HADES: Detecting and Investigating Active Directory Attacks via Whole Network Provenance Analytics

Qi Liu, Kaibin Bao, Wajih Ul Hassan, and Veit Hagenmeyer, *Member, IEEE*

***Abstract*—Due to its crucial role in identity and access management in modern enterprise networks, Active Directory (AD) is a top target of Advanced Persistence Threat (APT) actors. Conventional intrusion detection systems (IDS) excel at identifying malicious behaviors caused by malware, but often fail to detect stealthy attacks launched by APT actors. Recent advance in provenance-based IDS (PIDS) shows promises by exposing malicious system activities in causal attack graphs. However, existing approaches are restricted to intra-machine tracing, and unable to reveal the scope of attackers' traversal inside a network. We propose HADES, the first PIDS capable of performing accurate causality-based cross-machine tracing by leveraging a novel concept called *logon session based execution partitioning* to overcome several challenges in cross-machine tracing. We design HADES as an efficient on-demand tracing system, which performs whole-network tracing only when it first identifies an authentication anomaly signifying an ongoing AD attack, for which we introduce a novel lightweight authentication anomaly detection model rooted in our extensive analysis of AD attacks. To triage attack alerts, we present a new algorithm integrating two key insights we identified in AD attacks. Our evaluations show that HADES outperforms both popular open-source detection systems and a prominent commercial AD attack detector.**

***Index Terms*—Advanced persistence threat detection, active directory security, enterprise security, data provenance analysis, auditing, logging.**

I. Introduction

RECENT CrowdStrike studies [1], [2], [3], [4] show that 80% of modern cyberattacks are identity-driven, i.e., leveraging compromised credentials. After the initial access, attackers increasingly target Microsoft Active Directory for obtaining critical domain credentials, and hence gaining the ability to move laterally in the victim network and escalate privilege. For instance, Kerberoasting attacks [5] increased almost 600% from 2022 to 2023, and Pass-the-Hash attacks [6] increased 200% [2].

Received 15 July 2024; revised 14 September 2025; accepted 16 September 2025. Date of publication 22 September 2025; date of current version 14 January 2026. This work was supported by funding of the Helmholtz Association (HGF) through the Energy System Design (ESD) Program . *(Corresponding author: Qi Liu.)*

Qi Liu, Kaibin Bao, and Veit Hagenmeyer are with the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology (KIT), 76344 Eggenstein-Leopoldshafen, Germany (e-mail: qi.liu@kit.edu; kaibin.bao@kit.edu; veit.hagenmeyer@kit.edu).

Wajih Ul Hassan is with the School of Engineering & Applied Science, University of Virginia, Charlottesville, VA 22904-4740 USA (e-mail: hassan@virginia.edu).

Digital Object Identifier 10.1109/TDSC.2025.3611866

Active Directory (Domain Service) plays a crucial role for identity and access management in modern enterprises' IT infrastructures, with 90% of Fortune 1000 companies relying on it [4], [7]. To advance the attack after the initial access, attackers routinely leverage Active Directory (AD) functionalities in several stages of the cyber-kill-chain, in particular, internal reconnaissance, credential access, lateral movement and privilege escalation. Comparing to network scanning using third-party tools like `nmap` [8], SPN (Service Principle Name) scanning, a form of AD internal reconnaissance, via native Windows programs like `setspn` [9] is much less noisy while achieving the same goal, i.e., identifying potentially vulnerable servers in the network. Hence, advanced attackers like APT (Advanced Persistent Threat) actors primarily rely on Living-Off-the-Land Binaries (LOLBins) [10], [11], which are system pre-installed programs that are often used by system administrators and genuine users. This complicates intrusion detection and poses significant risks to enterprises.

Conventional intrusion detection systems (IDS) often focus on isolated system events, and excel at identifying malware causing a burst of malicious behaviors. Facing APT actors frequently employing LOLBins and the so-called low and slow strategy, these IDS tend to miss those attacks and suffer from a high false negative rate. To overcome this, security vendors resort to the MITRE ATT&CK Matrix [12] as a reference for further creating detection rules. The MITRE ATT&CK Matrix, maintained with contributions from leading industrial security vendors, provides a high-level summary of attack tactics and techniques, including usage of LOLBins. Our evaluation in Section VI shows that the isolated analysis of these IDS create many false alerts due to the fact that LOLBins are programs executed frequently also by genuine users.

Provenance-based intrusion detection systems (PIDS) [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24] emerged as a new solution, stitching causally related system activities by parsing system events like process creation, file access and network socket access into provenance graphs. Given a suspicious event as a starting point, a backward tracing and forward tracing in the provenance graph can expose more related malicious system activities caused by attackers, i.e., the root cause and attack ramifications, respectively. PIDS have proved to be effective in particular in reducing false alarm rates and presenting real attack activities in attack graphs. Attack graphs provide much richer context for security analysts to further investigate the scope of an attack, invaluable for an effective and efficient security operation center (SOC).

However, current PIDS are constrained to intra-machine provenance tracing, and lack the ability to track across machines in an enterprise environment, and accurately reveal the scale of attackers' traversal inside the network, crucial for attack remediation. Cross-machine provenance tracing faces significant challenges due to the notorious dependency explosion problem [25], [26]. While intra-machine dependency explosion occurs for long-running processes, in which each input is conservatively considered causally responsible for all subsequent outputs, and vice versa, cross-machine dependency explosion arises if cross-machine edges are created simply on a network connection basis. Naively connecting two intra-machine provenance graphs, whenever there is a logon event from one machine to another, or even whenever there is a network connection between them [27], would inevitably result in numerous false dependencies, as discussed further in Section IV-B in detail. Recognizing the critical importance of logon session ID, we propose a novel concept named *logon session based execution partitioning and tracing*. By leveraging both authentication & logon logs and system logs, our system HADES is capable of 1) fine-grained cross-machine provenance tracing, 2) alleviating dependency explosion also for intra-machine provenance tracing, 3) drastically reducing log size, 4) automatically pinpointing privilege escalation.

HADES employs a two-stage approach for efficient AD attack detection. Its first stage consists of a light-weight authentication anomaly detection model responsible for identifying potential AD attacks and forwarding its results to HADES's stage two component, which performs logon session-based tracing and attack graph triage. Both our authentication anomaly detection model and attack graph triage algorithm are rooted in our thorough analysis of AD attacks. Through HADES's development, we faced several difficulties in realizing accurate cross-machine tracing. First, depending on remote access type, each authentication & logon process causes varying number of logon events with distinct logon session ID, complicating the identification of the correct session ID. Second, under certain circumstances, system activities in a new logon session are assigned with an existing session ID, causing false dependencies. Third, it is often not possible to disclose the remote access type by examining the current logon event alone. We overcome these challenges by introducing a remote access type inference module, a logon session ID reassignment module, and a logon session linking module in HADES, based on our extensive profiling and analysis of Windows logging frameworks. Our implementation of logon session-based tracing avoids time-consuming instrumentation, error-prone training, and applies to the whole system rather than a single program.

Unlike previous techniques, HADES produces alarms satisfying all five properties of reliability, explainability, analytical depth, contextuality, and transferability as introduced in [28]. In its first stage, HADES's anomaly detection model avoids easily changeable indicators such as hard-coded IP addresses and file hashes, which are typical in popular Security Information and Event Management (SIEM) detection rules [29], [30], ensuring reliable detection. The attack graphs produced in the second stage of HADES are both explainable and contextual, providing an analytical overview of the attack. Finally, the system's customizable weighting factors for indicators introduced in its threat score calculation make HADES highly transferable and adaptable for practical use in various scenarios.

The main contributions of this paper are as follows:

- We give a succinct and contextualized AD attack overview based on a thorough analysis, critical for understanding and detecting AD attacks.
- We present a light-weight authentication anomaly detection model for AD attacks.
- We propose a novel concept called logon session-based execution partitioning and tracing.
- We demonstrate the first accurate and efficient causality-based cross-machine provenance tracing system HADES.
- We introduce a new alert triage algorithm for accurate AD attack detection.

II. BACKGROUND

AD attacks are a set of cyberattacks targeting Microsoft Active Directory service employed in most corporate environments. Half of organizations worldwide have experienced an AD attack in recent years, while 40% of those attacks were successful [4], [31]. AD uses a database to store critical information about organizations' network resources, and provides administrators the ability to manage the access to those resources. That is, AD holds the blueprint of an organization's environment, and the keys to all available resources. Consequently, various kinds of attacks specifically exploiting vulnerabilities in Microsoft's AD design and implementation emerged over time, resulting in a partial or even full-scale compromise of many enterprise networks [32], [33].

Often, an attack is launched first against a targeted user via spear-phishing, or a vulnerable domain-joined machine accessible from outside, to get an initial foothold inside the domain. The attacker then performs internal AD discovery to gain knowledge about the internal network and spot more machines, before moving to them typically via some form of stolen credentials from the first machine.

A proper understanding of the prerequisites of various AD attacks, and logging possibilities as well as limitations is critical for accurate AD attack detection. Fig. 1 shows the AD attack skeleton, in which most relevant AD attack techniques are listed. This model serves as a blueprint for reliably detecting AD attack techniques and reconstructing high-level AD attack graphs. These graphs can be further expanded to create whole network attack graphs, including all malicious system activities conducted by attackers.

III. THREAT MODEL

Like other PIDS [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [34], our system HADES assumes the integrity of the underlying operating systems, firmware and our logging frameworks. We only consider attacks in which threat actors already achieved an initial foothold on a domain-joined host via

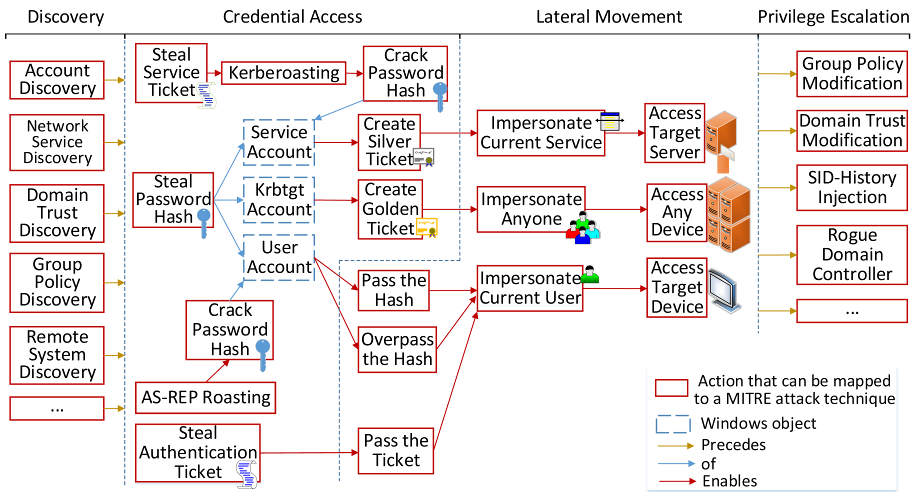


Fig. 1. Active Directory attack overview.

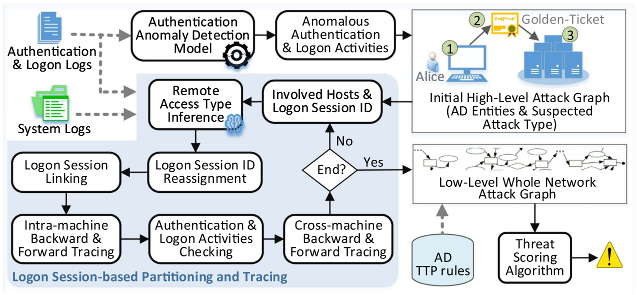

Authentication & Logon Logs
System Logs
Authentication Anomaly Detection Model
Anomalous Authentication & Logon Activities
Golden-Ticket
Alice
Initial High-Level Attack Graph (AD Entities & Suspected Attack Type)
Remote Access Type Inference
Involved Hosts & Logon Session ID
No
End?
Yes
Logon Session Linking
Logon Session ID Reassignment
Low-Level Whole Network Attack Graph
Intra-machine Backward & Forward Tracing
Authentication & Logon Activities Checking
Cross-machine Backward & Forward Tracing
AD TTP rules
Threat Scoring Algorithm
Logon Session-based Partitioning and Tracing

Fig. 2. HADES overview.

binary exploitation or social engineering, and move laterally to other machines leveraging stolen credentials instead of program exploitation. That is, only identity-based remote access types are of interest, in which valid accounts are used to access a computer. Attacks restricted to only one machine are deemed less severe and are hence out of HADES's detection scope.

IV. System Design

Our system HADES employs a two-stage approach for efficient and accurate detection of AD attacks, producing an initial high-level attack graph and a low-level whole network attack graph, respectively. Fig. 2 provides an overview of HADES, whereas a formal description of HADES's detection procedure is given in Algorithm 1. HADES's first stage starts with its authentication anomaly detection, and ends with producing a high-level attack graph. HADES's second stage starts with its logon session-based partitioning and tracing, and ends with producing a low-level whole network attack graph with assigned threat score. To enhance readability of Algorithm 1, we provide a short description for symbols and functions that are not self-explanatory in Table I.

In its first stage, HADES relies on authentication & logon logs only, and parses through each AD authentication & logon log event. It traces backward on each logon event, and traces backward & forward on each authentication event (Algorithm 1 Lines 24-25), as authentication is a multiple-step process. The core component of HADES's stage 1 is a light-weight authentication anomaly detection model, against which the authentication & logon tracing result is matched. Once an anomaly is found, HADES produces a high-level attack graph including involved AD entities (Algorithm 1 Line 5), i.e., users and machines, and labeled with a suspected AD attack type (Algorithm 1 Line 27), at the end of its stage 1. A concrete example of such high-level attack graphs is given in Fig. 3, in which 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, mimicking the Pass-the-Hash behavior.

The identified host names, user names and their logon session ID are passed to HADES's stage 2, in which it performs logon session-based execution partitioning and tracing. On the accessed host, e.g., the Data Server in Fig. 3, by leveraging both authentication & logon logs and system logs, HADES first infers the remote access type (Algorithm 1 Line 7), which is critical for deciding whether the logon session ID needs to be reassigned in the next step (Algorithm 1 Line 8). Then it links related logon sessions resulted from the same identity (Algorithm 1 Line 9), and performs intra-machine backward & forward tracing inside these logon sessions (Algorithm 1 Lines 10-11). Afterwards, HADES checks the authentication & logon logs to spot any logon event inside any domain-joined machine initiated from the accessed host, i.e., cross-machine forward tracing (Algorithm 1 Line 14). Meanwhile, HADES examines the authentication & logon logs to find out whether and from which domain-joined machine the current logon session inside the accessing host, e.g., the Exchange Server in Fig. 3, is initiated, i.e., cross-machine backward tracing. After the logon session-based tracing ends as no further involved machine can be found, HADES passes the low-level whole network provenance graph to its threat scoring algorithm (Algorithm 1 Line 14), the final step of stage 2. In this step, we leverage open-source TTP (Tactics, Techniques,

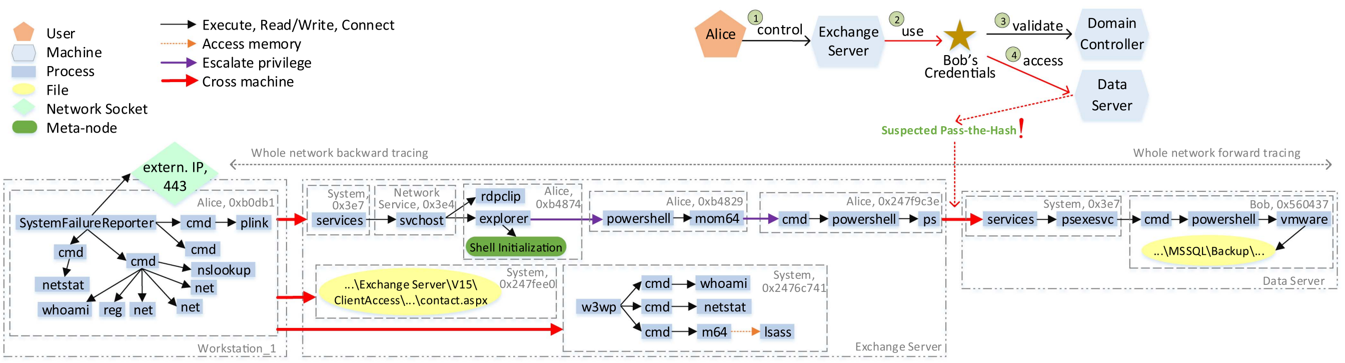


Fig. 3. An AD attack graph created by HADES on the Oilrig [35] dataset. HADES first creates an initial high-level attack graph involving AD entities like users and hosts, after it detects an authentication & logon anomaly and suspects a Pass-the-Hash attack. Then it performs system-level forward tracing inside the specific logon session under user Bob in the accessed host Data Server, and system-level forward & backward tracing inside the logon session of Alice in the accessing host Exchange Server. Subsequently, it traces back to a logon session of Alice in the Workstation_1. Next, it traces forward & backward inside this logon session, leading to another two logon sessions in the Exchange Server. This graph reveals that an attacker leveraged a C2 (Command & Control) agent disguised under the process name SystemFailureReporter on the Workstation_1 to perform AD discovery via LOLBins like `net` and `netstat`. The attacker then pivoted to the Exchange Server, performed further AD discovery, credential access, before moving to the Data Server for accessing critical data.[1].

Procedures) detection rules [30] for AD discovery and credential access (Algorithm 1 Lines 35-36).

In the example of Fig. 3, the whole network forward tracing from the accessed host, i.e., the Data Server, did not lead to further logon session on another machine, as the attacker found the targeted data and did not advance further in the network. However, the whole network backward tracing from the accessing host, i.e., the Exchange Server, reveals that the current logon session on it is a RDP (Remote Desktop Protocol) session that was initiated from another logon session of user Alice inside another machine Workstation_1. Backward tracing from the Workstation_1 did not lead to further domain-joined machine, but rather an external IP address, which belongs to the C2 (Command & Control) server operated by the attacker. Forward tracing from the Workstation_1 resulted in another two logon sessions in the Exchange Server preceding the RDP logon session on it. In one of these logon sessions, credential access was performed, contributing to the later lateral movement from the RDP logon session on the Exchange Server to a logon session on the Data Server. Note that backward tracing can only lead to exactly one logon session, while forwarding tracing can lead to multiple logon sessions, as one may remotely access multiple machines from one machine and multiple logon sessions inside one machine do not interfere with each other.

A. Authentication Anomaly Detection

The main purpose of AD is providing the service of authentication of user or machine accounts and authorization to different network resources. AD employs Kerberos as its default authentication protocol, and implements it in two components in

[1]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, we introduced a graph optimization technique called meta-nodes to conclude system activities of standard known benign routines.

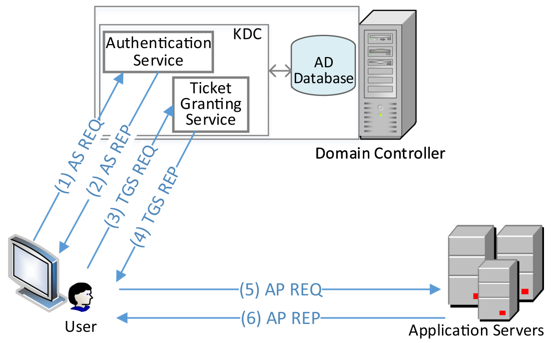


Fig. 4. Standard AD authentication process.

a domain controller (DC), i.e., the Authentication Service (AS) and the Ticket-Granting Service (TGS). Fig. 4 shows a simplified version of standard AD authentication process: 1) the user sends an encrypted AS request for a TGT (Ticket-Granting-Ticket), 2) the DC replies with a TGT if it can decrypt the request and hence verify the user's identity, 3) the user asks for a TGS ticket by providing the TGT, 4) the DC replies with a TGS ticket, 5) the user asks for access to an application server by providing the TGS ticket, 6) the server gives the user requested access after validating the TGS ticket. Note that each communication step is represented in a directed line in Fig. 4. The entire process uses shared secret cryptography to deter network-level eavesdropping, meaning that credential access is only possible inside hosts.

The complicated network authentication process is often exploited by attackers who manage to steal some form of credentials or credentials-related data on one machine, i.e., credential access, and ultimately use them to successfully authenticate against a domain controller and hence access another machine, i.e., lateral movement. Several AD attack techniques exhibit similarities, often causing confusion. However, we believe that a practical detection system needs to differentiate between these

Algorithm 1: Active Directory Attack Detection

```
Inputs : System audit log events $\mathcal{E}$;
         Authentication & logon events $\mathcal{A}$;
         AD TTP rules $\mathcal{R}$
Output: List $L_{<AG,TS>}$ of attack graph and its threat score pairs
1  Function GetADAttackGraph($\mathcal{E}$, $\mathcal{A}$)
       /* Get a list of authentication & logon
          anomalies                                   */
2      $L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>} \leftarrow$ GetAuthenticationAnomaly ($\mathcal{A}$)
3      foreach ($\mathcal{A}\gamma$, $\mathcal{L}\gamma$, $\mathcal{T}\gamma$) $\in L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>}$ do
4          $AG \leftarrow$ null
5          $HG \leftarrow$ CreateHighLevelGraph ($\mathcal{A}\gamma$, $\mathcal{L}\gamma$)
6          $AG \leftarrow AG \cup HG$
7          $AccessType \leftarrow$ CheckRemoteAccessType ($\mathcal{A}\gamma$, $\mathcal{E}$)
8          $SessionID \leftarrow$ ReassignSessionID ($AccessType$, $\mathcal{A}\gamma$, $\mathcal{E}$)
9          $LinkedSessionID \leftarrow$ LinkSessions ($SessionID$, $\mathcal{A}\gamma$, $\mathcal{E}$)
10         ($\mathcal{E}\alpha$, $\mathcal{G}\alpha$) $\leftarrow$ TraverseBackwardSys($SessionID$,
             $LinkedSessionID$, $\mathcal{E}$)
11         ($\mathcal{E}\kappa$, $\mathcal{G}\kappa$) $\leftarrow$ TraverseForwardSys($SessionID$,
             $LinkedSessionID$, $\mathcal{E}$)
12         $AG \leftarrow AG \cup \mathcal{G}\alpha \cup \mathcal{G}\kappa$
13         $\mathcal{E}\alpha \leftarrow \mathcal{E}\alpha \cup \mathcal{E}\kappa$
14         $\mathcal{A}\alpha \leftarrow$ CheckAuthenticationLogon($\mathcal{E}\alpha$, $\mathcal{A}$)
15         if $\mathcal{A}\alpha$ is not null then
16             goto 7
17         end
18         $TS \leftarrow$ GetThreatScore ($AG$)
19         $L_{<AG,TS>} \leftarrow L_{<AG,TS>} \cup (AG, TS)$
20     end
21     return $L_{<AG,TS>}$

22 Function GetAuthenticationAnomaly($\mathcal{A}$)
23     foreach $ae \in \mathcal{A}$ do
24         $\mathcal{A}_\alpha \leftarrow$ TraverseBackwardAuth($ae$)
25         $\mathcal{A}_\kappa \leftarrow$ TraverseForwardAuth($ae$)
26         $\mathcal{A}_\alpha \leftarrow \mathcal{A}_\alpha \cup \mathcal{A}_\kappa$
27         ($\mathcal{L}\alpha$, $\mathcal{T}\alpha$) $\leftarrow$ CheckAttackType($\mathcal{A}_\alpha$)
28         if $\mathcal{L}\alpha \in L_{<\mathcal{L}_{attack}>}$ then
29             $L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>} \leftarrow L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>} \cup (\mathcal{A}\alpha, \mathcal{L}\alpha, \mathcal{T}\alpha)$
30         end
31     end
32     return $L_{<\mathcal{A}\gamma,\mathcal{L}\gamma,\mathcal{T}\gamma>}$

33 Function GetThreatScore($AG$)
34     foreach ($CrossMachineEdge$, $\mathcal{T}\omega$) $\in AG$ do
           /* Get a list of discovery techniques
              and frequency                           */
35         $L_{<tec_d,freq>} \leftarrow$ CheckADDiscovery($AG$, $\mathcal{T}\omega$, $\mathcal{R}$)
36         $L_{<tec_{ca},freq>} \leftarrow$ CheckCredentialAccess($AG$, $\mathcal{T}\omega$, $\mathcal{R}$)
37         $L_{<pe>} \leftarrow$ CheckPrivilegeEscalation($AG$)
38         $TS_{sub} \leftarrow$ CalculateScore($L_{<tec_d,freq>}$,
             $L_{<tec_{ca},freq>}$, $L_{<pe>}$)
39         $L_{<TS_{sub}>} \leftarrow L_{<TS_{sub}>} \cup TS_{sub}$
40     end
41     $TS \leftarrow$ SumScore($L_{<TS_{sub}>}$)
42     return $TS$
```

TABLE I
Description of Symbols & Functions

Symbols/Functions	Description
$\mathcal{A}$	The collection of all authentication & logon log events.
$\mathcal{A}\gamma$, $\mathcal{A}\alpha$, $\mathcal{A}\kappa$	A collection of log events related to a specific authentication & logon anomaly.
ae	A single authentication or logon event.
AG	An attack graph.
$\mathcal{E}$	The collection of all system log events.
$\mathcal{E}\alpha$, $\mathcal{E}\kappa$	A collection of system log events related to some specific logon session(s).
$\mathcal{G}\alpha$, $\mathcal{G}\kappa$	A sub-graph representing system log events related to some specific logon session(s).
HG	A high-level graph.
$L_{<\mathcal{L}_{attack}>}$	The list of AD attack types.
$\mathcal{L}\gamma$, $\mathcal{L}\alpha$	The AD attack type labeled to a specific authentication & logon anomaly.
$\mathcal{R}$	The collection of all AD TTP rules.
$\mathcal{T}\gamma$, $\mathcal{T}\alpha$, $\mathcal{T}\omega$	The time of occurrence of a specific event, e.g., an authentication & logon anomaly.
CheckAD-Discovery	It takes as input the attack graph, the time of occurrence of a cross-machine logon and the set of TTP rules for AD discovery, and then checks how many AD discovery techniques were performed before the cross-machine logon and how often each technique was performed. (More in Section IV-C.)
CheckAuthenticationLogon	It takes as input the collection of system audit log events from some specific logon session(s), and all authentication & logon log events, and then checks if there are authentication & logon events on other machines related to those logon session(s).
Check-AttackType	It takes as input a collection of related authentication & logon events, and then checks if they correspond to a specific AD attack type according to Figure 5.
CheckRemote-AccessType	It ingests a collection of authentication & logon log events, and system audit log events, and then infers the remote access type. (More in Section IV-B3.)
CreateHigh-LevelGraph	It parses a collection of authentication & logon log events into a high-level graph, and labels the graph with an AD attack type.
LinkSessions	It links related logon sessions. (More in Section IV-B5.)
Reassign-SessionID	It reassigns session ID when necessary. (More in Section IV-B4.)
Traverse-BackwardAuth	It takes as input an authentication or logon event, finds related authentication & logon event(s) occurred before the current event. For instance, if the current event represents a TGS request, this function aims to find the corresponding event representing the TGT request. (More in Section IV-A.)
Traverse-BackwardSys	It parses system audit log events, finds the previous logon session that initiated the current logon session. It collects system audit log events in the previous logon session, and parses them into a sub-graph.
Traverse-ForwardAuth	It takes as input an authentication or logon event, finds related authentication & logon event(s) occurred after the current event. For instance, if the current event represents a TGS request, this function aims to find the corresponding event representing the logon. (More in Section IV-A.)
Traverse-ForwardSys	It parses system audit log events, finds the logon session(s) initiated from the current logon session. It collects system audit log events in those logon session(s), and parses them into a sub-graph.

attacks and hence triage the alerts due to the varying degrees of severity of those attacks. For instance, a Golden-Ticket [36] attack is more critical than others since it can lead to a full-scale domain compromise.

We analyzed each attack type, and identified anomaly in the corresponding authentication process, as illustrated in Fig. 5. Note that, in Fig. 5, each attack type is associated with directed lines of a specific color. Like in Fig. 4, each directed line represents a communication step between the user machine and the domain controller or an application server during the authentication process. Fig. 5 demonstrates that, comparing to the standard authentication process shown in Fig. 4, each attack type manifests itself in a specific non-standard, incomplete/abnormal authentication process.

For instance, in AS-REP Roasting attacks (associated with the green directed lines in Fig. 5), attackers only send a TGT request and aim to crack the user's password from the received TGT offline. Hence no TGS request follows. This is demonstrated in Fig. 5, in which only two green directed lines representing the first two communication steps are observed during the authentication process, instead of six directed lines. In Kerberoasting attacks (associated with the purple directed lines

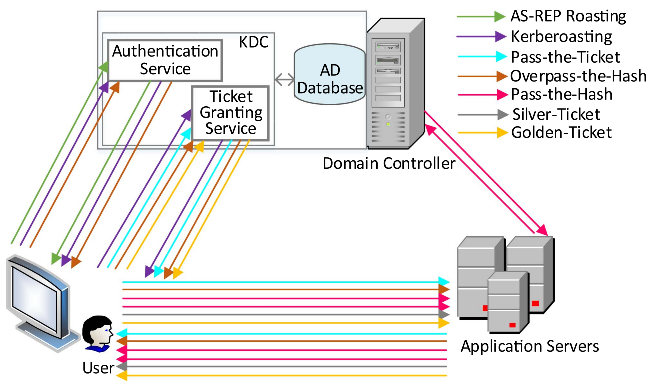


Fig. 5. Authentication incompleteness/abnormality.

in Fig. 5), attackers aim to crack a service account's password from a received TGS, meaning that no access on the application server follows the TGS request. Therefore, only four purple directed lines representing the first four communication steps are observed in Fig. 5. In Pass-the-Ticket attacks (associated with the light blue directed lines in Fig. 5), attackers send a TGS request to the DC with a stolen TGT, meaning that no TGT request precedes the TGS request. Thus, only four light blue directed lines representing the *last* four communication steps are observed in Fig. 5.

Therefore, we propose a light-weight authentication anomaly detection model based on our finding. This anomaly detection model relies on authentication & logon logs only; it parses through each AD authentication & logon log event. It traces backward on each logon event, and traces backward & forward on each authentication event (Algorithm 1 Lines 24-25), as authentication is a multi-step process. By doing so, HADES aims to identify whether there is an authentication anomaly, and which kind of authentication anomaly is present. Based on Fig. 5, HADES labels the authentication anomaly with a suspected AD attack type (Algorithm 1 Line 27). Note that AD authentication logs and logon logs can be linked with a logon GUID (Globally Unique Identifier), which is a critical information for HADES's authentication & logon tracing; a logon GUID is inserted into each AD authentication event and the corresponding logon event by default by Windows specifically for associating these events.

During runtime, HADES checks, for instance, each authentication event with the Windows Event ID 4769, which records each Kerberos service ticket (TGS) request [37], i.e., the step 3) in Fig. 4. Then it extracts the logon GUID from the authentication event, and checks if there is a logon event with Windows Event ID 4624 that contains the same logon GUID, which records each successful logon [38]. If such logon event does not exist, it indicates a Kerberoasting attack (associated with the purple directed lines in Fig. 5). For each authentication event with the Windows Event ID 4769, HADES also checks if there is a corresponding authentication event with the Windows Event ID 4768, which records each Kerberos authentication ticket (TGT) request [39], i.e., the step 1) in Fig. 4. If it does not exist, it indicates a Pass-the-Ticket attack (associated with the light blue directed lines in Fig. 5). HADES's authentication & logon tracing can also start with checking each authentication event with the Windows Event ID 4768. For each such event, i.e., a TGT request, if a corresponding event of Event ID 4769, i.e., a TGS request, does not exist, it indicates a AS-REP Roasting attack (associated with the green directed lines in Fig. 5).

In our evaluation in Section VI, this model proves to be effective in catching each attack instance, but is plagued with a high false positive rate, hindering its adaptability in practice. The reasons for false positives are manifold. For instance, false positives for AS-REP Roasting and Kerberoasting can be resulted from network disruption. Further, legitimate use of native Windows programs like `runas` causes false Pass-the-Hash alerts.

Note that the list of AD attack types involved in the authentication & logon process (in Fig. 5) was constructed based on our extensive analysis of AD attacks observed as of today, and is meant to be exhaustive. Our sources include the MITRE ATT&CK knowledge base [40], threat reports linked in it, and threat reports posted in leading security vendors' websites, like CrowdStrike [1], Mandiant [41], Trellix [11]. However, we cannot exclude the existence of an unknown anomalous authentication & logon sequence that cannot be mapped to our known AD attack types. Such a sequence would also be detected, as it, by definition, would deviate from the standard authentication & logon process illustrated in Fig. 4. It makes sense to flag such a sequence; this would likely further increase the false positive rate. Such a check can be easily implemented in the function CHECKATTACKTYPE (Algorithm 1 Line 27). Nevertheless, during our experiments, we did not observe any anomalous authentication & logon sequence that cannot be mapped to a known AD attack type in Fig. 5. If an anomalous authentication & logon sequence associated with an unseen attack type was observed by our system HADES, it would run its stage 2 for a whole network investigation and reveal attack tactics and techniques employed by the attacker, as it does for known AD attack types. That is, HADES handles each unknown authentication anomaly in the same way as it does for those that are associated with a known AD attack type. HADES would output an AD attack graph similar to Fig. 3; the initial high-level attack graph involving AD entities would be labeled as "Unknown AD Attack".

B. Logon Session-Based Execution Partitioning and Tracing

Recent PIDS have proved to be invaluable in reducing false alarms. To tackle the high rate of false positive, we aim to develop the first causality-based cross-machine PIDS. A naive approach would be to connect the intra-machine provenance graphs of two domain-joined machines with a cross-machine edge, whenever there is a network connection between them, as suggested in [27]. However, this kind of cross-machine edges do not represent a causality,[2] but rather a correlation, leading to numerous cross-machine edges between intra-machine provenance graphs. Take Fig. 3 as an example; this would result in

[2]Causality is defined as the relationship between cause and effect. A causality-based cross-machine edge connects two logon sessions, in which one session initiated the other session via remote access, while a correlation-based edge connects two logon sessions without this relationship.

7364 edges[3] between the Exchange Server's provenance graph and other hosts' graphs after only one day operation, while in fact there are only 4 causality-based cross-machine edges to/from the Exchange Server in the true attack chain, shown as the four thick, red directed lines in Fig. 3.

A more advanced approach would be to connect intra-machine provenance graphs, whenever there is a logon event from one machine to another. Yet, due to the prevalence of credential thefts, in particular in AD attacks, it is impossible to identify the true identity behind each logon. For instance, in Fig. 3, Alice's credentials were stolen by the attacker. Given that both Alice and the attacker have accessed the Exchange Sever from the Workstation_1 using the same credentials, this approach would still create false-dependencies. In fact, this approach would produce 479 correlation-based cross-machine edges,[4] instead of 4 causality-based ones, to/from the Exchange Server. Correlation-inferred edges are the root cause of the notorious dependency explosion problem [25], [42].

In order to enable causality-based provenance tracing, we resort to a critical yet previously undiscovered information: logon session ID, and propose a novel concept: logon session-based execution partitioning and tracing. To the best of our knowledge, this work presents the first PIDS leveraging logon session ID. Windows assigns a logon session ID (unique until next reboot) after each successful authentication & logon to label system activities run in that session in Windows Security logs [43]. A logon session is a computing session assigned with an access token representing the authenticated account's security context and permissions [44]. Every Windows process runs within a logon session's context.

Fig. 6 further illustrates the difference between the 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. 6(a), would lead to a dense graph indiscriminately connecting all system activities as depicted in Fig. 7(a). Logon-based tracing in Fig. 6(b) reduces false dependencies by considering only logon-initiated network connections, resulting in less dense graphs in Fig. 7(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. 6(c), truly linking system activities belonging to the same identity, and separating system activities unrelated to each other into individual provenance graphs shown in Fig. 7(c).

1) Benefits: The benefits of logon session-based tracing are fourfold: 1) it enables fine-grained causality-based

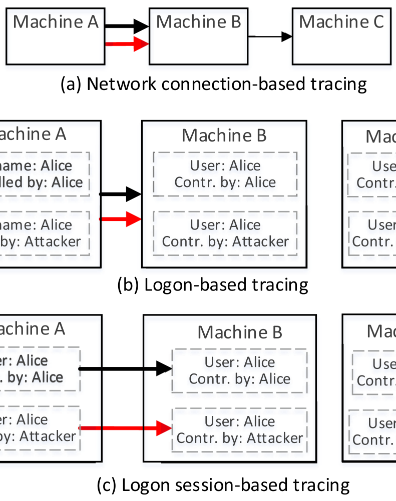


Fig. 6. Comparison of cross-machine tracing approaches. Whereas a thin black edge denotes a network connection, a thick black edge denotes a logon by a genuine user and a thick red edge denotes a logon by an attacker.

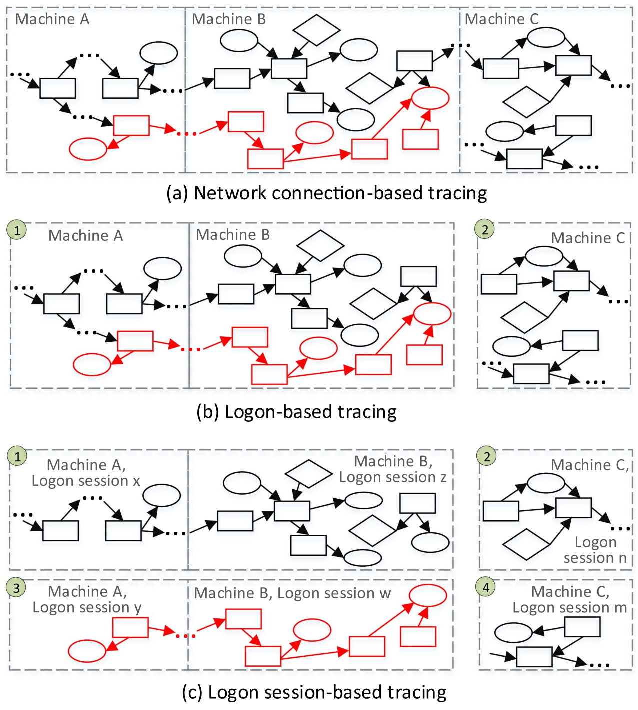

Machine A
Machine B
Machine C
(a) Network connection-based tracing
1
Machine A
Machine B
2
Machine C
(b) Logon-based tracing
1
Machine A, Logon session x
Machine B, Logon session z
2
Machine C, Logon session n
3
Machine A, Logon session y
Machine B, Logon session w
4
Machine C, Logon session m
(c) Logon session-based tracing

Fig. 7. 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.

cross-machine provenance tracing, 2) it alleviates dependency explosion also for intra-machine provenance tracing, 3) it reduces log size as most forensics-irrelevant system activities under logon sessions with predefined logon ID[5] can be safely discarded, 4) it automatically pinpoints privilege escalation during intra-machine provenance tracing.

[3]This number is returned from a query that checks for the number of established network connections. That is, there are 7364 network connections established between the Exchange Server and other machines during the corresponding emulation plan's execution.

[4]This number is returned from a query that checks for the number of succeeded logons. That is, there are 479 logons performed from/to the Exchange Server during the corresponding emulation plan's execution.

[5]Predefined logon ID include `0x3e4`, `0x3e5`, and `0x3e7`, belonging to Network Service account, Local Service account, System account, respectively [45].

When a user from one machine logs into another machine,[6] the corresponding logon session in the second machine is resulted from exactly one logon session in the first machine. For example, in Fig. 3, Alice's logon session `0xb4874` on the Exchange Server is only resulted from Alice's logon session `0xb0db1` on the Workstation_1, while there are multiple Alice's logon sessions on the Workstation_1 in parallel. Our system HADES can accurately disclose this, as it checks authentication & logon logs among involved machines and performs logon session-based tracing. *With logon session-based tracing, we can create a whole-network provenance graph representing activities conducted by exactly one true identity, be it the attacker or a normal user.*

The benefit is even more apparent during intra-machine provenance tracing. An enterprise-level application server typically hosts more than a hundred logon sessions at the same time, with some belonging to the same user, and runs cumulatively thousands of logon sessions after just one day operation. For instance, in the Oilrig [35] dataset, system activities run on the Exchange Server spread over more than 5 thousands logon sessions. While some logon sessions are (interactive) long-running ones, many others die soon after a short execution, e.g., accessing some resources. Oblivious to logon session ID, prior provenance tracing approaches would connect all these system activities, as they all at the end can be traced back to the root process, `system` (PID=4) on Windows. In contrast, our approach identifies the truly involved logon sessions and tracks system activities only inside them, greatly easing the dependency explosion problem.

The efficiency introduced by leveraging logon session ID shows not only during provenance tracing, but also for log storage reduction. Under the assumption of OS integrity, most system activities run under logon sessions with predefined logon ID belong to standard routines, and can be considered as forensics-irrelevant and safely discarded. In the example of the Exchange Server in the Oilrig dataset, these system activities account for over 90% of all log events created on it. However, some of these system activities are forensics-relevant, as they are related to remote access and logons. HADES identifies and tracks these system activities during its logon session linking. Further, as logon sessions also introduce the separation of privileges, HADES automatically identifies privilege escalation during intra-machine cross-session tracing by checking the value of "Token Elevation Type" and "Integrity Level" associated with a logon session ID. By doing so, it has identified, e.g., in Fig. 3, the privilege escalation from unprivileged user to Administrator, i.e., from `0xb4874` to `0xb4829`, and then privilege escalation from Administrator to System, i.e., from `0xb4829` to `0xb247f9c3e`.

2) Challenges: We encountered several challenges while developing HADES. First, each network authentication & logon process results in multiple logon events with distinct logon session ID on the accessed machine. Depending on the remote access type, e.g., via RDP or SMB (Server Message Block), the number of logon sessions created varies. Second, under certain circumstances, system activities resulted from a new logon are recorded with an existing session ID, leading to false dependency. Third, it is often not possible to reveal the remote access type by inspecting the logon event alone.

3) Remote Access Type Inference: Identifying remote access type is critical for logon session ID reassignment and session linking, both crucial for accurate cross-machine tracing. The way how exactly Windows emits logon events and assigns logon ID is obscure. Due to Windows' closed-source nature, revealing this via source code is not possible. Reference to the most detailed sources about Windows [45], [46], [47], [48] also failed to deliver an answer. Hence we resorted to extensive testing and analysis of each remote access type.

We found that both authentication & logon events and system activities events need to be processed to extract a list of attributes for accurately inferring remote access type. Hence, for each logon event, HADES checks related authentication & logon events and system events to analyze the context and then classify the remote access type. HADES can identify all standard remote access types: RDP, SSH, Powershell-Remoting (WinRM), WMI, RPC, PsExec, Network Share (SMB), internal web-server request. Note that only identity-based remote access types are of interest, in which valid accounts are used to access a computer. Remote access via malware/C2 agent is not identity-based, and will not create new logon sessions. Repetitive accesses via the same malware/C2 agent run in the same logon session and their system activities are recorded with the same session ID. That is, as long as HADES can trace to the session in which the C2 agent is present, it can detect all system activities conducted by it, without the need for inferring remote access type and session linking etc. For example, in Fig. 3, the attacker has performed several attack steps via the C2 agent on the Workstation_1 at different times, and all of these steps are detected by HADES, as they are all executed in the same session `0xb0db1` and HADES has successfully traced back to this session.

We also found that some logon events and associated logon sessions are purely byproducts. Some of these sessions terminate soon after being created and others live until a logoff occurs. These logon sessions typically do not contain any forensics-relevant system activities. Although their creations are resulted from a user logon, they are not influenced by the logged-on user. For instance, when a domain user logs into a machine via SSH, a byproduct logon session under a virtual user *sshd_xxx* gets created.

4) Logon Session ID Reassignment: For several remote access types, system activities performed on a new logon session are recorded under an existing logon session ID, necessitating session ID reassignment for causally correct tracing. A typical example is RDP, which is often misused by attackers [49]. Unlike SSH, a remote logon session via RDP in the accessed machine will not terminate, when the client program on the accessing machine exits. The user has to explicitly log out to terminate that session.

When the user's credentials are stolen by an attacker who then logs into the same remote machine with the same credentials, all system activities conducted by the attacker are labeled with

[6]Note that a user accessing resources on a remote machine also triggers a logon on the remote machine. The authentication & logon process is transparent to users due to Single Sign-on.

the ID of the user's long-running session, although a new logon session with a different ID is created. This is due to Fast User Switching [50], which provides users the same setup after reconnecting to an existing local console session or remote desktop session[7]. The new logon session starts with the new logon by the attacker, and ends when the attacker disconnects, marking the exact timespan of system activities conducted only by the attacker. Hence, if HADES identifies a logon as RDP access, it further checks for another specific related event indicating whether the user has entered a new remote desktop session or an existing one. If an existing remote desktop session is being reentered, HADES takes the ID of the new logon session and replaces the existing logon session's ID with it for the corresponding system activities.

Other remote access types requiring session ID reassignment include Powershell-Remoting and internal web-server request. In Fig. 3, the attacker from the Workstation_1 leveraged webshell to execute commands on the Exchange Server. However system activities caused by the attacker's web requests are recorded with System account's logon session ID, i.e., `0x3e7`, along with system activities resulted from other domain users' web requests. HADES correctly identified the remote access type and reassigned the attacker's system activities with `0x2476c741`, separating them from unrelated system activities caused by other users.

Note that the logon session ID `0x2476c741` is a random ID that Windows created for the corresponding internal web request of the attacker; this ID is recorded in the corresponding logon event. Each successful internal web request is preceded by a corresponding authentication event recorded on the domain controller and a logon event recorded on the web server. Although the attacker's system activities should be recorded with the logon session ID `0x2476c741`, Windows records system activities caused by internal web requests from *all* users with System account's logon session ID `0x3e7`. It is worth noting that our system HADES's objective is to perform tracing of system activities attributed to / caused by the same identity, no matter under which user account these system activities were executed. Therefore, it is important to reassign the logon session ID from `0x3e7` to `0x2476c741`, and separate the associated system activities from unrelated system activities caused by other users.

5) Logon Session Linking: Although most system activities under predefined logon ID are forensics-irrelevant, some are directly responsible for the creation of users' logon sessions. For instance, when a user from one machine executes remote commands on another machine via Powershell-Remoting, the remote commands are executed by the process `wsmprovhost.exe` on the second machine, which is spawned by an instance of `svchost.exe` process under the System logon session, i.e., `0x3e7`, once the user successfully authenticated against a domain controller. HADES links users' logon sessions with those predefined logon sessions, and performs only backward-tracing in those sessions until finding the root process responsible for the creation of the user logon session of each access type.

[7]Note that logon sessions and local console / remote desktop sessions are two different concepts. Only a few logon sessions "live" in local console / remote desktop sessions, which could be seen as a container.

Logon session linking is also needed when a privileged user logs in a remote machine via RDP. Due to UAC [51], two logon sessions are created inside a remote desktop session, of which one has a privileged access token and the other does not. System activities run under both logon sessions can only result from the same true identity. Hence these two sessions need to be linked for accurate tracing. Note that there can be no more than two RDP-based logon sessions under the same user account at any time, which is a restriction enforced on Windows OS. There may be more logon sessions under the same user account of other remote access types at the same time. Nevertheless, it is straightforward for HADES to differentiate between these logon sessions through its remote access type inference module introduced above, and hence correctly link the sessions.

It is worth noting that a logon session cannot be nested inside another logon session. That is, a logon session on one machine cannot originate from another logon session on the same machine, but from *exactly one* logon session on another machine. Race condition is not an issue for logon session-based execution partitioning and tracing. Take Fig. 6(c) as an example; even when two distinct logon sessions under the same user account from Machine A have each initiated another logon session inside Machine B at the same time, our system HADES can still correctly perform cross-machine system activity tracing and produce graphs as shown in Fig. 7(c). This is because that each newly created logon session is assigned with a logon session ID, e.g., logon session `z` inside Machine B, with which system activities performed inside this logon session are recorded, and a logon GUID, which is inserted into the corresponding logon event on Machine B by default by Windows specifically for associating this logon event with the authentication request originated from another machine [38]. That is, there is a logon event generated on Machine B for recording the creation of logon session `z`; this event contains a logon GUID as well. The same logon GUID is present in an authentication event recorded on the domain controller; this authentication event holds the information that leads to logon session `x` inside Machine A.

If a logon session is terminated, no more system activities will be created and recorded under the corresponding logon session ID; this logon session will be considered as a leaf logon session. That is, no further logon session on another machine can originate from this logon session; the logon session-based tracing will not advance from this logon session. If a logon attempt from a machine to another machine fails, it will not lead to any system activities on the second machine; no cross-machine system activity tracing is needed for this. For instance, in Fig. 6(c), if the attacker on Machine B tried to access Machine C, but failed due to lack of permission, only a failed logon event [52] will be generated on Machine C; the logon session-based tracing will not advance further from the logon session `w` as shown in Fig. 7(c). HADES's logon session-based tracing ends, when all relevant logon sessions are terminated or no more logon event can be observed.

C. Threat Score Assignment

With our summarization of AD attacks in Fig. 1, we identified two key insights for accurate AD attack detection. The first insight is that attacks against AD follow a rigid pattern: AD discovery, credential access, lateral movement and privilege escalation. The second insight is that the key enabler of an AD attack is successful credential access, and the most observable result of an AD attack is lateral movement. For instance, in Kerberoasting attacks, a service must be first identified, whose credentials will be then stolen, and used for lateral movement and privilege escalation afterwards.

Based on these two critical insights, HADES calculates a threat score for each provenance attack graph generated after logon session-based backward & forward tracing. This is to ensure that the most likely true attacks are always investigated first by security analysts. For each potential lateral movement, i.e., a cross-machine edge in an attack graph, HADES checks how many credential access techniques were executed in the corresponding hosts before, and how often each technique was performed (Algorithm 1 Line 36). For each performed credential access technique, HADES explicitly checks whether the corresponding process has accessed the system process `lsass.exe`[8], which is the most critical process for managing credentials like password hashes and access tokens on Windows OS, and is hence often abused by attackers. Credential access techniques involving accessing `lsass.exe`'s memory should be considered more severe. Then, HADES examines how many AD discovery techniques were operated on related hosts and the frequency of each discovery step conducted (Algorithm 1 Line 35). Also, HADES inspects how many times privilege escalation is observed in the provenance attack graph (Algorithm 1 Line 37).

We formulate the threat score calculation in the following two equations.

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

where n denotes the number of tactics involved, e.g., credential access. $Freq(teq_i)$ 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[9]. Considering both the intensiveness $Freq(teq_i)$ and the extensiveness $Var(tac_i)$ of a tactic being performed aligns with the fact broadly observed in threat reports [54] that attackers often apply multiple techniques within the same tactic to boost their chances of success. Besides, w_i is a weighting factor introduced for each tactic. The weights for discovery, privilege escalation, credential access, and credential access involving `lsass.exe` are currently set to 1.1, 1.2, 1.3, and 1.4, respectively.

The rationales for having a different weight for each tactic are explained in the following. Credential access is considered as the most relevant tactic, and should receive a higher weight than discovery and privilege escalation, as per our second insight. As mentioned above, credential access involving `lsass.exe` is deemed more severe, and hence is assigned with an even higher weight than credential access without involving `lsass.exe`. As the consequence of privilege escalation is more severe than the consequence of discovery, privilege escalation receives a higher weight than discovery as well. The differences between these weights are kept small, as having a much larger value for one tactic than others would mean that the effect of other tactics on threat score calculation would be negligible; note that each constant is the exponent of a power function.

Whereas (1) calculates the threat score for each cross-machine edge (Algorithm 1 Line 38), (2) accumulates the threat scores returned from (1), and outputs the threat score for the attack graph (Algorithm 1 Line 41).

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

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 (1). DA indicates whether credentials of domain administrators are involved in the cross-machine system activities, and takes the value 0 or 1. By doing so, we prioritize attack graphs for further investigation, in which credentials of domain administrators are involved due to more severe consequence of these attacks. Similarly, we assign a $Criticality$ to different attack types. Because, for instance, a Golden-Ticket attack may imply a compromise of the entire domain, whereas a Silver-Ticket attack's scope is limited to certain servers in the domain. If credentials of a non-privileged domain user are involved in a Pass-the-Hash attack, the consequence is even less critical. Currently, the Golden-Ticket attack type, the Silver-Ticket attack type, and the Keberoasting attack type are assigned with a $Criticality$[10] of value 4, 3, and 2, respectively, while all other attack types are assigned with a $Criticality$ of value 1. Note that Keberoasting targets service credentials, whereas other attack types (except Golden-Ticket and Silver-Ticket) target user credentials, which are typically less critical than service credentials.

V. IMPLEMENTATION

We implemented a prototype of HADES in Python, and deployed it on a 64-bit Ubuntu 22.04 OS with 256 GB of RAM and a 32-core processor. We developed HADES with considerations of its deployability in real world in mind, and aim to avoid possible deployment-related restrictions in practice. That is, HADES does not require any specific, non-standard hardware or network configuration; HADES can be easily integrated into an existing security infrastructure.

[8]Access to the system process `lsass.exe` from various processes is frequently observed on a Windows machine also during a normal operation. That is, this action *alone* does not necessarily indicate a malicious activity. However, this action can be suspicious, and is hence often logged. In fact, Sysmon [53] employed by HADES can record access to `lsass.exe`.

[9]MITRE defines tactics as attacker's strategic goals when performing an action, e.g., "credential access", and techniques as specific actions an attacker uses to achieve a tactic, e.g., "OS credential dumping". A tactic includes many techniques. For instance, to achieve "credential access", an attacker may employ "OS credential dumping", "input capture" or other techniques.

[10]The values of $Criticality$ are subjectively assigned, not objectively, as we do not think there is an answer (agreed by security researchers and practitioners) to the question how much more critical a particular attack type is than others. We believe that, in practice, these values should be decided/tuned by the security team of an organization according to their specific preferences, settings and past experiences etc., similar to risk assessment in practice.

First, HADES leverages system logs and authentication & logon logs collected by popular industry-standard instrumentation-free logging frameworks. In Active Directory, authentication logs are recorded on domain controllers by default, whereas logon logs are collected in the accessed hosts by default. We collect authentication logs from the domain controller and logon logs from each domain-joined host. Authentication logs and logon logs can be linked with a logon GUID. For system activities, we collect Windows Security [43] logs and Sysmon [53] logs; both are industry-standard logs. To ensure log integrity, recent Sysmon versions start as protected processes, preventing tampering or disabling by attackers. Windows introduced *Protected Event Logging* [55] to secure logs from unauthorized access.

Second, HADES'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, HADES's detection rules are based on the MITRE ATT&CK Matrix [12], 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. Last, HADES interfaces with Elasticsearch [56] via EQL [57], a query language designed for security purposes; Elasticsearch is a widely utilized tool for event searching and threat analysis within organizations.

HADES has been tested in an emulated environment with a typical organization network as specified in the MITRE Emulation Plans [58], but not yet deployed in a production system. There is a concern that the detailed logs required by HADES would violate users' privacy, especially because, by combining recorded system activities and logon activities, HADES 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 HADES. Deploying HADES as a real-time detection system in a production system requires the capability of real-time log ingestion, which can be easily realized, as HADES interfaces with Elasticsearch and relies on Winlogbeat [59] to ship logs to Elasticsearch. Winlogbeat is provided by Elastic, and designed to ship logs off of endpoints as soon as possible. Elastic also provides guidelines for data storage management in Elasticsearch in a production environment [60], [61], [62].

VI. EVALUATION

Public Datasets: Public datasets often do not contain AD attacks presumably due to higher requirement on setting up emulation infrastructures. For instance, the DARPA E3 dataset [63] does not contain any AD-based attack. Although in the DARPA E5 dataset [64], a so-called Copykatz module is used for credential access, the credentials obtained are local credentials, as the corresponding machines are not joined to a domain. The DARPA OpTC dataset [65] is the only public dataset including AD-based attacks that we can find. Unfortunately, this dataset only includes system logs on domain-joined hosts, but no logs from the domain controller. Authentication logs, which are recorded only in domain controllers, are critical for HADES's functionality. Hence we cannot evaluate HADES on the OpTC dataset either.

TABLE II
OVERVIEW OF THE EVALUATION DATASETS

Dataset	Target Host Number	Ground Truth Attack	Target Host OS	Data Size	Event Number
APT29	3	Golden-Ticket	Windows	253GB	160M
WizardSpider	3	Kerberoasting	Windows	151GB	87M
Oilrig	4	Pass-the-Hash	Windows	200GB	116M

MITRE Attack Emulation: AD attacks are present in almost all MITRE Emulation Plans [58]. MITRE Engenuity ATT&CK Evaluations [66] use these plans to assess commercial detection systems from leading security vendors. MITRE's emulation plans target enterprise networks with more sophisticated, cross-machine attacks than most public datasets, offering greater authenticity. However, MITRE has not published any corresponding datasets. We stringently implemented three emulation plans, i.e, APT29 [67], Oilrig [35], and WizardSpider [68], and then evaluated HADES on these datasets. Table II gives an overview of our datasets. Note that these three datasets are now combined into a published dataset called AVIATOR [69], [70]. A detailed comparison between the AVIATOR dataset and other APT datasets is provided in [69].

A. HADES vs. SIEM Detection Rules

We extracted all detection rules for AD-based attacks from the most popular and established SIEM rule repositories, i.e., Elastic [29], Sigma [30], Google Chronicle [71], and converted them into EQL queries, which we then run parallel to HADES on our datasets. We compare the detection results of Elastic, Chronicle, Sigma and our system HADES in Table III. To ensure fair comparison, in Table III, we only show the detection results of these SIEM rules for attack techniques that HADES's stage 1 can detect (Fig. 5), and not other AD attack techniques like AD discovery, which would lead to a much higher number of false positives.

Note that we implement HADES as an on-demand PIDS for enhanced efficiency. Only when HADES's stage 1 detects a potential lateral movement, it runs its stage 2 for a whole-network investigation, which unfolds other tactics like AD discovery and credential access when they are causally related to the lateral movement. That is, HADES cannot detect an early-stage AD attack, in which the attacker has only performed AD discovery and credential access, but not yet moved to other machines. In particular, for attacks like Golden-Ticket, HADES's stage 1 cannot detect the creation of a Golden-Ticket, i.e., credential access, but its usage for lateral movement. HADES's stage 2 checks credential access techniques related to Golden-Ticket, if the stage 1 detects a Golden-Ticket-based lateral movement. HADES avoids a high rate of false alarms by neglecting potential, yet less critical early-stage attack steps.

Table III reveals that, comparing to HADES, SIEM rules produced more false positives and missed some true attacks. Whereas Chronicle has a relatively low false positive rate in all

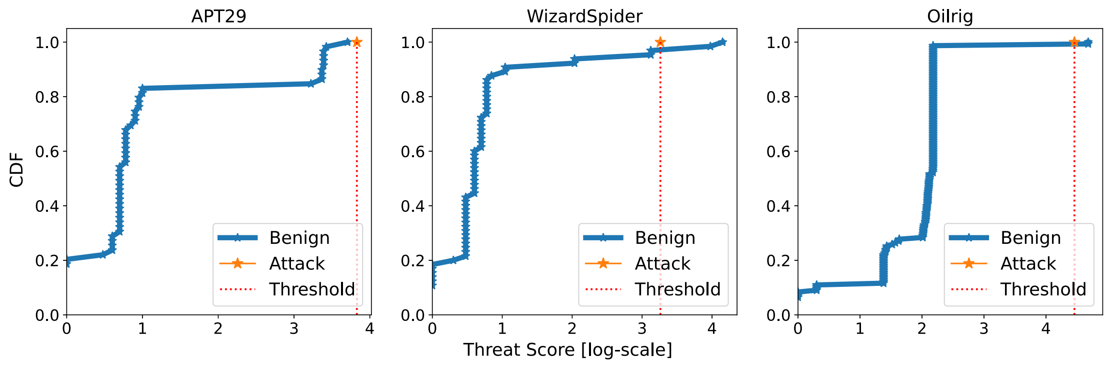


Fig. 8. CDF of threat score for false and true alerts.

TABLE III
COMPARISON OF HADES AND PRIOR DETECTION SYSTEMS

Datasets	HADES				Elastic		Chronicle		Sigma		CAD	
	Stage 1		Stage 2									
	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN
APT29	60	0	0	0	148	1	14	1	545	0	0	0
WizardSpider	66	0	2	0	194	1	30	1	605	0	0	1
Oilrig	156	0	2	0	213	0	37	1	417	0	0	1

[1] Note that each dataset has only one true positive (TP=1).

datasets, it missed the true attack in each dataset. In contrast, Sigma detected all true attacks, at the cost of having a much higher false positive rate. Elastic detected the Pass-the-Hash attack in the Oilrig dataset with less false positives, but missed the attacks in APT29 dataset and WizardSpider dataset.

To understand why SIEM rules have a high rate of false positives and false negatives, we resort to manually inspect them. On the one hand, we find that a high number of false positives often results from overly general rules matching system activities caused by benign programs often misused by attackers, i.e., LOLBins. Further, Sigma's unusually high number of false positives is also due to the fact that Sigma's rule repository includes many similar rules created by different contributors for the same attack techniques. On the other hand, both Chronicle's and Elastic's rules tend to be overly specific, i.e., having hard-coded strings as conditions, and also often allowlist system activities caused by known "benign" programs. Simply favoring high-degree of specificity and overly allowlisting lead to less false positives, but inevitably cause more false negatives.

Unlike open-source SIEM rules, HADES employs an advanced detection strategy that goes beyond solely looking for specific isolated log events, contributing to an average false positive reduction rate of 98%. Most activities involved in AD attacks are based on (mis)using legitimate AD infrastructures and services. For example, network shares are used intensively by normal domain users and only occasionally by attackers for lateral movement. That is, attackers can easily blend into the noise caused by legitimate users, as indicative system and network activities related to AD attacks are so prevalent in benign operations. However, when putting truly causally related system activities in a provenance graph via precise cross-machine tracing, it exposes provenance graphs containing system activities conducted by attackers as they typically follow a rigid AD attack pattern unlike benign graphs.

Detailed comparison between HADES's stages 1 and 2 is given in Fig. 8, which presents the cumulative distribution function of threat scores for benign and attack alerts. HADES's stage 2 results in Table III are based on the true attack threat score threshold. Our threat triage algorithm in HADES's stage 2, combined with logon session-based execution partitioning and tracing, contributes to an average false positive reduction rate of 99% when comparing to its stage 1.

B. *HADES vs. CAD*

Prior research [72] shows that, contrary to general perception, commercial security products may have poor detection quality despite high price tag. In order to answer the question how much value a commercial attack detection product could bring, we subscribed to a cloud-based dedicated AD attack detection solution from a popular security vendor and evaluated it alongside HADES on our datasets. The security vendor is considered as a significant security solution provider in the security market overviews of Gartner [73] and Forrester [74]. However we did not get the permission to name the vendor in the present paper, and hence refer to the security product we purchased simply as CAD (commercial attack detector).

We installed CAD on the domain controller of our emulation infrastructure. We were also provided with a documentation of the security product we purchased, in which a guideline

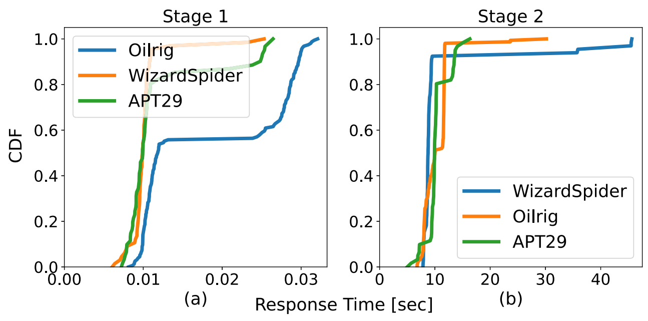


Fig. 9. CDF of response time of HADES.

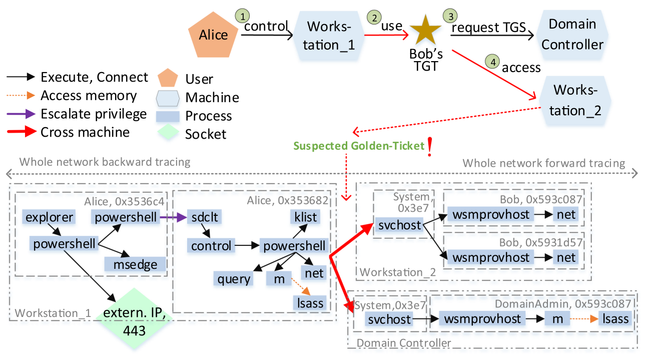


Fig. 10. An attack graph created by HADES on the APT29 dataset.

for CAD's logging configuration and its high-level detection logic/rules are included. To our surprise, as shown in Table III, the commercial product did not raise a single false alarm, but is plagued with a high false negative rate. The result of our investigation, however, aligns with the findings in prior work [72], i.e., commercial security products may sacrifice detection rate in exchange for analyst time.

By consulting on the documentation, in particular CAD's high-level detection logic, we were able to find some explanation for the poor detection rate. First, we find that CAD, as a dedicated AD attack detection system, does not collect enough data we consider as necessary for accurate detection. That is, CAD only collects a limited set of authentication & logon event types, leading to restricted visibility. Second, some of its detection rules do not fire alerts on events that are previously observed on the same computers. With the prevalence of LOLBins, doing so reduces false positives, but inevitably introduces false negatives. Third, CAD does not check system logs inside each host at all. We argue that credential access techniques, a critical step in AD attacks, are detectable mostly via system logs.

To avoid false conclusion, we engaged with the vendor over weeks by first checking if our installation and configuration of CAD was correct, and then presenting our finding, and asking for explanation on the high false negative rate. The answer of the vendor confirmed our evaluation findings. In particular, the vendor responded to the false negative for the Pass-the-Hash attack by saying that CAD's detection on Pass-the-Hash attacks has known issues and they are working on a fix.

C. Response Time

We measure the response time of HADES in two separate parts. Fig. 9 shows the cumulative distribution function of response time of HADES in its two stages, respectively. HADES's stage 1 response time is measured per authentication & logon alert. Fig. 9(a) reveals that it takes less than 35 milliseconds to find an authentication & logon anomaly for all authentication & logon events in each dataset. We measure HADES's stage 2 response time as the time to perform logon session-based backward & forward tracing on a high-level alert found in the stage 1, while checking for system activities related to AD discovery techniques, credential access, and privilege escalation, and calculate the threat score for the resulted attack graph. As shown in Fig. 9 (b), it takes at most 45 seconds to return a low-level provenance attack graph with associated threat score for each stage 1's high-level alert in every dataset.

D. Case Study

1) Pass-the-Hash: Pass-the-Hash attacks leverage NTLM authentication process, instead of the default AD authentication process Kerberos. This authentication anomaly manifests in our authentication anomaly detection model introduced in Section IV. However, using this model alone introduces many false alerts, as NTLM authentication process is still widely in use in Windows domains. For instance, when a user accesses a remote network share via NTLM, the authentication process against the target server and the domain controller looks the same as in Pass-the-Hash attacks. That is, without resorting to system logs, it is difficult to reliably detect Pass-the-Hash attacks. This is also why CAD missed this attack in the Oilrig dataset. However, we show that, by using whole network provenance tracing and an alert triage algorithm, HADES can reliably detect Pass-the-Hash, and drastically reduce false positives caused by benign user activities. Fig. 3 automatically created by HADES presents the entire attack chain conducted by the attacker.

2) Golden-Ticket: In a Golden-Ticket attack, the attacker manages to steal the credentials of the `krbtgt` account from a domain controller, and is then able to create a TGT impersonating any domain user. The attacker does not request a TGT from a domain controller before requesting a TGS to an application server, showing an anomaly in our authentication anomaly detection model. However, during a benign operation, when a cached TGT is used to request access to a server, it exhibits the same network authentication anomaly, making Golden-Ticket difficult to detect without inspecting system logs on each involved machines. For the Golden-Ticket attack in the APT29 dataset, as illustrated in Fig. 10, HADES first creates an initial high-level attack graph involving AD entities like users and hosts, after it detects an authentication & logon anomaly and suspects a Golden-Ticket attack. Then it performs system-level forward tracing inside the specific logon session of involved username Bob in the accessed host Workstation_2, and system-level forward & backward tracing inside the logon session of Alice in the accessing host Workstation_1, leading to a logon session in the Domain Controller. This attack graph discloses that the attacker first performed AD discovery and credential

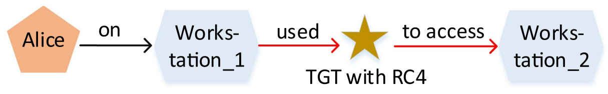


Fig. 11. An attack graph created by CAD on the APT29 dataset.

access on the initially compromised machine Workstation_1, then moved to the Domain Controller for further credential access, and finally created a Golden-Ticket for accessing the Workstation_2.

In comparison, attack graphs produced by the commercial detector CAD are incomplete and much less informative, hindering swift attack investigation by security analysts. For the attack scenario in the WizardSpider dataset and the attack scenario in the Oilrig dataset, CAD did not output an attack graph, as it has missed the attacks. Fig. 11 shows the attack graph produced by CAD for the attack scenario in the APT29 dataset. Comparing to the attack graph produced by our system HADES in Fig. 10, 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.

E. Threats to the Validity of the Experimental Evaluation

HADES is evaluated on datasets resulted from implementation of MITRE emulation plans. We stringently followed the emulation plans including the infrastructure setups, and are unaware of any uncontrolled variables within our experiments that could introduce threats to the internal validity. HADES's performance is measured not only by its false positive rate & false negative rate, but also by its ability to construct concise and complete attack graphs, which we consider as the most important performance metrics. We developed HADES with considerations of its deployability in real world in mind. HADES is evaluated on a typical organization network as specified in the MITRE emulation plans. HADES'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. We believe that HADES can achieve similar detection performance in various environments. However, there might be unknown factors influencing HADES's performance in real world. Due to privacy concern, HADES is not yet extensively tested in real world, as further discussed in Section VIII-C.

VII. DISCUSSION & RELATED WORK

A. Provenance-Based IDS

Prior PIDS [14], [15], [16], [17], [18], [19], [20], [21], [22], [23] are restricted to intra-machine tracing, and unaware of the network context, hindering their adaptability for AD attack detection. That is, by construction, these systems would fail to detect AD attacks. TRACE [27] is an early attempt to detect cross-machine attacks in enterprise networks by connecting intra-machine provenance graphs whenever there is a network connection between two machines, oblivious to the domain context. This naive approach inevitably leads to cross-machine dependency explosion, as discussed in Section IV. In contrast, HADES is designed with realistic multi-users computing environment in domain context in mind, and can narrow the cross-machine tracing down to a specific logon session containing system activities conducted truly by the same identity, drastically reducing false dependencies.

However, HADES's cross-machine tracing is limited to AD environment and domain-joined machines. Devices brought by employees themselves, aka BYOD, are out of HADES's scope, since for these devices the authentication & logon process is done locally with local credentials instead of domain credentials. Yet, local credentials cannot be used in identity-based attacks for moving laterally to other machines in a network. That is, BYOD cannot be used for identity-based attacks. Although attackers may move laterally from BYOD to domain-joined devices via binary exploitation, these kind of attacks are much more expensive and much less popular than identity-based attacks in reality, as recent CrowdStrike studies [1], [2], [3], [4] show that 80% of modern cyberattacks are identity-driven. Binary exploitation does not cause any authentication event. Nor does it cause any logon event. It may, however, cause some system activity events on an accessed device. Without authentication and logon events, HADES cannot perform accurate causality-based cross machine tracing. The only way to also address binary exploitation-based cross-machine attacks would be to conduct network connection-based cross-machine tracing, which is highly inaccurate, as discussed in Section IV-B.

B. Dependency Explosion

Intra-machine 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. Similarly, cross-machine dependency explosion occurs if edges are built simply on a (coarse-granular) network connection basis. Program execution partitioning techniques [13], [25], [26] are the first proposed to combat intra-machine dependency explosion. MPI [75] improves these techniques by introducing a semantics aware program annotation and instrumentation technique. Whereas all these techniques focus on execution of individual long-running processes, logon session-based execution partitioning operates at a higher level, beneficial for alleviating both intra-machine and cross-machine dependency explosion. Nonetheless, the existing techniques are complementary to HADES and can further reduce false dependencies in provenance graphs.

C. Lateral Movement Detection

Although HADES is more than a lateral movement detector, its design is centered on lateral movement detection. That is, HADES's stage 1 alerts on potential lateral movements, which are then further investigated in its stage 2 via whole-network provenance tracing, and triaged with the insights from an extensive analysis of AD attacks conducted by APT actors, accurately and comprehensively unraveling attackers' traversal inside enterprise networks. HADES overcomes several inherent weaknesses, in terms of logging and assumptions, of state-of-the-art lateral movement detection system Hopper [76]. First, Hopper proposes an inference algorithm to identify complete login paths in enterprise networks, as leveraging standard authentication logs alone is insufficient to identify those paths. In contrast, HADES combines authentication & logon logs and system logs to trace these paths on a logon session basis, producing login paths guaranteed to be correct, unlike Hopper's inference algorithm. Second, Hopper's detection algorithm operates on two assumptions: 1) attackers use a different set of credentials when moving to other machines, 2) attackers eventually access a machine which they previously could not access. While intuitive, these assumptions are not reliable enough, as HADES's stage 1 shows that detection based on authentication & logon anomaly alone leads to many false positives. A thorough investigation via detailed system activities checking is critical for reducing these false alarms.

D. Active Directory Attack Detection

Like the lateral movement detector Hopper [76], existing Active Directory attack detectors [77], [78] are only comparable to our system HADES's stage one. That is, these systems are anomaly detectors that can only output an (unranked) alert simply suggesting a possible attack, but no attack graph assisting in understanding the detection decision and hence further investigation. Anomaly detection alone often leads to many false positives. The lack of an explanation makes it less useful for security analysts in practice.

In comparison, HADES is not just a classification system. Rather, HADES facilitates the investigation process in unearthing the root causes and attack ramifications, by providing contextualized and more interpretable detection results. HADES can perform accurate cross-machine system activity tracing in its second stage, thanks to our novel concept called logon session-based execution partitioning, to automatically associate all system activities of each user or attacker. By mapping the system activities inside each graph to AD-related attack techniques when possible and then counting the number of attack techniques present in each graph, HADES ranks these graphs and therefore outputs ranked alerts, with the most likely AD attacks being ranked the highest. This is to help security analysts perform swift investigation, as they often find it difficult and time consuming to investigate on, associate and understand the detection results of currently deployed security systems [79], [80], [81]. We believe that this functionality makes HADES much more practical than existing systems. Note that a direct, empirical comparison with these existing systems [27], [76], [77], [78] is difficult given the absence of source code and differences in evaluation datasets.

VIII. LIMITATIONS

Currently, our system HADES has a few known limitations that need to be addressed in future work.

A. Evading HADES

In practice, a PIDS's ability to reconstruct concise and accurate 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 pinpointing 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. Knowing this, advanced attackers may employ attack techniques to specifically hinder HADES's accurate system activity tracing and prevent it from reconstructing correct and complete attack graphs.

For instance, attackers may perform logon session hijacking to disrupt HADES's cross-machine tracing. With sufficient privilege on one session, attackers can hijack another existing session. However, a session hijacking attack is not directly evident in system logs; 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. This would inevitably lead to premature end of the cross-machine system activity tracing, and result in incomplete or even incorrect attack graphs. Therefore, future work needs to make HADES more robust by discovering all possible evasion techniques and introducing countermeasures to safeguard HADES's cross-machine tracing.

B. Reliance on Log Integrity

A further limitation of HADES is its reliance on logs & log integrity. Like *all* detection systems that rely on logs, i.e., every system listed in Section VII, our system HADES would likely fail at detecting the attacks, if attackers have managed to manipulate logs without leaving any trace, or there are no logs generated or left at all. This would be a very effective evasion technique. However, as mentioned in Section V, there are already efforts being put into preventing log manipulation. For instance, Windows introduced *Protected Event Logging* [55] to secure logs from unauthorized access. Nevertheless, we believe that more log protection techniques need to be designed in future work in case attackers can bypass existing ones.

C. Deploying HADES in Real World

We developed HADES with considerations of its deployability in real world in mind; HADES has been tested in an emulated environment based on a typical organization network as specified in the MITRE Emulation Plans [58]. HADES leverages system logs and authentication & logon logs collected by popular industry-standard instrumentation-free logging tools provided by Microsoft. These logging tools are installed on each host and have low computational overhead. Collected logs are shipped to Elasticsearch and stored in a more efficient format. HADES interfaces with Elasticsearch, a widely utilized tool for scalable and near real-time event searching and threat analysis within organizations. HADES's response time, i.e., time to output an AD attack graph assigned with a threat score, does not directly depend on the size of an enterprise environment, but rather on the number of cross-machine hops in each suspected attack instance, i.e., the number of involved hosts. Note that HADES performs fine-grained provenance tracing only on involved hosts. That is, HADES's performance degradation in terms of response time in a large enterprise environment would be insignificant. Further, the size of an enterprise would have no direct impact on HADES's performance on generating accurate attack graphs and its detection accuracy.

HADES has not yet been deployed in a production system. This is due to a concern that the detailed logs required by HADES would violate users' privacy. By combining recorded system activities and logon activities, HADES can reveal which employee has carried out what activities on which machines at what time. That is, deploying HADES without integrating a privacy-preserving logging solution in an organization might lead to increased vulnerability to abuse of power, decreased trust, or even legal consequences. A proper privacy-preserving logging mechanism needs to be designed in future work and incorporated into HADES; only then is HADES readily deployable in real world. HADES is potentially compatible with popular privacy-preserving techniques like log anonymization, data masking, noise injection. That is, personally identifiable information from logs like user ID can be replaced with hashes or pseudonyms that are unique but non-reversible. Besides, information like IP addresses inside network activity logs that may reveal user identity but is irrelevant to HADES can be masked with a fixed value. Further, random forensics-irrelevant user activities can be injected into the logs to make any attempt to reverse-engineer personal information more difficult.

IX. CONCLUSION

We present a novel accurate AD attack detection and investigation system named HADES in this paper. Based on a thorough study of AD attacks launched by APT actors, we create a succinct and contextualized AD attack overview revealing key steps and prerequisites of various AD attack types. We leverage critical insights from our extensive analysis on AD attacks to design a light-weight authentication anomaly detection model, and a threat triage algorithm. We propose the concept logon session-based execution partitioning and tracing, which significantly reduces false dependencies, contributing to accurate and swift AD attack detection. Our evaluations, conducted on datasets derived from rigorously implemented MITRE emulation plans, demonstrate that HADES significantly outperforms open-source SIEM detection rules and a commercial dedicated AD attack detector.

REFERENCES

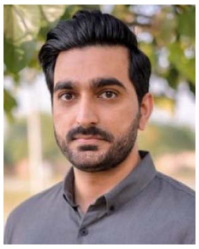
[1] CrowdStrike, Inc., "CrowdStrike 2023 global threat report," 2023. [Online]. Available: https://www.crowdstrike.com/global-threat-report/
[2] CrowdStrike, Inc., "CrowdStrike 2023 threat hunting report," 2023. [Online]. Available: https://www.crowdstrike.com/resources/reports/threat-hunting-report/
[3] V. Shastri, "Attackers set sights on active directory: Understanding your identity exposure," Accessed: Dec. 2023. [Online]. Available: https://www.crowdstrike.com/blog/attackers-set-sights-on-active-directory-understanding-your-identity-exposure/
[4] V. Shastri, "Endpoint and identity security: A critical combination to stop modern attacks," Accessed: Dec. 2023. [Online]. Available: https://www.crowdstrike.com/blog/unifying-endpoint-and-identity-security/
[5] The MITRE Corporation, "MITRE T1558.003," Accessed: Dec. 2023. [Online]. Available: https://attack.mitre.org/techniques/T1558/003/
[6] The MITRE Corporation, "MITRE T1550.002," Accessed: Dec. 2023. [Online]. Available: https://attack.mitre.org/techniques/T1550/002/
[7] S. Krishnamoorthi and J. Carleton, "Active directory holds the keys to your kingdom, but is it secure?" 2020. [Online]. Available: https://www.frost.com/frost-perspectives/active-directory-holds-the-keys-to-your-kingdom-but-is-it-secure/
[8] "Nmap," Accessed: May 2024. [Online]. Available: https://nmap.org/
[9] Microsoft, "Setspn," Accessed: Jan. 2024. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc731241(v=ws.11)
[10] Falcon OverWatch Team, "8 LOLBins every threat hunter should know," Accessed: May 2023. [Online]. Available: https://www.crowdstrike.com/blog/8-lolbins-every-threat-hunter-should-know/
[11] Trellix, "Trellix threat report 2023," 2023. [Online]. Available: https://www.trellix.com/advanced-research-center/threat-reports/feb-2023/
[12] The MITRE Corporation, "MITRE matrix," Accessed: Jan. 2023. [Online]. Available: https://attack.mitre.org/matrices/enterprise/
[13] S. Ma, X. Zhang, and D. Xu, "ProTracer: Towards practical provenance tracing by alternating between logging and tainting," in *Proc. Int. Conf. Netw. Distrib. Syst. Secur.*, 2016, pp. 1–15.
[14] M. N. Hossain et al., "SLEUTH: Real-time attack scenario reconstruction from COTS audit data," in *Proc. USENIX Secur. Symp.*, 2017, pp. 487–504.
[15] W. U. Hassan, L. Mark, N. Aguse, A. Bates, and T. Moyer, "Towards scalable cluster auditing through grammatical inference over provenance graphs," in *Proc. Netw. Distrib. Syst. Secur.*, 2018, pp. 1–15.
[16] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: Real-time APT detection through correlation of suspicious information flows," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 1137–1152.
[17] W. U. Hassan et al., "NoDoze: Combatting threat alert fatigue with automated provenance triage," in *Proc. Netw. Distrib. Syst. Secur.*, 2019, pp. 1–15.
[18] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 1172–1189.
[19] M. N. Hossain, S. Sheikhi, and R. Sekar, "Combating dependence explosion in forensic analysis using alternative tag propagation semantics," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 1139–1155.
[20] Z. Cheng et al., "KAIROS: Practical intrusion detection and investigation using whole-system provenance," in *Proc. IEEE Symp. Secur. Privacy*, 2024, pp. 9–28.
[21] M. Rehman, H. Ahmadi, and W. Hassan, "FLASH: A comprehensive approach to intrusion detection via provenance graph representation learning," in *Proc. IEEE Symp. Secur. Privacy*, 2024, pp. 142–161.
[22] J. Zeng et al., "SHADEWATCHER: Recommendation-guided cyber threat analysis using system audit records," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, 2022, pp. 489–506.
[23] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "PROGRAPHER: An anomaly detection system based on provenance graph embedding," in *Proc. USENIX Secur. Symp.*, 2023, pp. 4355–4372.
[24] F. Dong et al., "Are we there yet? An industrial viewpoint on provenance-based endpoint detection and response tools," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2023, pp. 2396–2410.
[25] K. H. Lee, X. Zhang, and D. Xu, "High accuracy attack provenance via binary-based execution partition," in *Proc. Netw. Distrib. Syst. Secur.*, 2013, pp. 1–16.
[26] S. Ma, K. H. Lee, C. H. Kim, J. Rhee, X. Zhang, and D. Xu, "Accurate, low cost and instrumentation-free security audit logging for windows," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2015, pp. 401–410.
[27] H. Irshad et al., "TRACE: Enterprise-wide provenance tracking for real-time APT detection," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4363–4376, 2021.
[28] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of SOC analysts' perspectives on security alarms," in *Proc. USENIX Secur. Symp.*, 2022, pp. 2783–2800.

[29] Elastic, "Elastic detection rules," Accessed: Sep. 2023. [Online]. Available: https://github.com/elastic/detection-rules
[30] SigmaHQ, "Sigma," Accessed: Sep. 2023. [Online]. Available: https://github.com/SigmaHQ/sigma
[31] M. Crockett, "Why 86% of organizations are increasing their investment in active directory security," Accessed: Dec. 2023. [Online]. Available: https://securityboulevard.com/2021/11/why-86-of-organizations-are-increasing-their-investment-in-active-directory-security/
[32] A. Talyanski, "nopac exploit: Latest microsoft AD flaw may lead to total domain compromise in seconds," Accessed: Jun. 2024. [Online]. Available: https://www.crowdstrike.com/blog/nopac-exploit-latest-microsoft-ad-flaw-may-lead-to-total-domain-compromise/
[33] Inc Tenable, "A gloabl threat to enterprises: The impact of active directory attacks," Accessed: Jun. 2024. [Online]. Available: https://de.tenable.com/whitepapers/a-global-threat-to-enterprises-the-impact-of-ad-attacks?page=2
[34] X. Han, T. Pasqueir, A. Bates, J. Mickens, and M. Seltzer, "UNICORN: Runtime provenance-based detector for advanced persistent threats," in *Proc. Netw. Distrib. Syst. Secur.*, 2020, pp. 1–18.
[35] The MITRE Corporation, "Oilrig emulation plan," Accessed: Oct. 2023. [Online]. Available: https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/oilrig
[36] The MITRE Corporation, "Golden ticket," Accessed: Jan. 2024. [Online]. Available: https://attack.mitre.org/techniques/T1558/001/
[37] V. Pamnani, "4769(s, f): A kerberos service ticket was requested," Accessed: Jun. 2024. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4769
[38] V. Pamnani, "4624(s): An account was successfully logged on," Accessed: Jun. 2024. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4624
[39] V. Pamnani, "4768(s, f): A kerberos authentication ticket (TGT) was requested," Accessed: Jun. 2024. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4768
[40] The MITRE Corporation, "MITRE ATT&CK," Accessed: Jan. 2023. [Online]. Available: https://attack.mitre.org
[41] Mandiant, "Mandiant M-trends 2023," 2023. [Online]. Available: https://www.mandiant.com/resources/reports
[42] M. A. Inam et al., "SoK: History is a vast early warning system: Auditing the provenance of system intrusions," in *Proc. IEEE Symp. Secur. Privacy*, 2023, pp. 2620–2638.
[43] Microsoft, "Security auditing," Accessed: May 2023. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/security-auditing-overview
[44] Microsoft, "LSA logon sessions," Accessed: Feb. 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/secauthn/lsa-logon-sessions
[45] M. E. Russinovich and A. Margosis, *Troubleshooting With the Windows Sysinternals Tools*, 2nd ed. Redmond, WA, USA: Microsoft Press, 2016.
[46] A. Allievi, A. Ionescu, D. A. Solomon, K. Chase, and M. E. Russinovich, *Windows Internals, Part 2*, 7th ed. Redmond, WA, USA: Microsoft Press, 2022.
[47] A. Miroshnikov, *Windows Security Monitoring: Scenarios and Patterns*. Hoboken, NJ, USA: Wiley, 2018.
[48] J. Forshaw, *Windows Security Internals: A Deep Dive Into Windows Authentication, Authorization, and Auditing*. San Francisco, CA, USA: No Starch Press, 2024.
[49] The MITRE Corporation, "T1021.001," Accessed: Jan. 2024. [Online]. Available: https://attack.mitre.org/techniques/T1021/001/
[50] Microsoft, "Fast user switching," Accessed: Feb. 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/shell/fast-user-switching
[51] Microsoft, "User account control," Accessed: Feb. 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/
[52] V. Pamnani, "4625(f): An account failed to log on," Accessed: Jun. 2024. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4625
[53] M. Russinovich and T. Garnier, "System monitor," Accessed: Feb. 2023. [Online]. Available: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon
[54] The MITRE Corporation, "MITRE ATT&CK campaigns," Accessed: May 2024. [Online]. Available: https://attack.mitre.org/campaigns/
[55] S. Wheeler and M. Lombardi, "About logging windows," Accessed: Apr. 2023. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_logging_windows?view=powershell-7.3
[56] Elastic NV, "Elasticsearch," Accessed: Sep. 2023. [Online]. Available: https://www.elastic.co/
[57] Elastic NV, "EQL search," Accessed: Sep. 2023. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/eql.html
[58] The MITRE Corporation, "MITRE adversary emulation library," Accessed: Jan. 2023. [Online]. Available: https://github.com/center-for-threat-informed-defense/adversary_emulation_library
[59] Elastic NV, "Lightweight shipper for windows event logs," Accessed: Jun. 2024. [Online]. Available: https://www.elastic.co/beats/winlogbeat/
[60] M. Davis, "How to design your elasticsearch data storage architecture for scale," Accessed: Apr. 2025. [Online]. Available: https://www.elastic.co/beats/winlogbeat/
[61] Elastic NV, "Resize your deployment," Accessed: Apr. 2025. [Online]. Available: https://www.elastic.co/guide/en/cloud-enterprise/current/ece-resize-deployment.html
[62] Elastic NV, "Get ready for production," Accessed: Apr. 2025. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html
[63] A. D. Keromytis, "DARPA transparent computing E3," Accessed: Sep. 2023. [Online]. Available: https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md
[64] J. Torrey, "DARPA transparent computing," Accessed: Sep. 2023. [Online]. Available: https://github.com/darpa-i2o/Transparent-Computing
[65] M. van Opstal and W. Arbaugh, "DARPA OpTC," Accessed: Sep. 2023. [Online]. Available: https://github.com/FiveDirections/OpTC-data
[66] The MITRE Corporation, "MITRE engenuity," Accessed: Jan. 2023. [Online]. Available: https://attackevals.mitre-engenuity.org/
[67] The MITRE Corporation, "APT29 emulation plan," Accessed: Oct. 2023. [Online]. Available: https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/apt29
[68] The MITRE Corporation, "WizardSpider emulation plan," Accessed: Oct. 2023. [Online]. Available: https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/wizard_spider
[69] Q. Liu, K. Bao, and V. Hagenmeyer, "AVIATOR: A MITRE emulation plan-derived living dataset for advanced persistent threat detection and investigation," in *Proc. 2024 IEEE Int. Conf. Big Data*, 2024, pp. 5592–5601.
[70] Q. Liu, K. Bao, and V. Hagenmeyer, "AVIATOR dataset," Accessed: Nov. 2024. [Online]. Available: https://gitlab.kit.edu/kit/iai/rsa/aviator
[71] Google Security Operations, "Chronicle detection rules," Accessed: Sep. 2023. [Online]. Available: https://github.com/chronicle/detection-rules
[72] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. van Eeten, "A different cup of TI? The added value of commercial threat intelligence," in *Proc. USENIX Secur. Symp.*, 2020, pp. 433–450.
[73] E. Mirolyubov, M. Taggett, F. Hinner, and N. Patel, "Magic quadrant for endpoint protection platforms," 2023. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-2FFCXFOM&ct=231025&st=sb
[74] A. Mellen, "The forrester new wave: Extended detection and response (XDR) providers, Q4 2021," 2021. [Online]. Available: https://www.forrester.com/report/the-forrester-new-wave-tm-extended-detection-and-response-xdr-providers-q4-2021/RES176400
[75] S. Ma, J. Zhai, F. Wang, K. H. Lee, X. Zhang, and D. Xu, "MPI: Multiple perspective attack investigation with semantic aware execution partitioning," in *Proc. USENIX Secur. Symp.*, 2017, pp. 1111–1128.
[76] G. Ho et al., "Hopper: Modeling and detecting lateral movement," in *Proc. USENIX Secur. Symp.*, 2021, pp. 3093–3110.
[77] L. Kotlaba, S. Buchovecká, and R. Lórencz, "Active directory kerberoasting attack: Detection using machine learning techniques," in *Proc. 7th Int. Conf. Inf. Syst. Secur. Privacy*, 2021, pp. 376–383.
[78] W. Matsuda, M. Fujimoto, and T. Mitsunaga, "Detecting APT attacks against active directory using machine leaning," in *Proc. 2018 IEEE Conf. Appl. Inf. Netw. Secur.*, 2018, pp. 60–65.
[79] E. Segal, "Alert fatigue," Accessed: Aug. 2023. [Online]. Available: https://www.forbes.com/sites/edwardsegal/2021/11/08/alert-fatigue-can-lead-to-missed-cyber-threats-and-staff-retentionrecruitment-issues-study/?sh=4c96871035c9
[80] C. Robinson, "In cybersecurity every alert matters," 2021. [Online]. Available: https://www.criticalstart.com/wp-content/uploads/2021/11/US48277521_TLWP.pdf

[81] CRITICALSTART, "The impact of security alert overload," 2019. [Online]. Available: https://www.criticalstart.com/wp-content/uploads/2021/02/CS_Report-The-Impact-of-Security-Alert-Overload.pdf

Qi Liu received the PhD degree from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2025. His research interests include system security, auditing & logging, data provenance analysis, and advanced persistence threat detection and investigation.

Kaibin Bao received the PhD degree from the Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2021. He is currently the head with Working Group Resilient Secure Automation, Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology. His research topics include cybersecurity for critical infrastructures and automation systems using machine learning methods.



Wajih Ul Hassan received the PhD degree in computer science from the University of Illinois Urbana-Champaign, Champaign, IL, USA, in 2021. He is currently an assistant professor with the Department of Computer Science, University of Virginia, Charlottesville, VA, USA. He has collaborated with NEC Labs and Symantec Research Labs to integrate his defensive techniques into commercial security products. His research interests include securing complex networked systems by leveraging data provenance approaches and scalable system design. He was the recipoient of the NSF CAREER Award, Symantec Research Labs Graduate Fellowship, recognition as a Young Researcher at the Heidelberg Laureate Forum, RSA Security Scholarship, Mavis Future Faculty Fellowship, Sohaib and Sara Abbasi Fellowship, and an ACM SIGSOFT Distinguished Paper Award.

Veit Hagenmeyer (Member, IEEE), received the PhD degree from Université Paris XI, Paris, France in 2002. He is currently a professor of energy informatics with the Faculty of Computer Science, and the director with the Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany. His research interests include modeling, optimization and control of energy systems, machine learning-based forecasting in energy systems, and integrated cybersecurity of such systems.