activities caused by attackers. If a tracing approach returns more than one graph, these graphs are numbered and considered as independent for further investigation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to develop and improve. However, the detection logic and results are comparatively less explainable. The common weakness of all compared studies is that only simplified APT attacks are addressed, rather than cross-machine multi-phase APT attacks in real world. To detect these more complex attacks, dedicated techniques are required, which are incorporated in our system COMMANDER.

Nevertheless, key techniques in the compared studies can be complementary to our key techniques for efficient and accurate APT detection and investigation. For instance, tag-based techniques from SLEUTH [10] and MORSE [15], and the log unification technique from OmegaLog [16] could be integrated in our system COMMANDER for further reducing dependency explosion during tracing. The main weakness of our system COMMANDER, like other heuristic-based approaches, lies in the fact that it requires expert knowledge in security domain to develop and maintain.

4. Motivating examples

4.1. HADES's *cross-machine tracing*

HADES [26] is the first PIDS capable of performing accurate causality-based cross-machine tracing, by tackling the cross-machine dependency explosion problem with a method called *logon session-based execution partitioning and tracing*. Whereas intra-machine dependency explosion problem occurs for long-running processes [65,66], in which each input is conservatively considered causally responsible for all subsequent outputs, and vice versa, cross-machine dependency explosion happens if cross-machine edges are created merely on a network connection basis.

Fig. 2 illustrates the difference between three cross-machine tracing approaches. Network connection-based tracing, which naively connects two intra-machine provenance graphs whenever there is a network connection between the two machines, as shown in Fig. 2(a), would lead to a dense graph indiscriminately connecting all system activities as depicted in Fig. 3(a). It is worth noting that most provenance-based detection systems mentioned in Section 3 can produce attack graphs including network activities/network socket accesses. However, this can only suggest that they would be capable of network connection-based cross-machine tracing, which is highly inaccurate and associated with prohibitively high cost for further investigation.

Logon-based tracing in Fig. 2(b) reduces false dependencies by considering only logon-initiated network connections, resulting in less dense graphs in Fig. 3(b). However, in the prevalence of identity theft, it is infeasible to discern the true identity behind each logon. That is, a provenance graph produced from logon-based tracing can still contain system activities of multiple identities. In contrast, logon session-based tracing narrows down system activities deemed as relevant from an entire machine to just one logon session, as shown in Fig. 2(c), truly linking system activities belonging to the same identity, and separating system activities unrelated to each other into individual provenance graphs displayed in Fig. 3(c).

While intuitive, logon session-based tracing faces several challenges, which are addressed in HADES. We refer readers to [26] for a detailed description of these challenges and the solutions. Note that a logon session is a computing session assigned with an access token representing the authenticated account's security context, and a unique ID (until next reboot) to label system activities run in that session [79,80]. Each logon session contains only a fraction of all system activities collected from a machine. An enterprise-level server can run cumulatively thousands of logon sessions in only one day's operation. This significantly increases the risk of creating an extremely dense provenance graph impossible to investigate for security analysts: Being unaware of logon sessions, a provenance tracing approach would connect all system activities inside a machine, as they all in the end can be traced back to the root process, i.e., `system` (PID=4) on Windows. The more logon sessions a machine hosts, the more obvious logon session-based tracing's performance in reducing provenance graphs' size will be, as it identifies truly involved logon sessions and tracks system activities only inside them.

For the sake of simplicity and readability, we assume there are only two logon sessions on each machine in Fig. 2. Also, note that we differentiate between benign system activities and malicious system activities through different colors for illustration purpose only. The task of accurate tracing is to connect system activities that can be attributed to the same identity, be it the attacker or a genuine user, and separate

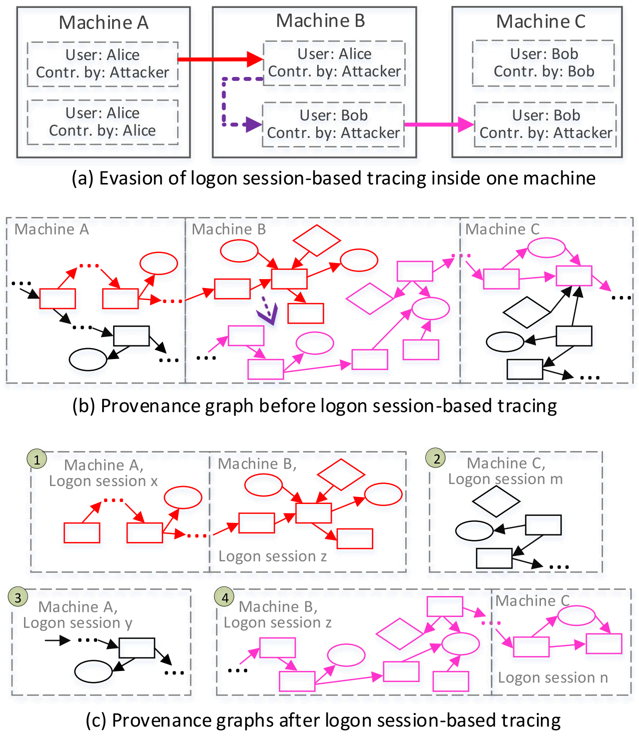

Fig. 4. Evasion of cross-machine tracing via persistence or session hijacking. Whereas black denotes benign system activities, red, violet and pink all denote malicious system activities. The dashed violet edges indicate a persistence or session hijacking attack. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

system activities belonging to different identities. That is, the color coding is transparent to logon session-based tracing.

4.2. *Evading* Hades*'s cross-machine tracing*

Through our study on the MITRE ATT&CK knowledge base [28] and linked APT threat reports, we found three attack tactics/techniques that can be leveraged to evade Hades's cross-machine tracing, i.e., persistence, session hijacking, and port forwarding. These attack techniques are employed frequently by APT attackers, even if their objective is not necessarily to evade Hades's cross-machine tracing. However, leveraging these attack techniques will inevitably misguide and impede cross-machine tracing, if they are not explicitly detected and the tracing process is not properly adjusted.

Evading HADES via Persistence. Cpd [25] is the first work systematically investigating persistence techniques and realizing that an effective persistence is always a two-phase process that includes a persistence setup phase and a persistence execution phase. While the persistence setup phase serves simply as preparation, the persistence execution phase reveals attackers' true intentions. During the persistence setup phase, attackers typically leverage a legitimate system functionality, e.g., Registry Run key [81] or Windows Service [82], to drop some code functioning as a C2 (Command & Control) agent. This C2 agent program is then run automatically during the persistence execution phase. Intuitively, during persistence execution, the C2 program needs to either receive or initiate a remote connection from or to the attacker for further instructions, disclosing the attacker's true objectives.

As the persistence execution phase often starts with a machine reboot or a new user logon, related system activities run under a new logon session. That is, the persistence setup phase and the persistence execution phase often exist on different logon sessions. These logon sessions can belong to the same user or different users. Being unaware of persistence attacks, broken links between involved logon sessions during intra-machine tracing are unavoidable. With logon session-based execution partitioning and tracing, missing cross-session edges in intra-machine tracing would lead to inaccurate and incomplete cross-machine tracing.

An example of how accurate cross-machine tracing can be hindered via persistence is given in Fig. 4. Fig. 4(a) shows a persistence attack inside the Machine B, where the persistence setup is conducted in a logon session under user Alice, and the persistence execution is run in a logon session under user Bob. This is possible when the attacker drops a C2 agent program in the Machine B and links the path to this program with the Registry Run key under user Bob. Therefore every time user Bob logs in, the C2 program is automatically started by the operating system and connects back to the same attacker for further commands. Note that the logon session associated with persistence execution, i.e., the second session in the Machine B, can be under the user Alice as well, if the attacker links the path to the C2 program with the Registry Run key under user Alice, or even under the System user, if, for instance, the attacker hides the C2 program in a Windows service.

The link between persistence setup and persistence execution cannot be established by parsing system logs alone. Only a persistence detection system like Cpd [25] that accurately detects persistence setups and persistence executions of various persistence attack techniques curated by MITRE [28] can discover the broken links and rebuild them into the provenance graphs. As Hades [26] is unaware of evasion attack via persistence, the dashed violet edge indicating a persistence attack in Fig. 4(b) is invisible, leading Hades to attribute the system activities in red and the system activities in pink to two different identities in Fig. 4(c), although they are conducted by the same attacker.

Evading HADES via Session Hijacking. In the present work, as session hijacking, we refer to console/remote desktop session hijacking [30,83], or more precisely logon session hijacking, rather than browser/web session hijacking. Note that console / remote desktop session and logon session are two different concepts, and are at different abstraction levels. A console/remote desktop session contains at least one logon session, and a console/remote desktop session hijacking indicates hijacking of all logon sessions within it. With sufficient privilege on one session, attackers can hijack another existing session. A typical reason for an attacker to conduct session hijacking is that the hijacked session may contain more (credentials-related) information enabling the attacker to further access other devices in the network.

Like persistence attacks, a session hijacking attack is not directly evident in system logs, meaning that the link between a hijacking logon session and a hijacked logon session cannot be created by solely processing system logs. If a session hijacking attack is not detected, the hijacking session would not be linked to the hijacked session during logon session-based tracing, inevitably leading to premature end of whole network tracing. The dashed violet edge in Fig. 4(a) and (b) can also indicate a session hijacking attack, apart from a persistence attack. Unlike in a persistence attack, the second session, i.e., the hijacked session, in the Machine B is typically under a user different from the user in the first session, i.e., the hijacking session. Being oblivious to evasive session hijacking attacks, and hence to the dashed violet edge in Fig. 4(b), Hades will attribute the system activities in red and the system activities in pink to two different identities, as shown in Fig. 4(c), even though they are in fact caused by the same attacker.

Evading HADES via Port Forwarding. Due to network segmentation, attackers frequently rely on port forwarding to reach other machines via an intermediate, already compromised machine, also known as proxy [31]. When port forwarding is employed, the source IP address in a logon event shows the IP address of the proxy machine, instead of the true source machine. However, having the correct source IP address is critical for Hades's cross-machine tracing: During Hades's cross-machine tracing, it takes the source IP address from each logon event, and checks against the authentication events collected from the domain controller and system events on the source host (resolved from the IP address)

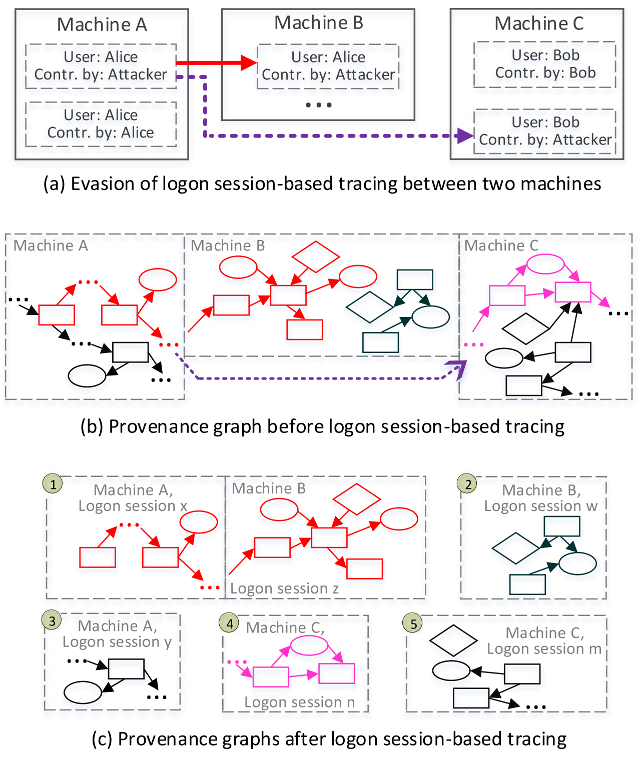
Machine A
User: Alice
Contr. by: Attacker
User: Alice
Contr. by: Alice
Machine B
User: Alice
Contr. by: Attacker
Machine C
User: Bob
Contr. by: Bob
User: Bob
Contr. by: Attacker
(a) Evasion of logon session-based tracing between two machines
Machine A
Machine B
Machine C
(b) Provenance graph before logon session-based tracing
1 Machine A, Logon session x
Machine B
Logon session z
2 Machine B, Logon session w
3 Machine A, Logon session y
4 Machine C,
Logon session n
5 Machine C, Logon session m
(c) Provenance graphs after logon session-based tracing

Fig. 5. Evasion of cross-machine tracing via port forwarding. The dashed violet edges indicate a port forwarding attack step. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to confirm that an authentication request was sent from this host, and a logon was indeed initiated from this host, respectively. If the authentication request and logon initiation are confirmed, Hades will process further events to identify the logon session and user name etc., and continue with the tracing. Otherwise Hades will not proceed the tracing from this logon event.

Apparently, taking the IP address of the proxy machine would lead to premature termination of the tracing, as the logon was not initiated from the proxy machine. Fig. 5(a) shows a port forwarding attack scenario, in which the attacker from the Machine A has logged on the Machine C via port forwarding through the Machine B. Being oblivious to port forwarding, Hades is unable to create the dashed violet edge in Fig. 5 (b), and end up wrongly attributing the system activities in red and in pink to two different identities, as shown in Fig. 5(c). That is, naively relying on the IP address in a logon event could lead to incomplete cross-machine tracing.

The objective of logon session-based execution partitioning and tracing itself is to precisely separate system activities under different identities, rather than classifying system activities as benign or malicious. Yet, the detection accuracy of Hades [26], like other PIDS [12,14], depends on the completeness of returned attack graphs, which in turn rests on the soundness of its tracing method. In the last step of Hades, it ranks attack graphs according to their threat scores, which are determined by indicative attack steps/techniques present in the corresponding graphs. An incomplete attack graph would likely lead to a lower threat score, which in turn could let a true attack graph be ranked lower than false-positive attack graphs.

5. Threat model

Like other PIDS [10–26], our system Commander works under the assumption that the integrity of the underlying OS, firmware and our logging frameworks is guaranteed. Commander focuses on detecting long-running attacks in which threat actors already managed to get an initial access to a domain-joined host via phishing or program exploitation, and move laterally to other machines leveraging stolen domain credentials instead of program exploitation. Attacks confined to a single machine are out of Commander's detection scope.

6. System design

Algorithm 1: Robust Cross-Machine Tracing and Attack Detection

```
Inputs : System audit log events $\mathcal{E}$;
         Authentication & logon events $\mathcal{A}$;
         AD TTP rules $\mathcal{R}_{AD}$;
         ICS TTP rules $\mathcal{R}_{ICS}$
Output: List $L_{<AG,TS>}$ of attack graph and its threat score pairs
1  Function GetAttackGraph($\mathcal{E}$, $\mathcal{A}$)
       /* Get a list of auth. & logon anomalies */
2      $L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>} \leftarrow$ DetectAuthenticationAnomaly($\mathcal{A}$)
3      $L_{<S_{set},S_{exe}>} \leftarrow$ DetectPersistence($\mathcal{E}$)
4      $L_{<S_{hg},S_{hd}>} \leftarrow$ DetectSessionHijacking($\mathcal{E}$, $\mathcal{A}$)
5      $L_{<S_{src},S_{des}>} \leftarrow$ DetectPortForwarding($\mathcal{E}$, $\mathcal{A}$)
6      $L_{<AG,TS>} \leftarrow$ NULL
7      foreach ($\mathcal{A}\gamma$, $\mathcal{L}\gamma$, $\mathcal{T}\gamma$) $\in L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>}$ do
8          $AG \leftarrow$ NULL
9          $HG \leftarrow$ CreateHighLevelGraph($\mathcal{A}\gamma$, $\mathcal{L}\gamma$)
10         $AG \leftarrow AG \cup HG$
11         $AccessType \leftarrow$ CheckRemoteAccessType($\mathcal{A}\gamma$, $\mathcal{E}$)
12         $SessionID \leftarrow$ ReassignSessionID($AccessType$, $\mathcal{A}\gamma$, $\mathcal{E}$)
13         $LinkedSessionID \leftarrow$ LinkSessions($SessionID$, $\mathcal{A}\gamma$, $\mathcal{E}$)
14         ($\mathcal{E}\alpha$, $G\alpha$) $\leftarrow$ TraverseBackward($SessionID$, $LinkedSessionID$, $\mathcal{E}$,
               $L_{<S_{set},S_{exe}>}$, $L_{<S_{hg},S_{hd}>}$, $L_{<S_{src},S_{des}>}$)
15         ($\mathcal{E}\kappa$, $G\kappa$) $\leftarrow$ TraverseForward($SessionID$, $LinkedSessionID$, $\mathcal{E}$,
               $L_{<S_{set},S_{exe}>}$, $L_{<S_{hg},S_{hd}>}$, $L_{<S_{src},S_{des}>}$)
16         $AG \leftarrow AG \cup G\alpha \cup G\kappa$
17         $\mathcal{E}\alpha \leftarrow \mathcal{E}\alpha \cup \mathcal{E}\kappa$
18         $\mathcal{A}\alpha \leftarrow$ CheckAuthenticationLogon($\mathcal{E}\alpha$, $\mathcal{A}$)
19         if $\mathcal{A}\alpha$ is not null then
20             goto 11
21         end
22         $TS \leftarrow$ GetThreatScore($AG$)
23         $L_{<AG,TS>} \leftarrow L_{<AG,TS>} \cup (AG, TS)$
24     end
25     return $L_{<AG,TS>}$

26 Function DetectAuthenticationAnomaly($\mathcal{A}$)
27     $L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>} \leftarrow$ null
28     foreach $ae \in \mathcal{A}$ do
29         $\mathcal{A}_\alpha \leftarrow$ TraversalBackward($ae$)
30         $\mathcal{A}_\kappa \leftarrow$ TraversalForward($ae$)
31         $\mathcal{A}_\alpha \leftarrow \mathcal{A}_\alpha \cup \mathcal{A}_\kappa$
32         ($\mathcal{L}\alpha$, $\mathcal{T}\alpha$) $\leftarrow$ CheckAttackType($\mathcal{A}_\alpha$)
33         if $\mathcal{L}\alpha \in L_{<\mathcal{L}_{attack}>}$ then
34             $L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>} \leftarrow L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>} \cup (\mathcal{A}\alpha, \mathcal{L}\alpha, \mathcal{T}\alpha)$
35         end
36     end
37     return $L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>}$

38 Function GetThreatScore($AG$)
39     $L_{<TS_{sub}>} \leftarrow$ null
40     foreach ($CrossMachineEdge$, $\mathcal{T}\omega$) $\in AG$ do
41         $TS_{sub} \leftarrow$ CalculateScore($AG$, $\mathcal{T}\omega$, $\mathcal{R}_{AD}$, $\mathcal{R}_{ICS}$)
42         $L_{<TS_{sub}>} \leftarrow L_{<TS_{sub}>} \cup TS_{sub}$
43     end
44     $TS \leftarrow$ SumScore($L_{<TS_{sub}>}$)
45     return $TS$
```

Our system Commander follows a three-stage approach for efficient and accurate attack detection. We present Commander's system architecture in Fig. 6, and a formal description of Commander's tracing and detection procedure in Algorithm 1. The first stage of Commander serves as preliminary detection, and consists of four specialized detectors running in parallel, i.e., an authentication anomaly detector, a cyber persistence detector, a session hijacking detector and a port forwarding detector. While the authentication anomaly detector is introduced in Hades [26] and the cyber persistence detector is introduced in Cpd [25], both the session hijacking detector and the port forwarding detector are proposed in our system Commander.

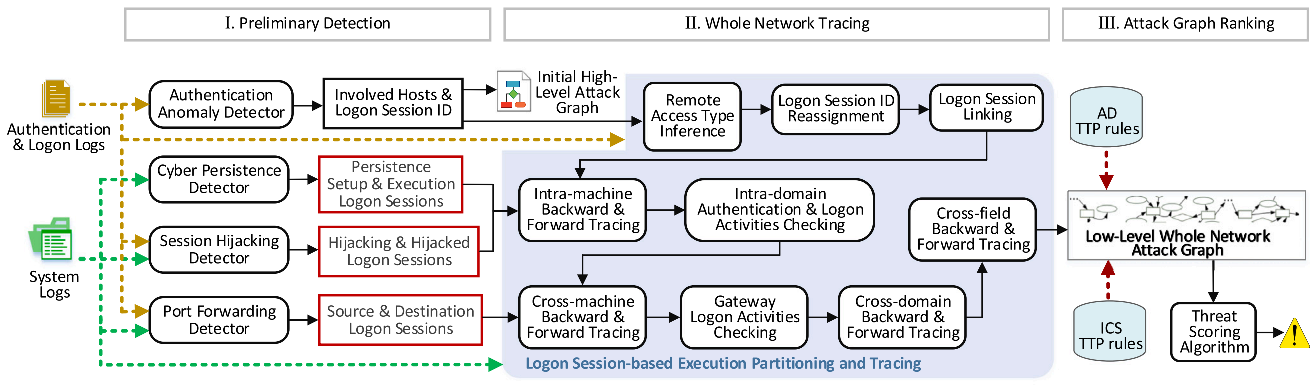
I. Preliminary Detection
II. Whole Network Tracing
III. Attack Graph Ranking
Authentication & Logon Logs
Authentication Anomaly Detector
Involved Hosts & Logon Session ID
Initial High-Level Attack Graph
Remote Access Type Inference
Logon Session ID Reassignment
Logon Session Linking
AD TTP rules
Cyber Persistence Detector
Persistence Setup & Execution Logon Sessions
Intra-machine Backward & Forward Tracing
Intra-domain Authentication & Logon Activities Checking
Cross-field Backward & Forward Tracing
Low-Level Whole Network Attack Graph
System Logs
Session Hijacking Detector
Hijacking & Hijacked Logon Sessions
Port Forwarding Detector
Source & Destination Logon Sessions
Cross-machine Backward & Forward Tracing
Gateway Logon Activities Checking
Cross-domain Backward & Forward Tracing
ICS TTP rules
Threat Scoring Algorithm
Logon Session-based Execution Partitioning and Tracing

Fig. 6. System overview of COMMANDER.

The authentication anomaly detector is integrated into COMMANDER as the first preliminary detector, which takes only authentication & logon logs as input. It is implemented as a dedicated module in COMMANDER. COMMANDER starts with running this module and waiting for its outcome, i.e., a list of authentication & logon anomalies. Each such anomaly includes information of all involved hosts and the corresponding logon sessions; this information is used to initiate the whole network tracing in COMMANDER's second stage.

In the meantime, i.e., before COMMANDER's second stage starts, the cyber persistence detector runs as the second integrated preliminary detector. This detector is also implemented as a dedicated module in COMMANDER. It parses through system logs to identify a list of persistence-related logon session pairs, which is then used in COMMANDER's second stage to defend against persistence-based evasion attacks. That is, during the intra-machine backward & forward tracing in COMMANDER's second stage (Fig. 6), COMMANDER checks against this list to find out whether there is a persistence-related logon session in the traced sessions. By doing so, COMMANDER can pinpoint further logon sessions involved in evasive persistence attacks. Otherwise, the whole network tracing could be interrupted like in Fig. 4, resulting in the entire attack chain being missed.

Unlike the first two above-mentioned preliminary detectors, the session hijacking detector and the port forwarding detector proposed in our system COMMANDER require both system logs and authentication & logon logs as input. These two detectors are implemented in two separate dedicated modules in COMMANDER, and run along with the first two preliminary detectors during COMMANDER's first stage. The objectives of the session hijacking detector and the port forwarding detector are the same; they serve to counter evasion attacks and hence ensure robust and correct tracing in COMMANDER's second stage. The session hijacking detector produces a list of hijacking & hijacked logon session pairs, against which COMMANDER checks during the intra-machine backward & forward tracing in its second stage. Like the cyber persistence detector, this detector prevents the whole network tracing from being interrupted by session hijacking attacks as in Fig. 4.

Further, the port forwarding detector outputs a list of source & destination logon session pairs, against which COMMANDER checks during the cross-machine backward & forward tracing in its second stage. This is to discover any port forwarding-related logon session in the traced sessions, so that COMMANDER can track down further logon sessions involved in evasive port forwarding attacks. Otherwise, the whole network tracing could be interrupted like in Fig. 5.

COMMANDER's second stage starts by taking as input the identified host names, user names and their logon session ID for each anomaly from the authentication anomaly detector. It performs robust logon session-based execution partitioning and tracing to discover all involved logon sessions from each host the same identity has accessed. The whole network tracing ends, if no further logon session and host can be found. It produces a low-level whole network provenance graph for each anomaly spotted in COMMANDER's first stage.

In the final stage of COMMANDER, it matches the provenance graphs resulted from the whole network tracing against open-source TTP (Tactics, Techniques, Procedures) detection rules [78] for Active Directory discovery and credential access, and ICS TTP detection rules we developed for Siemens SIPROTEC and Hitachi Energy RTU500 industrial controllers. Then it ranks these attack graphs according to a threat scoring algorithm.

Our system COMMANDER is unique for several reasons. First, COMMANDER can accurately detect *cross-machine multi-phase* APT attacks, while existing systems cannot. Second, COMMANDER's modular design further enhance the robustness against evasion attacks. Third, COMMANDER is designed as a provenance-based detection system for detecting attacks on industrial-sector organizations consisting of an enterprise network and an industrial control system (ICS). That is, COMMANDER includes parsers we developed for system logs from two popular industrial controllers in energy systems. Further, COMMANDER also incorporates TTP detection rules we developed with a reference to the MITRE ATT&CK Matrix for ICS [34] for these industrial controllers. Most importantly, COMMANDER can accurately attribute the system activities on these industrial controllers to the true identity behind those actions, even when the access originates from the enterprise network.

6.1. Preliminary detection

In the following, we discuss the four specialized detectors in the preliminary detection stage of COMMANDER one by one. As the authentication anomaly detector and the cyber persistence detector are described in HADES [26] and in CPD [25] in detail, respectively, we discuss these two detectors only briefly in the present paper.

6.1.1. Authentication anomaly detection

The authentication anomaly detector rests on a light-weight authentication anomaly detection model introduced in HADES [26]. This model results from an extensive analysis of identity-based attacks in Active Directory. In these attacks, the complex network authentication process in Windows domain is exploited by attackers who succeed in stealing some form of credentials or credentials-related data. The stolen credentials from one machine are then used to authenticate against a domain controller, and hence access other machines.

In order to detect authentication anomalies, this module of COMMANDER inspects each authentication & logon log event, and performs backward tracing on each logon event, and backward & forward tracing on each authentication event (Algorithm 1 Lines 29–30). Note that the authentication process in Active Directory is a multi-step process. The authentication & logon tracing's outcome is matched against the light-weight anomaly detection model to find out whether an authentication anomaly is evident and, if yes, to which attack type it belongs (Algorithm 1 Line 32). For each authentication anomaly, like HADES, COMMANDER produces a high-level attack graph including involved users and machines only (Algorithm 1 Line 9), marking the end of its stage one. Fig. 7 presents an example of such high-level attack graphs at its top right corner. It shows that user Alice from the Exchange Server has used user Bob's password hash to authenticate against the Domain Controller, and then accessed the Data Server, following the Pass-the-Hash behavior.

6.1.2. Cyber persistence detection

Understanding persistence attacks' nature is critical for persistence detection and hence correct provenance tracing. CPD [25] is the first specialized cyber persistence detection system, which recognizes that a successful persistence attack is always a two-part act consisting of the persistence setup phase and the persistence execution phase. While the persistence setup phase serves only as preparation, the persistence execution phase discloses attackers' true motives. As discussed in Section 4.2, the persistence setup phase and the persistence execution phase often exist on different logon sessions, necessitating explicit and correct association of these logon sessions. Otherwise, broken links between involved logon sessions controlled by the same attacker would be inevitable, ultimately resulting in incomplete cross-machine tracing.

We implement CPD as a preliminary detection module for COMMANDER, so that during COMMANDER's whole network tracing in the second stage, persistence related logon sessions can be explicitly associated. Thus, broken links due to persistence attacks can be avoided. The cyber persistence detector first parses all system logs, and matches them against a set of persistence setup detection rules introduced in [25] to create a *persistence setup table*. Next, it spots all processes with remote connections from those system logs, and performs backward tracing on these processes individually to create sub-graphs. Each sub-graph is then checked against a set of persistence execution detection rules, producing a *persistence execution table*.

Subsequently, the entries in the persistence setup table and persistence execution table are aligned with each other based on TTP labels, temporal order, and specific attributes. An alignment indicates a potential persistence attack. These alignments are fed to a false positive reduction algorithm introduced in CPD [25], which integrates key insights from real-world persistence attacks, for enhanced detection accuracy. The final outcome of the cyber persistence detector module in COMMANDER is a list of logon session pairs, each include a persistence setup logon session and a persistence execution logon session (Algorithm 1 Line 3).

6.1.3. Session hijacking detection

COMMANDER's session hijacking detector module leverages two key insights: (1) a successful session hijacking will cause a *logon* event under hijacked user and a *logoff* event under hijacking user almost at the same time,[1] and (2) in order to perform session hijacking, the attacker has to first conduct privilege escalation to receive the highest privilege, i.e. system privilege on Windows.

Algorithm 2 outlines the session hijacking detection procedure in COMMANDER, which starts from processing each logon event and then checking for a possibly related logoff event under a different user (Lines 3–4). After finding a related logoff event, it inspects the terminated logon session due to the logoff, and performs backward tracing from this logon session (Algorithm 2 Lines 6–7). Subsequently, it checks if the tracing result contains a privilege escalation to the highest privilege (Algorithm 2 Line 8). With logon session-based execution partitioning and tracing, detecting privilege escalation is not only most accurate

[1] Note that Windows Security events with event ID equal to 4778 or 4800 are considered as logon or logoff events, respectively. Whereas event ID 4778 reveals that a disconnected session was resumed/logged on, event ID 4800 states that an ongoing session was locked/logged out [84,85]. For instance, these events are always logged by Windows when a RDP session, and hence the associated logon session(s), are hijacked.

Algorithm 2: SESSION HIJACKING DETECTION

```
Inputs : System audit log events $\mathcal{E}$;
         Logon events $\mathcal{O}$;
         Logoff events $\mathcal{F}$
Output: List of hijacking & hijacked logon session pairs $L_{<S_{hg},S_{hd}>}$

1  $L_{<S_{hg},S_{hd}>} \leftarrow$ null
2  foreach $\mathcal{O}_\kappa \in \mathcal{O}$ do
       /* Get the logon time, user and host of current
          logon event                                      */
3      $(\mathcal{T}\kappa, \mathcal{U}\kappa, \mathcal{H}\kappa) \leftarrow$ GetAttribute($\mathcal{O}_\kappa$)
4      $\mathcal{F}_\gamma \leftarrow$ CheckLogOff($\mathcal{F}, \mathcal{T}\kappa, \mathcal{U}\kappa, \mathcal{H}\kappa$)
5      if $\mathcal{F}_\gamma$ is not null then
6          $(SessionID, \mathcal{T}\gamma, \mathcal{U}\gamma) \leftarrow$ GetAttribute($\mathcal{F}_\gamma$)
7          $(\mathcal{E}\gamma, G\gamma) \leftarrow$ TraverseBackward($\mathcal{E}, \mathcal{H}\kappa, SessionID, \mathcal{U}\gamma, \mathcal{T}\gamma$)
8          $peh \leftarrow$ CheckPrivilegeEscalationHst($\mathcal{E}\gamma, G\gamma$)
9          if $peh$ is not null then
10             $L_{<S_{hg},S_{hd}>} \leftarrow L_{<S_{hg},S_{hd}>} \cup (S\gamma, S\kappa)$
11         end
12     end
13 end
14 return $L_{<S_{hg},S_{hd}>}$
```

Algorithm 3: PORT FORWARDING DETECTION

```
Inputs : System audit log events $\mathcal{E}$;
         Authentication & logon events $\mathcal{A}$;
         List of port forwarding-related processes $L_{<\mathcal{P}>}$
Output: List of source & destination logon session pairs $L_{<S_{src},S_{des}>}$

1  $L_{<S_{src},S_{des}>} \leftarrow$ null
2  foreach $\mathcal{E}_\kappa \in \mathcal{E}$ do
       /* Get process name, host name, and timestamp of
          current event                                    */
3      $(\mathcal{P}_\kappa, \mathcal{H}\kappa, \mathcal{T}\kappa) \leftarrow$ GetSubject($\mathcal{E}_\kappa$)
4      if $\mathcal{P}_\kappa \in L_{<\mathcal{P}>}$ then
5          $(IP_{src}, Port_{src}, IP_{det}, Port_{det}) \leftarrow$ CheckCmd($\mathcal{P}_\kappa$)
6          if $Port_{src}$ is not null then
7              $po \leftarrow$ CheckPortOpening($Port_{src}, \mathcal{E}$)
8              $IP_{pro} \leftarrow$ GetIp($\mathcal{H}\kappa$)
9              if $po$ is not null then
10                 $(\mathcal{A}_\gamma) \leftarrow$ CheckLogon($\mathcal{A}, \mathcal{T}\kappa, IP_{pro}, IP_{det}, Port_{det}$)
11                 foreach $ae \in \mathcal{A}_\gamma$ do
12                     $(S\gamma, S\omega) \leftarrow$ TraversalBackward($ae, IP_{src}$)
13                     $L_{<S_{src},S_{des}>} \leftarrow L_{<S_{src},S_{des}>} \cup (S\gamma, S\omega)$
14                 end
15             end
16         end
17     end
18 end
19 return $L_{<S_{src},S_{des}>}$
```

but also very efficient, as an operating system like Windows introduces the separation of privileges with logon sessions. That is, given the OS integrity, COMMANDER's session hijacking detector only needs to check a few attributes, i.e., "Token Elevation Type" and "Integrity Level", in a process generation system event to detect a privilege escalation. Finally, the session hijacking detector outputs a list of logon session pairs consisting of a hijacking logon session and a hijacked logon session.

6.1.4. Port forwarding detection

The port forwarding detector module in COMMANDER identifies port forwarding setup in each machine from system logs, and checks for occurrence of port forwarding-based logon from authentication & logon logs. First, it parses each system log event to get the process name, and checks if it belongs to the list of port forwarding-related processes, e.g., `plink`, `ssh`, `netsh` (Algorithm 3 Lines 3–4). Next, it parses the command line arguments of that process to determine the source port & IP and destination port & IP for the port forwarding (Algorithm 3 Line 5). If a source port can be identified, it checks in system logs if this port was opened through the host-based firewall, and the IP address of this proxy machine (Algorithm 3 Lines 7–8). Subsequently, it parses all logon events on the destination host that have happened after the timestamp of the port forwarding setup event, and have the IP address of the proxy machine as source IP (Algorithm 3 Line 10). For each found logon event, it performs backward tracing with the corrected source IP address to find the source logon session, and then appends the source & destination logon session pair to a list.

6.2. Whole network tracing

In its stage two, Commander performs whole network tracing, which improves Hades's logon session-based tracing with enhanced robustness against evasion attacks, and extends it to trace also across domains inside an industrial-sector organization. This stage starts with processing the data delivered from the authentication anomaly detector, i.e., identified host names, user names and logon session ID. First, it infers the remote access type, in which it pulls both related authentication & logon events and system events. Then it extracts a list of attributes from the events to analyze the context, and classify the remote access type (Algorithm 1 Line 11).

Next, it performs logon session ID reassignment and logon session linking according to the remote access type (Algorithm 1 Lines 12–13). Logon session ID reassignment is necessary for several remote access types, e.g., RDP (Remote Desktop Protocol) and Powershell-Remoting, as the system activities performed on a new logon session are recorded under an existing logon session ID. With logon session linking, a user logon session is coupled with the system logon session directly responsible for the creation of that user logon session. Note that a system logon session is normally a long-running session accommodating numerous system activities. The logon session linking module selects only a fraction of these system activities, which are related to the user session creation, to parse into the provenance graph.

Subsequently, Commander proceeds with intra-machine backward & forward tracing to find all involved logon sessions and system activities inside them (Algorithm 1 Lines 14–15). This step automatically spots privilege escalation, as system activities run with different privileges are contained in different logon sessions and labeled with different session ID. A process's privilege is tied to the security context of the logon session in which the process is running. During normal process creation, both child process and parent process run in the same logon session. But, during a privilege escalation, when a process is spawned from another process with lower privilege, the spawned process runs in a new logon session. Unlike Hades, Commander checks in this step also if current logon sessions are involved in a persistence attack or a session hijacking attack, by inspecting the list of persistence setup & execution logon session pairs, and the list of hijacking & hijacked logon session pairs. These lists are passed from the cyber persistence detector and the session hijacking detector, respectively. If it finds a match, it connects the related logon sessions with a corresponding edge, and continues the intra-machine backward or forward tracing on the newly joined logon session.

Afterwards, Commander examines the authentication & logon logs from the same domain. This is, on the one hand, to determine the source machine & logon session of current logon sessions, i.e., cross-machine backward tracing, and, on the other hand, to identify logon sessions inside any domain-joined machine initiated from the current machine, i.e., cross-machine forward tracing. During this step, Commander also checks if the traced logon sessions are involved in a port forwarding attack, by inspecting the list of source & destination logon session pairs output from the port forwarding detector. It attaches any newly found logon sessions to existing ones. At the end of intra-domain tracing, Commander checks logon activities on the gateway server. If any logon initiated from a machine in the IT domain is found, and that machine is involved in the current tracing process, Commander begins the cross-domain tracing. Commander's performance to trace across domains inside an organization benefits from the fact that an industrial-sector organization's network is generally segmented into an IT domain network and an OT domain network, and the connection of these two domains is done via a gateway/jump server, as mentioned in Section 2.

Following the cross-domain tracing, Commander checks if the Engineering Workstations are involved in the tracing process. If yes, it then examines whether there are logon activities on the field controllers coming from a logon session in a Engineering Workstation. For each logon event, it parses all log events between that logon and the corresponding logoff on the field controller. Field controllers typically have a single-user single-task embedded OS. Unlike general-purpose OS, embedded OS do not implement the concept of logon session, meaning that system activities cannot be sorted by logon session ID. However, thanks to its single-user restriction, it is straightforward to correctly link a specific logon session on the Engineering Workstation with only system activities on a field controller that are indeed caused by commands from that session.

Fig. 7 renders the result of whole network tracing for the extended Oilrig attack scenario in the AVIATOR dataset, unraveling the attacker's traversal inside the target organization. Commander first creates an initial high-level attack graph including only high-level entities, i.e., users and hosts, after the authentication anomaly detector identifies a potential Pass-the-Hash attack. In Pass-the-Hash attacks [86], attackers leverage stolen password hashes and choose NTLM authentication process over the default Active Directory authentication process Kerberos. Then it performs forward tracing inside the specific logon session under user Bob in the accessed host Data Server, and forward & backward tracing inside the logon session of Alice in the accessing host Exchange Server. The remote access type on the Data Server is identified as PsExec [87], for which session ID reassignment is not required. User Bob's logon session (`0x4aa81e86`) is linked to the System logon session ($0 \times 3e7$), as the server process `psexesvc` runs in the System logon session. `psexesvc` receives remote commands from the Exchange Server, and the remote commands run in Bob's logon session. The whole network forward tracing from the Data Server leads to no further logon session on another machine, as the attacker found the targeted data and did not move forward from the Data Server.

In contrast, the whole network backward tracing from the Exchange Server discloses that the current logon sessions (`0xb4874`, `0xb4829`, `0xb37f41d122`) on it belong to a RDP session that was initiated from another logon session of user Alice inside another machine Workstation_1. Besides, it spots a privilege escalation from unprivileged user to Administrator, i.e., from `0xb4874` to `0xb4829`, and a further privilege escalation from Administrator to System, i.e., from `0xb4829` to `0xb37f41d122`. Subsequently, it proceeds with the backward & forward tracing from the logon session of Alice (`0xc8cf0a`) in the Workstation_1. While backward tracing from the Workstation_1 leads to no further domain-joined machine, but rather an external IP address that belongs to the C2 server run by the attacker, forward tracing from the Workstation_1 unveils multiple logon sessions on several machines.[2]

Among the newly found logon sessions, two sessions are on the Exchange Server. These two sessions predate other sessions on the Exchange Server. One of these two sessions contains system activities related to credential access, contributing to the lateral movement to the Data Server. Unlike Hades, Commander performs also cross-domain tracing inside the same organization, resulting in identifying a logon session in the Gateway Server located in the OT domain. Besides, each time Commander discovers a new logon session during tracing, it checks against the logon session pairs identified by its cyber persistence detector, session hijacking detector and port forwarding detector, individually. This is to find out whether this logon session is involved

[2] Note that backward tracing can lead to no more than one logon session, whereas forwarding tracing may lead to multiple logon sessions, as one may remotely access several other machines from one machine.

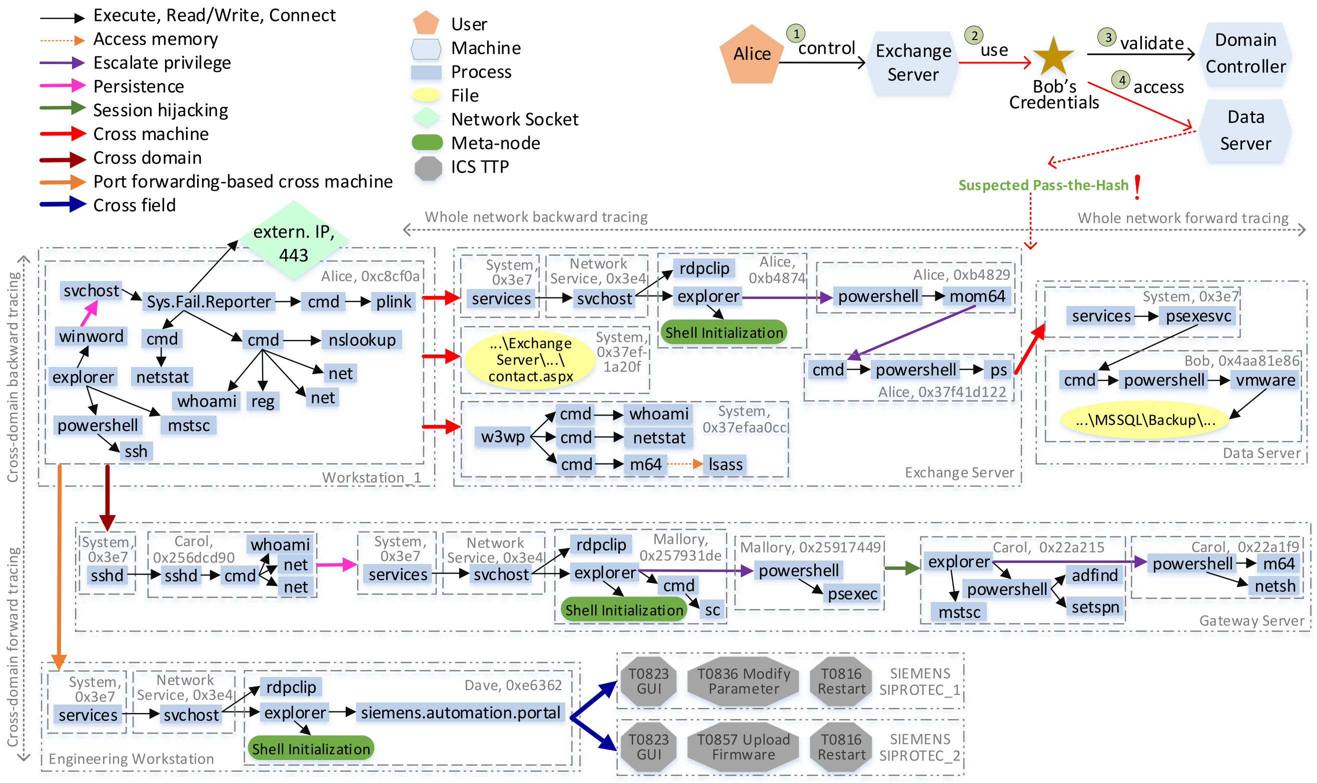

Fig. 7. An attack graph created by Commander on the AVIATOR dataset for the extended Oilrig attack scenario. Note that the bounding boxes with a session ID label or host name are manually added for better readability, this information is encoded in nodes in the original attack graph. Besides, we replaced the user names in the original emulation plan with shorter names. Some processes, e.g., ps and vmware, in the attack graph are malicious processes disguised under a false name. Further, meta-nodes are introduced as an optimization technique in Hades [26] to conclude system activities of standard known benign routines.

in stealthy attacks that could impede accurate tracing. By doing so, Commander discovers a few more logon sessions in the Gateway Server and in the Engineering Workstation. The logon session in the Engineering Workstation under user Dave finally leads to logon sessions on two Siemens field controllers.

Because the first logon session of user Carol (0 × 256dcd90) on the Gateway Server is identified as a persistence setup logon session by the cyber persistence detector, a persistence-based cross-session edge, named as pseudo-edge in Cpd [25], is created to connect it to the persistence execution logon session (0 × 3e7). Without the cyber persistence detector, forward tracing from Carol's first logon session would end up finding no further logon session and stopping the tracing process. During the persistence attack step, the attacker has created a new local user Mallory on the Gateway Server. This persistence technique [88] provides the attacker the ability to access the desktop environment on the Gateway Server. Note that the first access to the Gateway Server was done via SSH and through public key authentication. That is, without knowing user Carol's password, the attacker was able to log into the Gateway Sever as Carol using a private key stored in the Workstation_1.

Following the persistence execution, the attacker hijacked user Carol's RDP session and hence the logon sessions (0 × 22a215, 0 × 22a1f9) inside it. From these logon sessions, the attacker was able to conduct Active Directory discovery to find more machines in the OT domain, as the user Carol is a domain user and the attacker-created user Mallory is not.[3] Besides, the attacker has set up a port forwarding in the session with higher privilege (0 × 22a1f9). Enabled by the port forwarding, the attacker was able to access the Engineering Workstation in the OT domain directly from the Workstation_1 in the IT domain. Thanks to the port forwarding detector, the link to the Engineering Workstation can be discovered during the tracing. Further, cross-field tracing is initiated from the Engineering Workstation. Due to the coarse granularity of system logs on field devices, it is not feasible to create links between the system activities inside a device. Hence, in the attack graph, we use more abstract entities, i.e., matched ICS TTP, and cluster and link them to the logon session on the Engineering Workstation from which the corresponding commands originate.

6.3. Attack graph ranking

In an operational environment, as a detection system cannot guarantee the perfect detection performance, i.e., no false positives and no false negatives at the same time, each detection result needs to be further investigated by a security analyst. The reality is however that most detection results/alerts cannot be investigated by security analysts due to limited time [9,89]. Our system Commander can not only produce attack graphs explaining attack history to accelerate further investigation by security analysts, but also rank the attack graphs with threat score. Ranked attack graphs/alerts are a very practical way to handle the trade-offs between false positives and false negatives in

[3] Note that only domain administrators and specifically delegated users can create new domain accounts. Local users are restricted from accessing domain information. Depending on the remote access type, a domain user's logon session's security context may not be sufficient for accessing domain information, a restriction implemented on Windows. That is, from the SSH-enabled logon session (0 × 256dcd90) of the domain user Carol, the attacker was unable to conduct Active Directory discovery.

reality. This is because that, in order to avoid false negatives, it still allows a great number of false positives, as a trade-off. However, with threat score ranking, those false positive alerts are more likely ranked lower than true positive alerts, and get investigated later, if a security analyst still has time for them. That is, security analysts can *prioritize* the highest ranked alerts for investigation due to limited time.

Commander's threat scoring scheme in its final stage builds on Hades's, resting on two key insights introduced in Hades [26]: (1) long-running attacks against a large organization follow a rigid pattern after the initial access: Active Directory discovery, credential access, lateral movement and privilege escalation, (2) a successful credential access is essential for the attacker to proceed with the attack, and the most observable result of an ongoing attack is lateral movement. For each whole network attack graph output from Commander's stage two, Commander calculates a threat score. By ranking attack graphs via threat score, Commander presents security analysts with the most likely true attack graphs first for further investigation.

Commander extends Hades's threat scoring scheme by also considering the occurrence of (1) persistence, session hijacking and port forwarding attacks, (2) cross-domain activities, and (3) ICS TTP alerts in an attack graph for score calculation. That is, for each potential lateral movement, i.e., a cross-machine edge in an attack graph, it checks the variety and frequency of credential access techniques and Active Directory discovery techniques used in the source host. It also checks the presence of persistence and session hijacking attacks in both source and destination hosts, and whether this is a port forwarding-enabled or a cross-domain lateral movement. Further, it checks whether the destination host is a field controller on which ICS TTP alerts are present. Each occurrence of these indicators increases the threat score.

More formally, the threat score is determined by the following two equations, of which Eq. (1) calculates the threat score for each cross-machine edge, and Eq. (2) accumulates the threat scores passed from Eq. (1), and yields the final score.

$$TS_E = \sum_{i=1}^{n} (Freq(teq_i) \times Var(tac_i))^{w_i} \tag{1}$$

where n denotes the number of tactics involved, e.g., credential access or persistence. $Freq(teq_i)$ denotes the accumulated execution frequency of techniques in a given tactic, and $Var(tac_i)$ denotes the number of techniques being applied for the given tactic. w_i is a weighting factor for each tactic, with credential access having the highest weight due to its importance as per the second key insight.

$$TS_G = \sum_{i=1}^{n} (TS_i \times (DA + 1) \times Criticality) \tag{2}$$

where n denotes the number of cross-machine edges found in a given attack graph, and TS_i denotes the threat score assigned for each cross-machine edge from Eq. (1). DA indicates whether credentials of domain administrators are involved in the cross-machine activities, and has the value 0 or 1. Different values of $Criticality$ are given to different attack types, in which the Golden-Ticket attack type receives the highest value, as it implies a compromise of an entire domain. Besides, it scales up $Criticality$ accordingly, if a cross-machine edge (1) is port forwarding enabled, or (2) connects two machines from two different domains, i.e., cross domain, or (3) is tied to a field controller on which ICS TTP alerts are present, i.e., cross field.

7. Implementation

We implemented a prototype of Commander in Python, and deployed it on a 64-bit Ubuntu 22.04 OS with 256 GB of RAM and a 32-core processor. This machine also hosts Elasticsearch [90], to which Commander interfaces via EQL [91], a query language designed for security purposes. Like Hades, Commander operates as an on-demand tracing system for enhanced efficiency. That is, it starts the whole network tracing, only after an authentication anomaly is identified by the authentication anomaly detector in the preliminary detection stage. During the tracing, it searches for relevant events to process via Elasticsearch, which provides scalable and near real-time search for log data investigation. Moreover, we use Python NetworkX [92] for creating provenance graphs on demand, and PyVis [93] for graph visualization.

Whereas authentication logs for each domain are collected from the corresponding domain controller, logon logs are collected from each accessed domain-joined machine. System logs are collected from all machines. On Windows, we collect Windows Security [80] logs and Sysmon [94] logs. Sysmon generates and records a process GUID (Globally Unique Identifier) for each process, reducing false dependencies during post-processing. However, Sysmon lacks logging for file/Registry *read* operations.[4] Hence, we rely on Windows Security [80] logs for these system activities, as well as authentication and logon activities. On Linux we collect Auditd [95] logs. These logs are all industry-standard logs, boosting Commander's applicability in industry. To ensure log integrity, recent versions of Sysmon start as protected processes, which prevent tampering or disabling by attackers. Windows introduced *Protected Event Logging* [96] to secure logs from unauthorized access. On Linux, making Auditd configuration immutable [97] safeguards against rule changes and rule disabling.

8. Evaluation

In this section, we evaluate Commander's performance on the AVIATOR [27] dataset. In particular, we compare Commander's detection accuracy with prior detection systems, showcase Commander's response time, and analyze attack graphs produced by Commander. Note that a direct, empirical comparison with prior PIDS is difficult given the absence of source code and differences in evaluation datasets. A comparison between one of Commander's preliminary detectors Cpd and state-of-the-art PIDS KAIROS [23] and FLASH [24] is given in [25], in which it shows that prior PIDS fall short of detecting persistence, due to failure to recognize that persistence attack is always a two-part act. That is, prior PIDS would not be able to detect multi-phase APT attacks. Further, as mentioned in Section 3.1, prior PIDS are, *by construction*, limited to intra-machine tracing, and unable to causally associate an attacker's activities across machines in an organization's network(s), as a result of an obliviousness to the network context.

8.1. Datasets

Attack scenarios in most public datasets taken for evaluation of APT detection and investigation systems often lack not only a persistence attack step, but also a lateral movement attack step, even though these steps are omnipresent in real-world APT attacks [27]. For instance, the DARPA E3 dataset [69], which is one the most used datasets for PIDS evaluation, contains neither a persistence attack step, nor a lateral movement trace. Although the DARPA E5 dataset [71] includes two Windows-based persistence attack instances, attacks in this dataset are confined in individual machines. The more recent DARPA OpTC dataset [98] is the only public dataset emulating APT attacks in a realistic enterprise environment with domain-joined machines. However, this dataset contains only system logs on domain-joined machines, but no logs from the domain controller. Authentication logs, which are recorded only on domain controllers, are critical for not only three out of four Commander's preliminary detectors, but also the whole network tracing. Further, logs in the DARPA OpTC dataset lack a logon session ID, making it even more infeasible to evaluate Commander on this dataset.

[4] Sysmon only records file/Registry modification operations, e.g., creation, deletion, value change, name change. Note that the "Event ID 9: RawAccessRead" will not record *most* file/Registry read operations, as it only records read operations from the drive using the \\.\denotation [94].

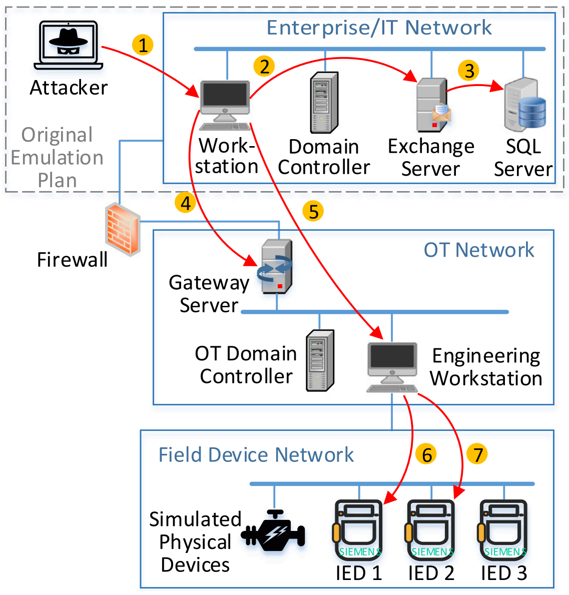

Fig. 8. Extended Oilrig emulation plan [27]. Edges in red denote initial access or lateral movement conducted by the attacker. These edges are numbered in chronological order.

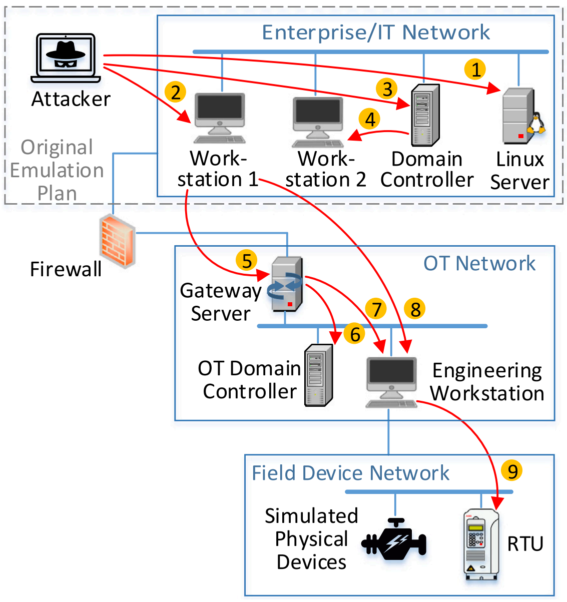

Fig. 9. Extended Sandworm emulation plan [27].

In contrast, at least two persistence attack techniques are included in each of MITRE's eleven full emulation plans [35], whereas Active Directory-based lateral movement is present in nine out of eleven emulation plans. These emulation plans are based on the observation of past real-world APT attacks, and target enterprise networks with more sophisticated, cross-machine attacks than most public datasets, offering greater authenticity. AVIATOR is a dataset derived from MITRE emulation plans. Apart from the original MITRE emulation plans, it also includes data from emulating two extended MITRE emulation plans. These extended MITRE emulation plans are based on real-world attacks on both enterprise and ICS networks conducted by notorious APT groups Sandworm [36] and Oilrig [37]. Whereas Sandworm is responsible for the attacks against Ukrainian electrical companies in 2015, 2016 and 2022 [99–102], Oilrig has targeted energy and chemical sectors in Middle Eastern in 2017 [103–105].

The extended emulation plans are demonstrated in Fig. 8 and Fig. 9, in which the original emulation plans are shown in a dashed bounding box. As the extended emulation plans are the only ones including attack steps on ICS, we evaluate Commander only on the corresponding subsets of the AVIATOR dataset. Table 2 gives an overview of the two subsets, which are also referred to as AVIATOR-Oilrig dataset and AVIATOR-Sandworm dataset, respectively, for brevity.

It is worth noting that the possible errors of a dataset presented in [106], i.e., incorrect labeling logic and wrongly generated features, do not apply to the AVIATOR dataset. Like the DARPA E3 [69], the DARPA E5 [71] and the DARPA OpTC [98] datasets, the AVIATOR dataset is not a dataset specifically created for training and evaluating machine learning approaches. That is, unlike the examined network log-based datasets in [106], DARPA E3, DARPA E5, DARPA OpTC and AVIATOR are host log-based datasets, in which no fine-grained automatic labeling was performed for each system activity. Only coarse-granular labeling for each attack trace was performed, which is manual, simple, and easily verifiable. Along with the raw data, source code for environment setups, configuration files, attack scripts etc. is also provided as a documentation in the dataset's repository [107]. Besides, no feature generation was performed for APT datasets like DARPA E3, DARPA E5, DARPA OpTC and AVIATOR. That is, these datasets do not contain any wrong synthetic feature.

8.2. Commander *vs. Prior detection systems*

To compare Commander with prior detection systems, including open-source detection rules and a commercial Active Directory attack detector, we first extracted all detection rules for Active Directory-based attacks from the most well-known rule repositories, i.e., Elastic [108], Sigma [78], Google Chronicle [109], and installed a commercial Active Directory attack detector. Both the EQL queries converted from the detection rules and the commercial attack detector are evaluated alongside Commander on the datasets. The commercial attack detector is referred to as Cad in the present paper, since we did not get the permission to reveal the name of the security vendor. This vendor is considered as a significant security solution provider in the security market overviews [110,111]. We compare the detection results w.r.t. the number of false positives and false negatives on each dataset in Table 3.

Table 3 shows that open-source detection rules generated more false positives than a full-fledged attack detector like Commander and Cad, and missed some true attacks. More specifically, Elastic produced more false positives than Commander's preliminary detection stage on both datasets, while missing the true attack in the Sandworm dataset. Although Chronicle has a lower false positive rate on both datasets, it did not detect any true attack. In contrast, Sigma identified the true attacks in both datasets, however at the expense of producing much more false positives. Open-source detection rules' overly high false positive rate is well demonstrated and discussed in [25,26].

Contrary to open-source detection rules, a dedicated attack detector like Commander and Cad associates events related to each other, and analyzes the context, moving beyond inspecting individual log events independently. This context-awareness can significantly lower the false positive rate, as demonstrated by the comparison between the detection results of Commander's preliminary detection stage and final detection stage, i.e., after the whole network tracing. Note that most attack steps involved in stealthy APT attacks are based on misusing legitimate system infrastructures and services, which are used intensively by genuine users as well. The prevalence of "indicative" system activities in benign operations helps attackers easily blend each of their attack steps into the noise caused by genuine users. However, with precise tracing, the system activities of individual attack steps can be linked into a provenance graph, as they are causally related.

Table 2
Overview of the evaluation datasets.

Dataset	Target host number	Ground truth attack	Target host OS	Data size	Event number
AVIATOR-Oilrig	10	Pass-the-Hash	Windows Embedded OS	566 GB	360 M
AVIATOR-Sandworm	8	Golden-Ticket	Windows Linux Embedded OS	610 GB	479 M

Note that the AVIATOR-Oilrig dataset and AVIATOR-Sandworm dataset are the subsets of the AVIATOR dataset including attack steps on industrial control systems.

Table 3
Comparison of Commander and prior detection systems.

Dataset	COMMANDER				Elastic		Chronicle		Sigma		CAD	
	Preliminary		Final									
	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN
AVIATOR-Oilrig	105	0	0	0	208	0	50	1	415	0	0	1
AVIATOR-Sandworm	59	0	0	0	196	1	31	1	722	0	0	0

Note that each dataset has only one true positive (TP=1). FP: false positives. FN: false negatives.

Comparing to a provenance graph built from a genuine user's all system activities, a provenance graph containing the whole of an attacker's system activities typically includes more "indicative" activities. It also outlines a logical order, e.g., lateral movement after credential access, underpinning threat scoring schemes in PIDS like [12–14,25,26] and Commander. Often the longer the attack chain is, the more obvious the attack pattern is, and thus the more likely the attack provenance graph is ranked higher than benign ones. In comparison to Hades, which has assigned the third-highest score to the true attacks in two datasets, Commander has produced no false positive by ranking the true attack in each dataset the highest. This is due to the longer attack chain in the extended emulation plans and Commander's robust whole network tracing ability.

Commander's final detection results in Table 3 are based on the true attack threat score threshold in Fig. 10, which shows the cumulative distribution function of threat scores for benign and attack provenance graphs. Like other provenance-based detection systems, e.g., HOLMES [12] and RapSheet [14], the objective of our system Commander is not to remove all false alarms, but to prioritize investigation of attack graphs based on their threat score. The value of the threat score threshold can be configured depending on the human resources of a security operation center has in practice. Fig. 10 shows that the true attack in each dataset has the highest threat score. That is, when the threshold is set with the value of the highest threat score, Commander has a false positive rate of 0 and a true positive rate of 1 in both datasets. However, if the threshold is set, for instance, 50% lower than the highest threat score, Commander still has a true positive rate of 1 in both datasets, but a false positive rate of 1.9% in the Oilrig dataset and a false positive rate of 3.4% in the Sandworm dataset. That is, the threat score assigned by Commander to the true attack is significantly higher than those assigned to most benign graphs in each dataset.

Further, the detection result comparison between Cad and Commander's final stage in Table 3 shows that Commander outperforms Cad by accurately detecting the true attacks in both datasets, thanks to Commander's more advanced tracing module than Cad's. By investigating attack graphs output from Cad, e.g., Fig. 13, we find that these are more abstract graphs, whose graph granularity is only comparable to the initial high-level attack graphs produced at the end of Commander's preliminary detection stage. Consulting on the documentation, which was provided by the vendor and includes a guideline for Cad's logging configuration and detection logic for various kinds of attacks, reveals that Cad does not collect system log events, but rather a subset of authentication & logon event types. This leads to limited visibility into intra-machine system activities, which we deem indispensable for accurate tracing and attack detection. Our engagement with the vendor over weeks confirmed that Cad's detection on Pass-the-Hash attacks has known issues, contributing to Cad's false negative on the Oilrig dataset.

Table 4
Summarized comparison of detection systems.

System	Precision	Recall	F1 score
COMMANDER	1.0	1.0	1.0
Elastic	0.005	0.667	0.01
Chronicle	0.024	0.5	0.046
Sigma	0.002	1.0	0.004
CAD	1.0	0.667	0.8

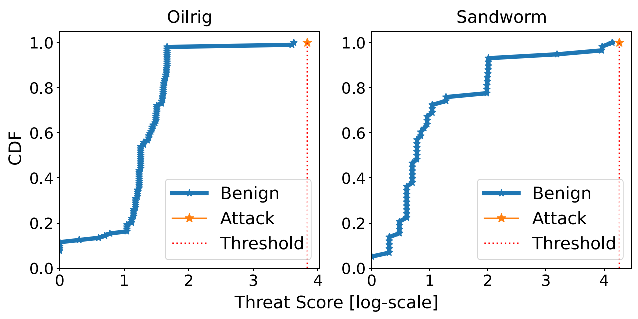

Fig. 10. CDF of threat score for false and true alerts.

We further compare the detection results w.r.t. precision, recall and F1 score in Table 4, in which the detection results on the Oilrig dataset and on the Sandworm dataset are merged for brevity. Table 4 shows that our system Commander significantly outperforms existing systems. Note that the superiority of Commander over existing systems is proved not only by Table 3 and Table 4, but also by the succinct and complete attack graphs it produces, e.g., Fig. 7 and Fig. 12. Commander's ability to produce concise and informative attack graphs explaining attack history of cross-machine multi-phase APT attacks is invaluable for security analysts in practice. Other systems lack this functionality. We compare attack graphs produced by Commander and Cad further in Section 8.4.

8.3. Response time

Fig. 11 shows the cumulative distribution function of Commander's response time on each dataset. Commander's response time is measured per attack graph, i.e., the time it takes to output a single attack graph with calculated threat score. Intuitively, Commander has longer response time than its four integrated detectors, as only after the preliminary detection stage has finished, i.e., all four integrated detectors have finished, Commander's next two stages start, i.e., whole network tracing and threat score assignment. For each attack graph, the time spent on Commander's stage two & three, vary between only 7 s and 32 s on the

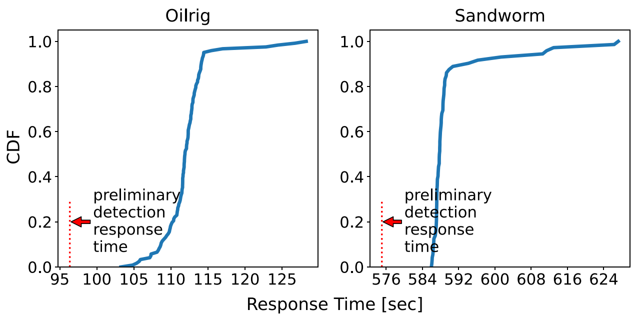

Fig. 11. CDF of response time of Commander.

Oilrig dataset, and between 11 s and 52 s on the Sandworm dataset, indicating how much longer Commander's response time is than its integrated detectors. For more than 90% of attack graphs, Commander's stage two & three takes less than 18 s in total per graph on the Oilrig dataset, and less than 20 s in total per graph on the Sandworm dataset.

However, the absolute response time to output an attack graph ranges from 103 s to 128 s on the Oilrig dataset, and from 586 s to 627 s on the Sandworm dataset. This is due to a bottleneck resulted from Commander's system design, i.e., Commander's stage two only starts after all four preliminary detectors in its stage one have finished. The preliminary detection response time (marked with a dotted line pointed by a red arrow in Fig. 11) takes the largest of response times of the four preliminary detectors running in parallel. On both datasets, the cyber persistence detector has the longest response time among the four detectors, reaching 96 s on the Oilrig dataset and 575 s on the Sandworm dataset. The response time of the cyber persistence detector, as a preliminary detector, is a value accumulated from times for processing all suspected persistence instances identified on each host. That is, this detector's longer response time on the Sandworm dataset results from the fact that it has picked out noticeably more suspected persistence instances to process on this dataset, including false-positive ones.

8.4. Reconstructed attack graphs

Whereas the reconstructed attack graph for the attack in the Oilrig dataset (Fig. 7) is already interpreted in Section 6.2, we discuss in this section the true positive attack graph from the Sandworm dataset in Fig. 12. The tracing on this attack starts from an authentication anomaly mimicking the Golden-Ticket behavior in the OT domain. In Golden-Ticket attacks [112], attackers use stolen credentials of the `krbtgt` account from a domain controller for authentication, and can impersonate any domain account, essentially having the ability to move laterally to any domain-joined machine.

After identifying the Gateway Server as the source host, from which the attacker has used the stolen credentials, and the Engineering Workstation as the destination host to which the attacker has moved, Commander performs forward tracing from the involved logon session under user Dave (0 × `2b8d50717`) on the Engineering Workstation, and forward & backward tracing from the involved System logon session (0 × `3e7`) on the Gateway Server. The forward tracing from the session 0 × `2b8d50717` leads to no further machine or logon session, while the forward tracing from the session (0 × `3e7`) on the Gateway Server exposes a user logon session (0 × `2b65c94e`) belonging to the domain administrator on the OT domain controller.

As Commander also checks the list of persistence setup & execution logon session pairs during the tracing, and correctly identified the session (0 × `3e7`) on the Gateway Server as a persistence execution logon session, the backward tracing from this session results in a logon session under user Carol (0 × `3c128d94`) on the Gateway Server, i.e., the corresponding persistence setup logon session. The link between these two sessions can only be found by a persistence detector. The cyber persistence detector in Commander's stage one correctly identifies this as a WMI persistence instance [113]. Otherwise the backward tracing would lead to nowhere, but a overly premature termination of the whole network tracing and largely incomplete attack graph, like the one Hades would output in this case.

Further, another logon session of user Carol (0 × `3c1ac442`) is spotted during the forward tracing from the newly found session (0 × `3c128d94`), as the session (0 × `3c1ac442`) is a persistence execution logon session of another persistence instance Registry run keys [81]. That is, the attacker has executed two persistence techniques on the same logon session 0 × `3c128d94`, so that every time the machine restarts, the attacker gets a call back from a System logon session, or every time the user Carol logs in, the attacker receives a call back from Carol's logon session like 0 × `3c1ac442`. On the System logon session (0 × `3e7`), the attacker conducted credential access without performing privilege escalation first, and port forwarding & port opening through host-based firewall, since this logon session is always assigned with the highest privilege. On the user logon session 0 × `3c1ac442`, the attacker executed several Active Directory discovery techniques with the programs `net`, `setspn` and `oradad`. Note that chronologically speaking, instead of in the tracing order, this happened before the lateral movement to the OT Domain Controller and the Engineering Workstation.

Next, backward tracing from the session 0 × `3c128d94` reveals a logon session of user Alice (0 × `4500223c`) on the Workstation_1 from the IT domain. Further backward tracing from Alice's session 0 × `4500223c` discloses that this session is a persistence execution logon session, and hence is linked to the corresponding persistence setup logon session, i.e., 0 × `3e7` on the Workstation_1. Forward tracing from the session 0 × `4500223c` leads to another user logon session of Dave (`0xe6362`) on the Engineering Workstation in the OT domain, thanks to the list of source & destination logon session pairs output from the port forwarding detector from Commander's stage one. Without this list, the tracing would end prematurely at this step. Finally, forward tracing from Dave's logon session `0xe6362` results in a session on the field controller Hitachi Energy RTU500 and several activities in it that can be mapped to ICS TTP alerts, e.g., modify parameters, further increasing the threat score of this attack graph.

In comparison, attack graphs produced by our compared system Cad are incomplete and much less informative, slowing down attack investigation by security analysts. For the attack scenario in the AVIATOR-Oilrig dataset, Cad did not output an attack graph, as it has missed the attack. Fig. 13 shows the attack graph produced by Cad for the attack scenario in the AVIATOR-Sandworm dataset. Comparing to the attack graph produced by our system Commander in Fig. 12, Cad's attack graph is much less valuable for security analysts in practice. First, it does not present the complete list of machines and users that are involved in the attack. Second, due to Cad's limited tracing ability, its attack graph reveals no malicious system activities conducted by the attacker. That is, when presented with such an attack graph, security analysts need to perform laborious, error-prone manual analysis to assess the attacker's traversal inside the network for attack remediation and recovery.

9. Limitations & discussion

Although Commander has successfully detected the true attacks in both datasets, the reconstructed attack graph from the Sandworm dataset is not entirely complete. *In practice, a PIDS's ability to reconstruct concise and complete attack graphs is invaluable for security analysts, as detection systems cannot guarantee perfect accuracy.* Further investigation by security analysts is vitally important for further removing false positives and locating true attacks. Concise and complete attack graphs automatically generated from PIDS can substantially speed up the investigation process, and therefore contribute to swift attack recovery & remediation.

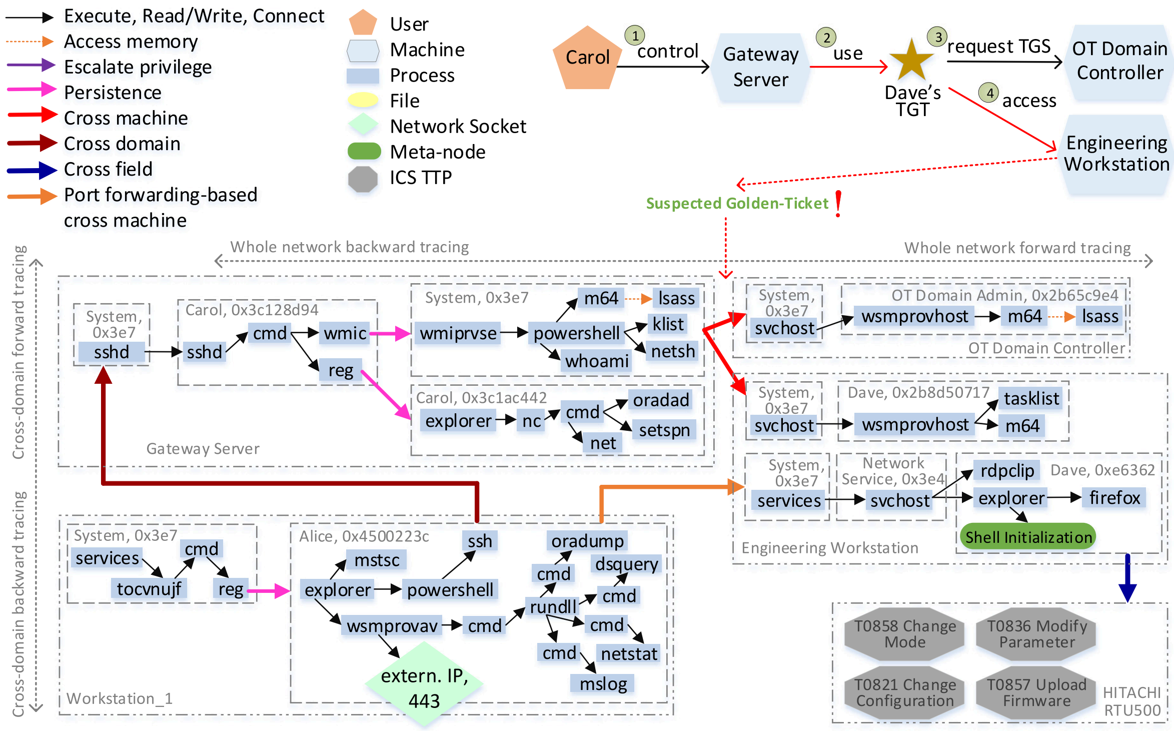

Fig. 12. An attack graph created by Commander on the AVIATOR dataset for the extended Sandworm attack scenario.

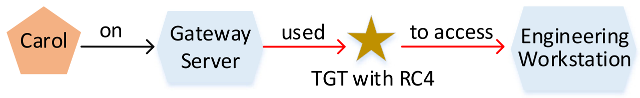

Fig. 13. An attack graph created by Cad on the AVIATOR dataset for the extended Sandworm attack scenario.

During its tracing on the true attack in the Sandworm dataset (Fig. 9), Commander could not identify the attacker's activities on the Workstation_2, the Domain Controller and the Linux Server in the IT domain. The whole network backward tracing from the OT domain, where the initial authentication anomaly was detected, could only lead to the Workstation_1 in the IT domain. This is because, in the Sandworm emulation plan, the attacker has gained three initial accesses (edges labeled as 1, 2 and 3, in Fig. 9) on different machines during the engagement, instead of an initial access on one machine. Since no system logs from the attacker's machine are present, Commander's logon session-based tracing cannot causally link the three initial accesses from the attacker's machine. Note that Commander cannot detect initial access, nor can any other PIDS. Rather, it adds network sockets with an external IP address in the reconstructed attack graphs as an annotation, suggesting where the initial access could come from. For the sake of simplicity, we only show one such network socket in Fig. 7 and Fig. 12. Commander does not perform backward tracing on an edge connecting to a network socket, as, in Section 4, we show that network connection based-tracing would inevitably lead to the dependency explosion problem.

Moreover, although Linux distributions also implement the concept of logon sessions, popular Linux logging tools like Auditd [95] and CamFlow [114] do not insert logon session ID into system log events, nor do any other Linux logging tools we are aware of. That is, Commander cannot perform logon session-based tracing on Linux system logs yet. We leave it to future work (1) to extend existing Linux logging tools with means for embedding logon session ID into system logs, (2) to design a sound scheme for accurately associating initial accesses on different machines from the same attacker.

Another limitation of Commander is its reliance on logs & log integrity. Like *any* other detection system that relies on logs, i.e., every system listed in Section 3, our system Commander would likely fail at detecting the attacks, if attackers have managed to manipulate critical logs without leaving any trace, or there are no logs generated or left at all. This would be a very effective evasion technique for attackers to bypass our system Commander. However, as mentioned in Section 7, there are already efforts being put into preventing log manipulation. For instance, Windows introduced *Protected Event Logging* [96] to secure logs from unauthorized access. Nevertheless, we believe that more log protection techniques need to be proposed in future study in case attackers can bypass existing ones.

A further limitation of system log-based detection systems like Commander is the demand for detailed system logs, which require more computational resources to collect and process. On the one hand, this presents a challenge for resource-constrained devices. On the other hand, it may take longer to detect complex attacks that spread over many machines. Therefore, it is important for future study to investigate techniques that can enhance logging efficiency. For instance, one may selectively, dynamically log system activities based on the system environment, in order to reduce the amount of likely non-essential system logs.

9.1. *Generality of* Commander

We developed Commander with considerations of its deployability in real world in mind, in order to avoid possible deployment-related restrictions in practice. First, Commander leverages system logs and authentication & logon logs collected by popular industry-standard instrumentation-free logging frameworks. Second, Commander's whole network tracing rests on a general concept called logon session-based execution partitioning and tracing, which is not tied to any specific environment. Third, Commander's detection rules are based on the MITRE ATT&CK Matrix [8], which is a widely recognized framework esteemed by organizations worldwide. Leading security vendors, therefore, not only contribute to this framework but also utilize it as a reference for the development of detection mechanisms. Fourth, Commander interfaces with Elasticsearch, a widely utilized tool for event searching and threat analysis within organizations. Last, Commander is evaluated on a typical organization network as specified in the MITRE emulation plans. These characteristics should make Commander easily deployable and adaptable in various environments.

9.2. Integration of machine learning models into Commander

Currently, Commander is a heuristic-based system. So are all integrated detectors in Commander's preliminary detection stage. Future study can explore the possibilities of enhancing Commander's robustness by incorporating machine learning-based detectors, in particular graph neural network-based ones, in case APT actors find new evasion techniques or tactics. However, we believe that only when a machine learning approach can accurately learn an attack technique's semantics, it would truly outperform heuristic-based approaches. We refer readers to [25] for a case study-based comparison between a heuristic-based approach and graph neural network-based approaches for detecting a specific evasion tactic: persistence. It shows that state-of-the-art machine learning-based systems [23,24] fail to learn persistence attacks' semantics, and hence cannot outperform the heuristic-based approach.

9.3. Deploying Commander *as a real-time detection system*

Our system Commander has been tested in an emulated environment, but not yet deployed in a production system. There is a concern that the detailed logs required by Commander would violate users' privacy, especially because, by combining recorded system activities and logon activities, Commander can reveal which employee has carried out what activities at what time. A proper log anonymization mechanism needs to be designed in future work and incorporated into Commander.

Deploying Commander as a real-time detection system in a production system requires the capability of real-time log ingestion, which can be easily realized, as Commander interfaces with Elasticsearch and relies on Auditbeat [115] and Winlogbeat [116] to ship Linux and Windows logs, respectively, to Elasticsearch. These log ingestion tools provided by Elastic are designed to ship logs off of endpoints as soon as possible. With real-time log ingestion, graphs created by Commander will be dynamically updated. A critical step of Commander's graph creation or update is the query process for finding relevant events and filtering out irrelevant ones. Although Elasticsearch provides scalable and near real-time search for log query, the cumulative computing resources required cannot be underestimated. That is, operators need to strike the right balance between the available computing resources and the graph update frequency according to the level of real-time requirement, when deploying Commander.

10. Conclusion & future work

In this paper, we present Commander, a novel PIDS for detecting long-running APT campaigns with the so-called low-and-slow strategy in industrial-sector organizations. Commander's modular design allows it to incorporate specialized detectors for evasion attack techniques, ensuring robust and accurate whole network tracing. We evaluate Commander on two extended MITRE emulation plans, featuring APT groups most known for their attacks against industrial-sector organizations in the past. Our evaluation results demonstrate that Commander can accurately and efficiently detect cross-machine multi-phase APT attacks, and reconstruct concise and insightful attack graphs, crucial for understanding the intrusions and proper attack recovery & remediation. We believe that Commander's ability to perform robust whole network tracing could make it a valuable asset for XDR (eXtended Detection and Response) systems.

In future work, we plan to improve existing Linux auditing frameworks by including logon session ID into system logs emitted by them. Moreover, we aim to create a reliable approach for associating initial accesses on different machines in the same network from the same attacker.

CRediT authorship contribution statement

Qi Liu: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Kaibin Bao:** Writing – review & editing, Supervision, Resources, Project administration. **Veit Hagenmeyer:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by funding of the Helmholtz Association (HGF) through the Energy System Design (ESD) program.

Data availability

Data will be made available on request.

References

[1] CrowdStrike, Inc. CrowdStrike 2023 global threat report. 2023, URL https://www.crowdstrike.com/global-threat-report/.

[2] CrowdStrike, Inc. CrowdStrike 2023 threat hunting report. 2023, URL https://www.crowdstrike.com/resources/reports/threat-hunting-report/.

[3] Shastri V. Attackers set sights on active directory: Understanding your identity exposure. 2023, URL https://www.crowdstrike.com/blog/attackers-set-sights-on-active-directory-understanding-your-identity-exposure/. [Accessed: Dec. 2023].

[4] Shastri V. Endpoint and identity security: A critical combination to stop modern attacks. 2023, URL https://www.crowdstrike.com/blog/unifying-endpoint-and-identity-security/. [Accessed: Dec. 2023].

[5] Dragos. Credentials across IT and OT environments. 2024, URL https://www.dragos.com/blog/minimizing-the-consequences-of-shared-credentials-across-it-and-ot-environments/. [Accessed: July 2024].

[6] Falcon OverWatch Team. 8 LOLBins every threat hunter should know. 2023, URL https://www.crowdstrike.com/blog/8-lolbins-every-threat-hunter-should-know/. [Accessed: May 2023].

[7] Trellix. Trellix threat report 2023. 2023, URL https://www.trellix.com/advanced-research-center/threat-reports/feb-2023/.

[8] The MITRE Corporation. MITRE matrix. 2023, URL https://attack.mitre.org/matrices/enterprise/. [Accessed: Jan. 2023].

[9] Alahmadi BA, Axon L, Martinovic I. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In: USeNIX security symposium. 2022, p. 2783–800.

[10] Hossain MN, Milajerdi SM, Wang J, Eshete B, Gjomemo R, Sekar R, et al. SLEUTH: Real-time attack scenario reconstruction from COTS audit data. In: USeNIX security symposium. 2017, p. 487–504.

[11] Hassan WU, Mark L, Aguse N, Bates A, Moyer T. Towards scalable cluster auditing through grammatical inference over provenance graphs. In: Network and distributed system security. NDSS, 2018, p. 1–15.

[12] Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan VN. HOLMES: Real-time APT detection through correlation of suspicious information flows. In: IEEE symposium on security and privacy (S&P). 2019, p. 1137–52.

[13] Hassan WU, Guo S, Li D, Chen Z, Jee K, Li Z, et al. NoDoze: Combatting threat alert fatigue with automated provenance triage. In: Network and distributed system security. NDSS, 2019, p. 1–15.

[14] Hassan WU, Bates A, Marino D. Tactical provenance analysis for endpoint detection and response systems. In: IEEE symposium on security and privacy (S&P). 2020, p. 1172–89.

[15] Hossain MN, Sheikhi S, Sekar R. Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In: IEEE symposium on security and privacy (S&P). 2020, p. 1139–55.

[16] Hassan WU, Noureddine MA, Datta P, Bates A. OmegaLog: High-fidelity attack investigation via transparent multi-layer log analysis. In: Network and distributed system security. NDSS, 2020, p. 1–16.

[17] Xiong C, Zhu T, Dong W, Ruan L, Yang R, Cheng Y, et al. Conan: A practical real-time APT detection system with high accuracy and efficiency. IEEE Trans Dependable Secur Comput 2022;19(1):551–65. http://dx.doi.org/10.1109/TDSC.2020.2971484.

[18] Irshad H, Ciocarlie G, Gehani A, Yegneswaran V, Lee KH, Patel J, et al. TRACE: Enterprise-wide provenance tracking for real-time APT detection. IEEE Trans Inf Forensics Secur 2021;16:4363–76.

[19] Han X, Pasquier T, Bates A, Mickens J, Seltzer M. UNICORN: Runtime provenance-based detector for advanced persistent threats. In: Network and distributed system security. NDSS, 2020, p. 1–18.

[20] Zeng J, Wang X, Liu J, Chen Y, Liang Z, Chua T-S, et al. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In: IEEE symposium on security and privacy (S&P). 2022, p. 489–506.

[21] Alsaheel A, Nan Y, Ma S, Yu L, Walkup G, Celik ZB, et al. ATLAS: A sequence-based learning approach for attack investigation. In: USeNIX security symposium. 2021, p. 3005–22.

[22] Dong F, Wang L, Nie X, Shao F, Wang H, Li D, et al. DISTDET: A Cost-Effective distributed cyber threat detection system. In: USeNIX security symposium. 2023, p. 6575–92.

[23] Cheng Z, Lv Q, Liang J, Wang Y, Sun D, Pasquier T, et al. KAIROS: Practical intrusion detection and investigation using whole-system provenance. In: IEEE symposium on security and privacy (S&P). 2024, p. 9–28.

[24] Rehman M, Ahmadi H, Hassan W. FLASH: A comprehensive approach to intrusion detection via provenance graph representation learning. In: IEEE symposium on security and privacy (S&P). 2024, p. 142–61.

[25] Liu Q, Shoaib M, Rehman MU, Bao K, Hagenmeyer V, Hassan WU. Accurate and scalable detection and investigation of cyber persistence threats. 2024, arXiv:2407.18832. URL https://arxiv.org/abs/2407.18832.

[26] Liu Q, Bao K, Hassan WU, Hagenmeyer V. HADES: Detecting active directory attacks via whole network provenance analytics. 2024, arXiv:2407.18858. URL https://arxiv.org/abs/2407.18858.

[27] Liu Q, Bao K, Hagenmeyer V. AVIATOR: A MITRE emulation plan-derived living dataset for Advanced Persistent Threat detection and investigation. In: Proceedings of 2024 IEEE international conference on big data. 2024, p. 5592–601.

[28] The MITRE Corporation. MITRE ATT&CK. 2023, URL https://attack.mitre.org. [Accessed: Jan. 2023].

[29] The MITRE Corporation. Persistence. 2024, URL https://attack.mitre.org/tactics/TA0003/. [Accessed: June 2024].

[30] The MITRE Corporation. T1563. 2024, URL https://attack.mitre.org/techniques/T1563/. [Accessed: June 2024].

[31] The MITRE Corporation. T1090.001. 2024, URL https://attack.mitre.org/techniques/T1090/001/. [Accessed: June 2024].

[32] Siemens. Digital protection relays and control - SIPROTEC 5. 2024, URL https://www.siemens.com/global/en/products/energy/energy-automation-and-smart-grid/protection-relays-and-control/siprotec-5.html. [Accessed: June 2024].

[33] Hitachi Energy. RTU500 series function and software. 2024, URL https://www.hitachienergy.com/products-and-solutions/substation-automation-protection-and-control/products/remote-terminal-units/rtu500-series-function-and-software. [Accessed: June 2024].

[34] The MITRE Corporation. ICS matrix. 2024, URL https://attack.mitre.org/matrices/ics/. [Accessed: June 2024].

[35] The MITRE Corporation. MITRE adversary emulation library. 2023, URL https://github.com/center-for-threat-informed-defense/adversary_emulation_library. [Accessed: Jan. 2023].

[36] The MITRE Corporation. Sandworm emulation plan. 2023, URL https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/sandworm. [Accessed: Feb. 2023].

[37] The MITRE Corporation. Oilrig emulation plan. 2023, URL https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/oilrig. [Accessed: Oct. 2023].

[38] NIST. Guide to industrial control systems (ICS) security. 2024, URL https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf. [Accessed: July 2024].

[39] Mathezer S. Introduction to ICS security part 3. 2024, URL https://www.sans.org/blog/introduction-to-ics-security-part-3/. [Accessed: July 2024].

[40] 6sense. Identity and access management. 2024, URL https://6sense.com/tech/identity-and-access-management. [Accessed: July 2024].

[41] Krishnamoorthi S, Carleton J. Active directory holds the keys to your kingdom, but is it secure? 2020, URL https://www.frost.com/frost-perspectives/active-directory-holds-the-keys-to-your-kingdom-but-is-it-secure/.

[42] Security W. Cybersecurity in the AVEVA enterprise SCADA product – going deep. 2024, URL https://waterfall-security.com/ot-insights-center/ot-security-standards/cybersecurity-in-the-aveva-enterprise-scada-product-going-deep-jake-hawkes-episode-122/. [Accessed: July 2024].

[43] Liu Q, Bao K, Hagenmeyer V. Binary exploitation in industrial control systems: Past, present and future. IEEE Access 2022;10:48242–73. http://dx.doi.org/10.1109/ACCESS.2022.3171922.

[44] Nmap. 2024, https://nmap.org/. [Last accessed: May, 2024].

[45] Microsoft. Setspn. 2024, URL https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc731241(v=ws.11). [Accessed: Jan. 2024].

[46] Ho G, Dhiman M, Akhawe D, Paxson V, Savage S, Voelker GM, et al. Hopper: Modeling and detecting lateral movement. In: USeNIX security symposium. 2021, p. 3093–110.

[47] King IJ, Huang HH. Euler: Detecting network lateral movement via scalable temporal link prediction. In: Network and distributed system security. NDSS, 2022, p. 1–16.

[48] Husák M, Yang SJ, Khoury J, Klisura Đ, Bou-Harb E. Unraveling network-based pivoting maneuvers: Empirical insights and challenges. In: Goel S, Nunes de Souza PR, editors. Digital forensics and cyber crime. Cham: Springer Nature Switzerland; 2024, p. 132–51.

[49] Ho G, Sharma A, Javed M, Paxson V, Wagner D. Detecting credential spearphishing in enterprise settings. In: USeNIX security symposium. 2017, p. 469–85.

[50] Novo C, Morla R. Flow-based detection and proxy-based evasion of encrypted malware C2 traffic. In: Proc. ACM workshop on artificial intelligence and security. 2020, p. 83–91.

[51] Alageel A, Maffeis S. Hawk-eye: holistic detection of APT command and control domains. In: Proceedings of the 36th annual ACM symposium on applied computing. SAC '21, 2021, p. 1664–73. http://dx.doi.org/10.1145/3412841.3442040.

[52] Ozery Y, Nadler A, Shabtai A. Information based heavy hitters for real-time DNS data exfiltration detection. In: Network and distributed system security. NDSS, 2024, p. 1–15.

[53] Oprea A, Li Z, Yen T-F, Chin SH, Alrwais S. Detection of early-stage enterprise infection by mining large-scale log data. In: 2015 45th annual IEEE/iFIP international conference on dependable systems and networks. 2015, p. 45–56. http://dx.doi.org/10.1109/DSN.2015.14.

[54] Wilkens F, Ortmann F, Haas S, Vallentin M, Fischer M. Multi-stage attack detection via kill chain state machines. In: Proceedings of the 3rd workshop on cyber-security arms race. CYSARM '21, 2021, p. 13–24. http://dx.doi.org/10.1145/3474374.3486918.

[55] Ah-Fat P, Huth M, Mead R, Burrell T, Neil J. Effective detection of credential thefts from windows memory: Learning access behaviours to local security authority subsystem service. In: 23rd international symposium on research in attacks, intrusions and defenses (RAID 2020). San Sebastian; 2020, p. 181–94.

[56] Han W, Xue J, Wang Y, Zhang F, Gao X. APTMalInsight: Identify and cognize APT malware based on system call information and ontology knowledge framework. Inform Sci 2021;546:633–64. http://dx.doi.org/10.1016/j.ins.2020.08.095, URL https://www.sciencedirect.com/science/article/pii/S0020025520308628.

[57] Shenderovitz G, Nissim N. Bon-APT: Detection, attribution, and explainability of APT malware using temporal segmentation of API calls. Comput Secur 2024;142:103862. http://dx.doi.org/10.1016/j.cose.2024.103862, URL https://www.sciencedirect.com/science/article/pii/S0167404824001639.

[58] Dong F, Li S, Jiang P, Li D, Wang H, Huang L, et al. Are we there yet? An industrial viewpoint on provenance-based endpoint detection and response tools. In: ACM conference on computer and communications security. CCS, 2023, p. 2396–410.

[59] Xu Z, Wu Z, Li Z, Jee K, Rhee J, Xiao X, et al. High fidelity data reduction for big data security dependency analyses. In: ACM conference on computer and communications security. CCS, 2016, p. 504—-516.

[60] Tang Y, Li D, Li Z, Zhang M, Jee K, Xiao X, et al. Nodemerge: Template based efficient data reduction for big-data causality analysis. In: ACM conference on computer and communications security. CCS, 2018, p. 1324—-1337.

[61] Hossain MN, Wang J, Sekar R, Stoller SD. Dependence-Preserving data compaction for scalable forensic analysis. In: USeNIX security symposium. 2018, p. 1723–40.

[62] Michael N, Mink J, Liu J, Gaur S, Hassan WU, Bates A. On the forensic validity of approximated audit logs. In: Proceedings of the 36th annual computer security applications conference. ACSAC '20, 2020, p. 189–202. http://dx.doi.org/10.1145/3427228.3427272.

[63] Fei P, Li Z, Wang Z, Yu X, Li D, Jee K. SEAL: Storage-efficient causality analysis on enterprise logs with query-friendly compression. In: 30th USeNIX security symposium (USeNIX security 21). USENIX Association; 2021, p. 2987–3004, URL https://www.usenix.org/conference/usenixsecurity21/presentation/fei.

[64] Ding H, Yan S, Zhai J, Ma S. ELISE: A storage efficient logging system powered by redundancy reduction and representation learning. In: USeNIX security symposium. 2021, p. 3023–40.

[65] Lee KH, Zhang X, Xu D. High accuracy attack provenance via binary-based execution partition. In: Network and distributed system security. NDSS, 2013, p. 1–16.
[66] Ma S, Lee KH, Kim CH, Rhee J, Zhang X, Xu D. Accurate, low cost and instrumentation-free security audit logging for Windows. In: Annual computer security applications conference. ACSAC, 2015, p. 401–10.
[67] Ma S, Zhang X, Xu D. ProTracer: Towards practical provenance tracing by alternating between logging and tainting. In: Network and distributed system security. NDSS, 2016, p. 1–15.
[68] Ma S, Zhai J, Wang F, Lee KH, Zhang X, Xu D. MPI: Multiple perspective attack investigation with semantic aware execution partitioning. In: USeNIX security symposium. 2017, p. 1111–28.
[69] Keromytis AD. DARPA transparent computing E3. 2023, URL https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md. [Accessed: Sept. 2023].
[70] Manzoor E, Milajerdi SM, Akoglu L. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. KDD '16, 2016, p. 1035–44. http://dx.doi.org/10.1145/2939672.2939783.
[71] Torrey J. DARPA transparent computing. 2023, URL https://github.com/darpa-i2o/Transparent-Computing. [Accessed: Sept. 2023].
[72] Chen T, Dong C, Lv M, Song Q, Liu H, Zhu T, et al. APT-kgl: An intelligent APT detection system based on threat knowledge and heterogeneous provenance graph learning. IEEE Trans Dependable Secur Comput 2022;1–15. http://dx.doi.org/10.1109/TDSC.2022.3229472.
[73] Zeng J, Chua ZL, Chen Y, Ji K, Liang Z, Mao J. WATSON: Abstracting behaviors from audit logs via aggregation of contextual semantics. In: Network and distributed system security. NDSS, 2021, p. 1–18.
[74] Wang S, Wang Z, Zhou T, Sun H, Yin X, Han D, et al. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. IEEE Trans Inf Forensics Secur 2022;17:3972–87.
[75] Fang P, Gao P, Liu C, Ayday E, Jee K, Wang T, et al. Back-Propagating system dependency impact for attack investigation. In: 31st USeNIX security symposium (USeNIX security 22). Boston, MA; 2022, p. 2461–78.
[76] Yang F, Xu J, Xiong C, Li Z, Zhang K. Prographer: An anomaly detection system based on provenance graph embedding. In: USeNIX security symposium. 2023, p. 4355–72.
[77] Jia Z, Xiong Y, Nan Y, Zhang Y, Zhao J, Wen M. MAGIC: Detecting advanced persistent threats via masked graph representation learning. 2023, arXiv:2310.09831.
[78] SigmaHQ. Sigma. 2023, URL https://github.com/SigmaHQ/sigma. [Accessed: Sept. 2023].
[79] Microsoft. LSA logon sessions. 2024, URL https://learn.microsoft.com/en-us/windows/win32/secauthn/lsa-logon-sessions. [Accessed: Feb. 2024].
[80] Microsoft. Security auditing. 2023, URL https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/security-auditing-overview. [Accessed: May 2023].
[81] The MITRE Corporation. MITRE T1547.001. 2023, URL https://attack.mitre.org/techniques/T1547/001/. [Accessed: May 2023].
[82] The MITRE Corporation. MITRE T1543.003. 2023, URL https://attack.mitre.org/techniques/T1543/003/. [Accessed: Feb. 2023].
[83] Koecher I. Are disconnected RDP sessions ticking time bombs in your network?. 2024, URL https://www.eventsentry.com/blog/2022/04/are-disconnected-rdp-sessions-ticking-time-bombs-in-your-network.html. [Accessed: June 2024].
[84] Pamnani V. 4778(S): A session was reconnected to a Window Station. 2024, URL https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4778. [Accessed: June 2024].
[85] Pamnani V. 4800(S): The workstation was locked.. 2024, URL https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4800. [Accessed: June 2024].
[86] The MITRE Corporation. MITRE T1550.002. 2023, URL https://attack.mitre.org/techniques/T1550/002/. [Accessed: Dec. 2023].
[87] Russinovich M. Psexec. 2024, URL https://learn.microsoft.com/en-us/sysinternals/downloads/psexec. [Accessed: June 2024].
[88] The MITRE Corporation. T1136.001. 2024, URL https://attack.mitre.org/techniques/T1136/001/. [Accessed: June 2024].
[89] Segal E. Alert fatigue. 2023, URL https://www.forbes.com/sites/edwardsegal/2021/11/08/alert-fatigue-can-lead-to-missed-cyber-threats-and-staff-retentionrecruitment-issues-study/?sh=4c96871035c9. [Accessed: Aug. 2023].
[90] Elastic NV. Elasticsearch. 2023, URL https://www.elastic.co/. [Accessed: Sept. 2023].
[91] Elastic NV. EQL search. 2023, URL https://www.elastic.co/guide/en/elasticsearch/reference/current/eql.html. [Accessed: Sept. 2023].
[92] NetworkX developers. NetworkX. 2023, URL https://networkx.org/. [Accessed: Sept. 2023].
[93] West Health Institute. PyVis. 2023, URL https://pyvis.readthedocs.io/en/latest/. [Accessed: Sept. 2023].
[94] Russinovich M, Garnier T. System monitor. 2023, URL https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon. [Accessed: Feb. 2023].
[95] Grubb S. The Linux audit daemon. 2023, URL https://linux.die.net/man/8/auditd. [Accessed: Feb. 2023].
[96] Wheeler S, Lombardi M. About logging windows. 2023, URL https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_logging_windows?view=powershell-7.3. [Accessed: April 2023].
[97] Red Hat, Inc. Red hat security guide. 2023, URL https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/chap-system_auditing. [Accessed: Feb. 2023].
[98] van Opstal M, Arbaugh W. DARPA OpTC. 2023, URL https://github.com/FiveDirections/OpTC-data. [Accessed: Sept. 2023].
[99] Hamilton BA. when the lights went out. 2024, URL https://www.boozallen.com/content/dam/boozallen/documents/2016/09/ukraine-report-when-the-lights-went-out.pdf. [Accessed: June 2024].
[100] Cherepanov A. WIN32/indUStroyer: A new threat for industrial control systems. 2024, URL https://web-assets.esetstatic.com/wls/2017/06/Win32_Industroyer.pdf. [Accessed: June 2024].
[101] Slowik J. Anatomy of an attack: Detecting and defeating crashoverride. 2024, URL https://www.dragos.com/wp-content/uploads/CRASHOVERRIDE2018.pdf. [Accessed: June 2024].
[102] Mandiant. Sandworm disrupts power in Ukraine using a novel attack against operational technology. 2024, URL https://cloud.google.com/blog/topics/threat-intelligence/sandworm-disrupts-power-ukraine-operational-technology/. [Accessed: June 2024].
[103] The MITRE Corporation. OilRig. 2024, URL https://attack.mitre.org/groups/G0049/. [Accessed: June 2024].
[104] Mandiant. New targeted attack in the middle east by APT34. 2024, URL https://cloud.google.com/blog/topics/threat-intelligence/targeted-attack-in-middle-east-by-apt34/. [Accessed: June 2024].
[105] Dragos Threat Intelligence. CHRYSENE threat group operations. 2024, URL https://www.dragos.com/threat/chrysene/. [Accessed: June 2024].
[106] Liu L, Engelen G, Lynar T, Essam D, Joosen W. Error prevalence in NIDS datasets: A case study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In: 2022 IEEE conference on communications and network security. CNS, 2022, p. 254–62. http://dx.doi.org/10.1109/CNS56114.2022.9947235.
[107] Liu Q, Bao K, Hagenmeyer V. AVIATOR dataset. 2024, URL https://gitlab.kit.edu/kit/iai/rsa/aviator. [Accessed: November 2024].
[108] Elastic. Elastic detection rules. 2023, URL https://github.com/elastic/detection-rules. [Accessed: Sept. 2023].
[109] Google Security Operations. Chronicle detection rules. 2023, URL https://github.com/chronicle/detection-rules. [Accessed: Sept. 2023].
[110] Mirolyubov E, Taggett M, Hinner F, Patel N. Magic quadrant for endpoint protection platforms. 2023, URL https://www.gartner.com/doc/reprints?id=1-2FFCXFOM&ct=231025&st=sb.
[111] Mellen A. The forrester new wave: Extended detection and response (XDR) providers, Q4 2021. 2021, URL https://www.forrester.com/report/the-forrester-new-wave-tm-extended-detection-and-response-xdr-providers-q4-2021/RES176400.
[112] The MITRE Corporation. Golden ticket. 2024, URL https://attack.mitre.org/techniques/T1558/001/. [Accessed: Jan. 2024].
[113] The MITRE Corporation. T1546.003. 2024, URL https://attack.mitre.org/techniques/T1546/003/. [Accessed: June 2024].
[114] Pasquier T, Han X, Goldstein M, Moyer T, Eyers D, Seltzer M, et al. Practical whole-system provenance capture. In: Symposium on cloud computing (SoCC'17). ACM; 2017.
[115] Elastic NV. Lightweight shipper for audit data. 2023, URL https://www.elastic.co/beats/auditbeat/. [Accessed: June 2023].
[116] Elastic NV. Lightweight shipper for windows event logs. 2023, URL https://www.elastic.co/beats/winlogbeat/. [Accessed: June 2023].