

The Consequences of Benchmark Scarcity in Provenance-Based Intrusion Detection

Simon Klancher^{*}, Joel Flores^{*}, Josh Wiedemeier^{*}, Murat Kantarcioglu[†], Kangkook Jee^{*}

^{*}The University of Texas at Dallas

[†]Virginia Polytechnic Institute and State University

^{*}{srk200018, jjf190004, jdw170000, kangkook.jee}@utdallas.edu,

[†]muratk@vt.edu

Abstract—Provenance-based Intrusion Detection Systems (PIDSs) have achieved exceptional detection performance on publicly available datasets in recent years. Yet outside the research community, their adoption has stalled. A key reason for this stagnation is that PIDSs are largely tailored and optimized to benchmark datasets in which benign activity is predictable and unrealistic, failing to capture real-world user dynamics and behavioral diversity. Moreover, the attack scenarios in these datasets are often short-lived and outdated. For anomaly-based PIDSs, the repetitive, automated benign activity in these benchmarks masks the risk of false positives under more realistic and diverse user behavior. The absence of long-running attacks also allows PIDSs to bound runtime complexity by partitioning large provenance graphs into short temporal slices without severing long-range attack dependencies. To bridge this gap, we collect DOLLHOUSE, a novel large-scale system provenance dataset comprising over 200 million events collected across 12 hosts over two weeks, which we make available to researchers upon request. DOLLHOUSE captures real user activity, where participants used the machines for everyday tasks during the collection period. We also executed a spectrum of attacks *against these users on live systems*, ranging from simple credential stealers to day-long Advanced Persistent Threat (APT) scenarios. Finally, we re-evaluate state-of-the-art PIDSs to assess their performance in realistic user environments with long-running attacks.

I. INTRODUCTION

System provenance traces a detailed, tamper-evident audit trail of host activity by capturing the causal relationships among kernel-level entities such as *processes*, *files*, and *network connections* [1–3].

This rich semantic representation has emerged as a principled foundation for host-level security monitoring, enabling dynamic, behavior-based analysis that transcends the inherent limitations of traditional signature-based defenses [4–9]. This paradigm is especially effective for detecting APTs that unfold as multi-stage campaigns over days or weeks, because the causal graph can, in principle, link the earliest foothold of an intrusion to its downstream consequences [10–14]. Inspired

by this promise, a large and growing body of Machine Learning (ML)-assisted PIDSs has emerged, ingesting continuous streams of system audit logs and raising alerts when observed behaviors deviate from a learned baseline or match known attack patterns [15–22].

Recent PIDSs have achieved impressive detection performance on widely used benchmark datasets, most notably the DARPA Transparent Computing (TC) engagement datasets [23, 24], which have served as the field’s primary proving ground for the past decade. State-of-the-art systems report high detection accuracy with manageable false positive rates, timely alert generation [25], and interpretable outputs including root cause analysis and attack graph reconstruction [21].

Despite this apparent progress, PIDSs have seen limited deployment outside the research community. Recent critical reevaluations have illuminated troubling practices that contribute to the high reported performance. PIDS evaluations have pervasively overestimated the ground-truth attack scope [19, 21, 22, 26, 27], with detection performance metrics degrading sharply when evaluated against a more tightly-scoped ground truth [28]. Further, Bilot et al. [29] found that PIDSs frequently rely on benchmark-specific hyperparameter tuning that requires foreknowledge of the test set, which yields an unrealistically high estimate of detection efficacy. Beyond evaluation irregularities, Liu et al. [30] identified unnatural regularities in the benign activity of popular benchmark datasets, exacerbating overestimations of anomaly detection approaches. These results suggest that the performance gains reported on standard benchmarks reflect *overfitting* to benchmark conditions rather than genuine generalization.

A primary root cause of inflated benchmark performance is the quality of available datasets. The DARPA TC datasets, now six to ten years old [23], were collected under highly controlled and largely unrealistic conditions: benign activity consists predominantly of scripted or automated behaviors, and attack scenarios are compressed into short time spans to pack a wider variety of malicious activity into a limited collection window. As a result, these datasets provide little coverage of the low-and-slow APT campaigns that motivate provenance-based detection in the first place, and they misrepresent normal system behavior, since real users generate far more diverse and

unpredictable activity than automated workloads. Subsequent datasets have partially addressed individual shortcomings — for instance, ATLASv2 [31] introduces more realistic benign traces, and CARBANAKv2 [32] includes a week-long APT simulation with human users, but none have simultaneously delivered authentic user behavior, long-running attacks, and equivalent scale to the DARPA TC datasets. Building such a dataset is inherently difficult: beyond the technical and operational challenges of maintaining a stable, long-duration collection infrastructure, researchers must navigate stringent ethical and legal requirements, including sustained Institutional Review Board (IRB) oversight, that make it costly to capture the behavioral diversity found in real enterprise environments.

A second root cause is a fundamental mismatch between the dominant PIDS design paradigm and the nature of real-world threats. The central challenge of processing provenance data at scale is managing the intractably large dependency graphs that accumulate over extended system operation. Older PIDSs addressed this by backward- (or forward-) tracking from known Point-Of-Interest (POI) events to reconstruct relevant subgraphs [15, 16, 33, 34]. Modern systems instead partition audit logs into fixed-size time windows, typically on the order of fifteen minutes, and run detection independently over each slice [18, 19]. This *time-slice* paradigm offers practical benefits: it bounds computational cost, enables streaming inference, and avoids the need to presuppose knowledge of what entities will be involved in an attack. However, it is also strongly reinforced by the benchmark datasets on which systems are evaluated: because DARPA TC attack scenarios are short-lived and self-contained, time-sliced systems appear to work well. Time-slicing can fragment the semantic continuity of multi-stage attack campaigns, severing the long-range causal dependencies that may be essential for detecting persistent adversaries. This design choice is at odds with the original promise of provenance-based intrusion detection: catching attacks that unfold over days or weeks. Moreover, time-slice-based processing is further strained in large enterprise deployments where even short time windows contain massive event volumes. Attack duration should be varied to allow evaluation of the impact of time-sliced detector design.

These two problems have co-evolved into a self-reinforcing cycle. Research designs are optimized for existing benchmark datasets, and those benchmarks in turn validate and entrench the design choices. Without datasets that include long-running attacks embedded in realistic user behavior, there is no experimental basis to challenge the time-slice assumption, so the cycle perpetuates itself. Recent reevaluations [30] provide suggestive evidence that this cycle has inflated the perceived performance of academic PIDSs, but the dataset needed to close the argument has not previously existed.

In this work, we address this gap directly. Under IRB supervision and with informed consent from participants¹, we deployed system audit log collection agents on the personal

workstations of graduate students and research assistants, capturing over 200 million events across twelve hosts over two weeks. Participants used their machines for real everyday tasks to capture natural unscripted benign activity. In this real-world environment, we executed a spectrum of attack campaigns ranging from rapid credential stealers to day-long APT scenarios against these users’ live systems. To support reproducibility, DOLLHOUSE is available to researchers upon request. Using our new dataset — alongside the recently released CARBANAKv2 dataset [32] — we re-evaluate state-of-the-art PIDSs and find that systems which perform well on DARPA TC produce far too many false positives to be useful when benign activity is genuinely diverse, for both short and long-running attacks.

Our findings are a call to action for the provenance security community. The field has converged on design choices that are well-matched to an increasingly outdated set of benchmark conditions, creating a risk of architectural overfitting that is difficult to detect from within the existing evaluation regime. We urge the community to critically examine the biases embedded in current data collection practices, to establish recurring, living dataset engagements that evolve alongside adversarial techniques, and to invest in the ethical and operational infrastructure needed to sustain realistic provenance research at scale.

In summary, our contributions are as follows:

- We collected a large-scale system provenance dataset with over 200M events from twelve hosts collected over two weeks, capturing authentic user behavior and diverse long-running attacks under IRB-approved study conditions with informed user consent. The dataset is available to researchers upon request.
- We re-evaluate state-of-the-art PIDSs on DOLLHOUSE and CARBANAKv2 using threshold-free metrics, showing that every PIDS produces too many false positives on DOLLHOUSE to be practical when evaluated against unseen datasets.
- We analyze the co-evolution of datasets and methods in provenance-based security and argue for sustained, realistic data collection to mitigate architectural overfitting that has hindered progress.

II. BACKGROUND AND RELATED WORK

This section reviews system provenance, the evolution of PIDS, the datasets that underpin the field, and the temporal segmentation strategies that dominate current designs.

A. System Provenance Graphs

System provenance captures fine-grained interactions among kernel-level entities (processes, files, and network sockets) by recording system call activity on host machines. Monitoring infrastructure represents these interactions as a directed graph, where nodes denote system entities and edges encode causal dependencies. Formally, a provenance graph consists of timestamped edges $e = (u, v, r, t)$, where $u, v \in \{\text{processes} \cup \text{files} \cup \text{sockets}\}$, r denotes the relationship type

¹UTD-IRB-22-326

(e.g., READ, WRITE, EXECUTE, CREATE), and t is the event timestamp. An edge (u, v, r) asserts that v causally depends on u . Nodes and edges may carry additional attributes such as executable names, file paths, and IP addresses.

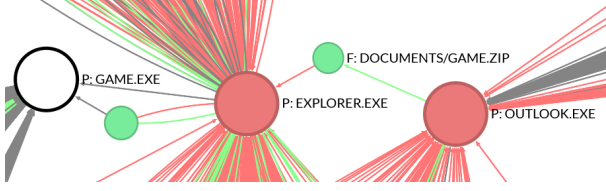


Figure 1: A provenance graph snippet depicting a user downloading, unpacking, and running an executable email attachment.

This causal structure makes provenance graphs well suited for security analysis. Analysts can anchor investigations at a POI event, such as execution of a suspicious binary, and traverse the graph both forward (to identify affected entities) and backward (to reconstruct the attack origin). This bidirectional traceability links initial footholds to downstream consequences throughout an entire campaign. Figure 1 shows a small portion of one such provenance graph that was rooted in the execution of a binary named GAME.EXE, which traces its source back to a zip archive attached to an email. Beyond forensics, provenance graphs provide a principled foundation for behavior-based anomaly detection. Structural patterns and attribute correlations are difficult for adversaries to manipulate without inducing detectable deviations.

B. System Provenance Datasets

The effectiveness of provenance-based security research depends critically on dataset realism and quality. Collecting system provenance at scale is inherently challenging. A single host may generate tens of millions of events per day, requiring efficient ingestion pipelines, scalable storage, and consistent schema design to preserve causal fidelity over long durations. Table I enumerates the prior research on PIDSs and their associated datasets for detection model training and evaluations. **DARPA TC datasets.** The DARPA TC [35] program produced the most widely used benchmarks in this space. Across five engagements, red teams executed attack scenarios in controlled environments while performers collected system-level telemetry using a common data model (CDM). The most commonly used datasets, Engagement 3 (TC3) [36] and Engagement 5 (TC5) [37], span Windows and Linux hosts such as FiveDirections [38], THEIA [23], and TRACE [39]. Each involves 3 to 5 hosts over roughly 11 to 13 days. Attack scenarios include spearphishing, credential theft, lateral movement, and data exfiltration, while benign activity is generated primarily through automated scripts.

Subsequent datasets. Later efforts address specific gaps. OpTC [24] expands scale and environmental realism. StreamSpot [40] supports streaming evaluation. ATLASv2 [31] incorporates real user activity but remains limited in scope and attack duration. CARBANAKv2 [32] increases attack diversity. However, no existing dataset simultaneously captures

long-running attacks, authentic user behavior, and multi-host enterprise scale.

Operational constraints. Constructing realistic provenance datasets remains fundamentally difficult. Beyond engineering challenges, researchers must satisfy strict ethical and legal requirements, including sustained IRB oversight and informed consent. These constraints significantly increase cost and complexity, often forcing trade-offs such as shorter collection windows, simplified attack scenarios, and scripted benign workloads that avoid privacy risks.

C. A Brief History of PIDS

Table I: Curated provenance research that utilized DARPA datasets.

System	Datasets	Granularity	Event Batching
HOLMES [6] (2019)	DARPA TC E3 + In-house	Edge	Online
MORSE [8] (2020)	DARPA TC E3	Edge	Online
UNICORN [34] (2020)	DARPA TC E3 + StreamSpot + In-house	Batch	Time-Slice
SHADEWATCHER [41] (2022)	DARPA TC E3 + In-house	Edge	Behavior-Centric
PROGRAPHER [42] (2023)	DARPA TC E3 + StreamSpot + ATLAS	Batch	Sliding Window
DISTDET [43] (2023)	DARPA TC E3 + Industry Arena	Node	POI-Centric
FLASH [19] (2024)	DARPA TC (E3, OpTC)	Node	Node-Centric
R-CAID [21] (2024)	DARPA TC E3 + StreamSpot + ATLASv2	Node	Node-Centric
NODLINK [44] (2024)	DARPA TC E3 + Industry Arena + In-house	Node	Time-Slice
MAGIC [45] (2024)	DARPA TC E3 + StreamSpot + Unicorn Wget	Batch	Externalized
KAIROS [27] (2024)	DARPA TC (E3, E5, OpTC) + StreamSpot	Batch	Time-Slice
CAPTAIN [25] (2025)	DARPA TC (E3, E5)	Edge	Online
ORTHURUS [28] (2025)	DARPA TC (E3, E5)	Node	Time-Slice
VELOX [29] (2025)	DARPA TC (E3, E5, OpTC)	Edge	Online

Early provenance-based intrusion detection systems focused on automating forensic analysis by leveraging graph structure to reconstruct attack campaigns from POI events. These systems typically operated at the graph level, classifying entire provenance graphs as benign or malicious [15–17, 33, 34].

As the field matured, this coarse granularity proved insufficient for operational use. A single malicious interaction embedded in millions of benign events cannot be localized from a binary graph label. This limitation drove a shift toward finer-grained evaluation. Systems such as ORTHURUS [28] introduced node- and edge-level labeling, enabling precise performance measurement and exposing weaknesses in earlier approaches. This transition is often framed as improving the *quality of annotations (QoA)*. It raised the standard for evaluation and prompted reexamination of prior results.

In parallel, modeling techniques became increasingly sophisticated. Representative systems span diverse approaches. SLEUTH [17] leverages graph traversal and summarization. ShadeWatcher [41] formulates detection as knowledge-graph reasoning. MAGIC [22] performs multi-granularity causal analysis. KAIROS [18] applies graph neural networks over time-sliced graphs. R-CAID [21] emphasizes interpretable root-cause analysis. VELOX [29] targets real-time detection. On benchmark datasets, these systems report high detection accuracy, manageable false positive rates, and interpretable outputs.

Despite this progress, real-world adoption remains limited. Recent reevaluations reveal significant performance degradation under more realistic conditions, including node-level ground truth and cross-dataset testing without benchmark-specific tuning [29, 32, 46]. These findings suggest that *many systems overfit to benchmark characteristics rather than learning generalizable behaviors*. We examine this hypothesis directly.

D. Temporal Segmentation of System Logs

Scalability remains a central challenge in provenance-based detection. Provenance graphs grow continuously, and analyzing multi-day dependency structures in full is computationally infeasible.

Early systems mitigated this by restricting analysis to subgraphs anchored at known POIs. While effective for forensics, this approach assumes prior knowledge of suspicious entities and limits applicability to proactive detection.

Modern PIDS instead rely on *temporal segmentation*. They partition audit logs into fixed-size windows, typically on the order of minutes (usually 15 minutes), and perform detection independently on each resulting subgraph. This approach bounds computational cost, enables streaming operation, and removes reliance on predefined POIs.

However, temporal segmentation fundamentally conflicts with the strengths of provenance analysis. Time slicing fragments causal continuity. Multi-stage attacks that span multiple windows are split into disconnected subgraphs, which severs long-range dependencies between early and late stages. As a result, detectors may struggle to capture the full lifecycle of persistent attacks.

This limitation is reinforced by benchmark design. In DARPA TC datasets, attack scenarios are short-lived and typically fit within a single time window. Time-sliced systems therefore perform well under these conditions. The datasets validate the assumptions they embed and create a feedback loop between evaluation and system design.

Long-running attacks are where this weakness would matter most. Their behavioral footprint spreads across many windows, which can prevent accumulation of evidence and break causal chains at window boundaries. Detecting such attacks would require maintaining long temporal context, which undermines scalability, or accepting reduced visibility.

At enterprise scale, the problem intensifies. Even short time windows may contain massive volumes of events across many hosts. This strains both computation and storage and weakens the original motivation for segmentation.

These observations reveal a self-reinforcing cycle. Systems optimize for available benchmarks, and benchmarks reinforce those design choices. Breaking this cycle requires datasets that capture long-running attacks within realistic user environments. This objective is central to our work.

III. DATA COLLECTION AND METHODOLOGY

To evaluate State-Of-The-Art (SOTA) PIDS in realistic test conditions, we constructed a testbed network that simulated a

Table II: Schema mapping between DARPA and in-house dataset.

	DARPA	In-house
Nodes	Subject	Process
	NetflowObject	IPChannel
Edges	FileObject	File
	Event (Subject → Subject)	ProcessEvent (Process → Process)
	Event (Subject → FileObject)	FileEvent (Process → File)
	Event (Subject → NetflowObject)	IPChannelEvent (Process → IPChannel)

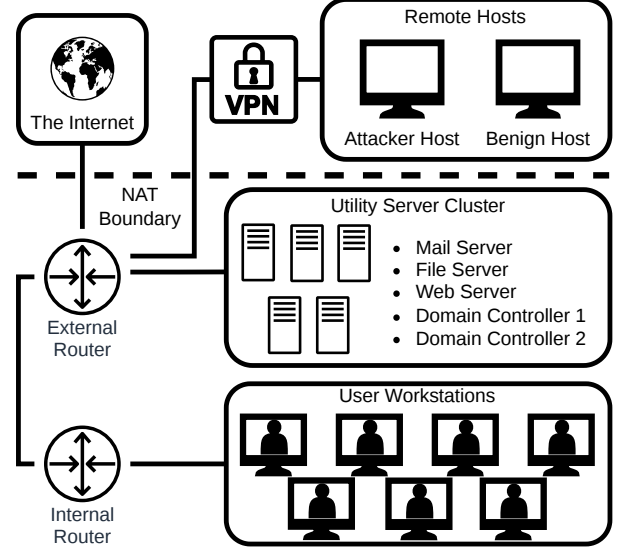


Figure 2: Attack network layout. Connections between routers and to outside the NAT boundary use WireGuard VPN tunnels, and connections to internal servers and workstations use physical links.

small organization’s IT infrastructure using freshly provisioned physical workstations and virtualized utility servers as shown in Figure 2. With IRB¹ approval, we recruited five participants to migrate to using the instrumented workstations as their primary work devices for two weeks. The first week of data collection was attack-free to serve as a clean benign training set, and in the second week we conducted live attacks against the user workstations. These attacks (detailed in §III-B) included both fast and slow variants of common information stealers and reverse shells. Through this experimental setup and by making the collected data available upon request, our study aims to address the scarcity of datasets for PIDS research.

A. Data Collection Network

The data collection network illustrated in Figure 2 represents a simulated enterprise environment with seven workstations, five servers, and transparent routers. Five of the workstations were regularly used by participants, one was idle, and one was used sparsely to administer the Active Directory environment. All workstations and servers were instrumented with a system provenance data collection agent that condenses host-level logs (Linux Audit [47] and Windows ETW [48]) into event streams that are streamed to a PostgreSQL database. We mapped the event streams provided by our data collection

agent to the DARPA format by following the schema translation guidelines in Table II.

Physical workstations were used by participants of the experiment to complete their typical workloads. The seven freshly provisioned workstations were physically connected to an internal network router through ethernet. Each workstation ran Windows 11, and was preloaded with common office applications (Microsoft Office, Adobe Acrobat, Slack, *etc.*) and web browsers (Firefox, Edge, and Chrome), as well as additional programs requested by the participants (*e.g.*, Google Antigravity, VS Code, JetBrains Rider, and Discord).

The five utility servers (mail server, file server, web server, and primary/backup domain controllers) were virtualized using ProxMox on a single physical host connected to an external network router through ethernet. The web server and mail server ran Linux on Ubuntu 24.04 LTS, and the file server and domain controllers ran Windows Server 2025.

Outside the simulated enterprise environment, we provisioned two uninstrumented remote hosts to provide benign and malicious external connections. The attacker host ran an HTTP server and an FTP server to receive exfiltrated files, and used the Metasploit Framework [49] for managing reverse shell sessions and developing attack payloads. The benign host periodically accessed the enterprise web server and sent emails with attachments to the simulated enterprise environment to provide contrast against the attacker’s phishing emails.

The internal and external routers were bridged via a WireGuard VPN tunnel. All enterprise devices were connected to the internet through a single gateway router using an external VPN to isolate external traffic from the university network. The remote hosts were directly connected to the gateway router via WireGuard VPN tunnels. These hosts were behind NAT with port-forwarding rules for the web server and mail server to simulate external internet hosts without exposing insecure services to the internet.

B. Attack Scenarios

During the second week of data collection, we executed a series of custom-made attacks including file stealers, credential stealers, and reverse shells on physical systems, mirroring the evaluation methodologies of the DARPA Transparent Computing program. All 22 attack instances we ran are detailed in Table IX. Initial access was achieved through spearphishing links distributed through a compromised account on the organization’s self-hosted messaging service and a malicious attachment from an external email account. To isolate the behavioral detection of PIDS, all the machines had Microsoft Defender disabled to simulate a scenario where antivirus and other static detectors were bypassed or failed to detect the malicious payload.

File Stealer and Browser Credential Stealer. These scenarios (Figure 3) were engineered to be similar to MITRE Enterprise 2025 Initial Access Malware Execution Operational Flow [50]. We deployed an archive with a `report.pdf` and a malicious `report.pif` executable. To reduce suspicion from the victim, the malicious `pif` file opened the decoy PDF file

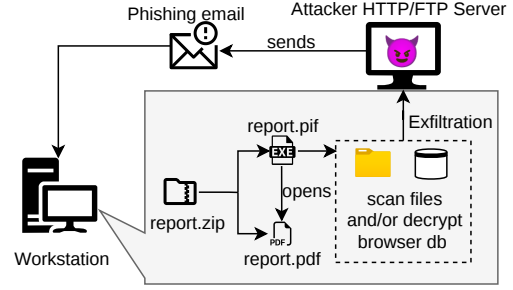


Figure 3: Document/Credential stealer attack operational flow.

after execution. The file stealing attack traversed the victim’s documents folder and created a temporary zip file to exfiltrate through HTTP and FTP protocols to the attacker’s server. Similarly, the credential stealer attack went through a victim’s Firefox, Edge, and Chrome profiles and SQLite databases to write passwords to a temporary text file for exfiltration to the attacker’s server. To add a temporal component, we also ran the file stealer and credential stealer with added 30-minute delays between each file read, extending their execution times to over 8 hours. Success was determined by the exfiltration of passwords or files.

These stealer scenarios are deliberately designed to be simple to detect, with paths like `sensitive_file` → `malware.exe` → `10.11.12.13` providing an obvious two-hop attack chain in the provenance graph.

Interactive Shells and Command and Control. Interactive shell and Command and Control (C2) attacks provide adversaries with persistent remote access to compromised hosts, enabling multi-stage operations that unfold over extended periods of time. We implemented representative reverse-shell and C2-based attack scenarios to evaluate how well PIDS preserve long-range behavioral dependencies across temporally separated malicious activities.

Metasploit Reverse Shell. This attack was performed by sending a phishing link to an unsuspecting employee who downloaded a Windows executable program and ran it. The attack used the reverse TCP module from Metasploit. These attacks were considered a success if we were able to exfiltrate important credentials such as SSH keys, local documents, mapped network share data, or log onto SSH servers and run commands. In one instance, this attack lasted over 18 hours. This was intentionally done to demonstrate the effects of relying on temporal proximity. We initially investigated local directories and known SSH hosts, and logged onto an SSH server and ran commands 14 hours later.

Custom Command and Control Worm. This attack was distributed similarly to the Metasploit reverse shell where an employee downloaded the program via a phishing link or a malicious attachment forwarded from an infected employee. This was a program that communicated with a command and control server over HTTPS that sent the malware client a

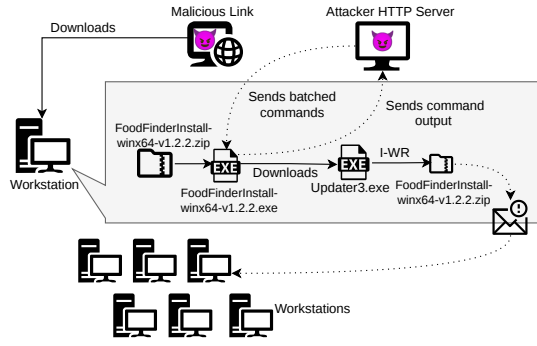


Figure 4: Command and control worm operational flow.

command to run and a delay for the next command. The commands given by the custom C2 server stole SSH keys and downloaded and ran an external payload. This external payload was an email spammer that used credentials found on the system to spread the original C2 payload through the victim’s email account. Success was determined by exfiltrating data and executing the external payload.

The custom C2 attack (Figure 4) was executed under two temporal conditions. In the first condition, all the operator commands were run consecutively, completing the attack within minutes. In the second condition, a one-hour delay was included between each command, extending the attack duration to over 6 hours. This methodology was designed to expose the limitations of current PIDS, which overly rely on temporal proximity for intrusion detection.

C. Benign Workloads

Throughout the two-week experimental period, participants engaged in a range of typical work activities that generated a substantial volume of provenance data. In addition to their standard workloads, study participants were encouraged to use the provided internal email server for communication with other participants, and to deploy and consume services via the internal web server and file server.

Office Tasks. Participants performed a variety of routine administrative and academic tasks common in an office environment. This included academic writing using local $\text{\LaTeX}$ editors (e.g., VSCode) and browser-based platforms (e.g., Overleaf), as well as preparing PowerPoint presentations for group meetings. Web browsing constituted a significant portion of the activity, ranging from technical research and literature review to general-interest news. Participants actively used the internal mail server for correspondence within the simulated environment, generating realistic traffic between the workstations and the utility servers.

Software Development. The software development workload involved the creation and maintenance of small-scale programs and scripts. Participants used VSCode, JetBrains Rider, and the Antigravity AI coding assistant to write primarily Python and C# code. The development process included writing

code, managing dependencies, and performing both local and remote testing. Remote development was a key component, with participants using Docker containers for environment isolation and connecting to the Ubuntu-based web server via SSH for deployment and debugging. Frequent documentation research via web browsers accompanied these development tasks, reflecting the information-seeking behavior typical of professional developers.

System Administration. Some participants engaged in system administration tasks both within the simulated network (e.g., reconfiguring router settings through the OPNSense web interface from a workstation, and RDPing into the domain controllers to modify configuration settings), and on remote Linux hosts (e.g., managing a remote Linux server used for CTF events). These activities included configuring network services, managing user permissions, and maintaining an on-line CTFd instance for competition administration. These tasks introduced complex process trees and network connection patterns such as bulk file transfers, configuration file edits, and service restarts that contrast with routine office work.

D. Limitations of DOLLHOUSE

DOLLHOUSE does not capture the full breadth of activity in a real enterprise. Our five participants were graduate students in a research lab, so the benign data is mostly software development, academic writing, and general office work, and does not include other roles or the scale of a large organization. Participants also knew their workstations were being monitored, which would have some effect on their behavior. However, the workloads we recorded and the services we ran would be expected in real enterprise networks. Additionally, our attacks do not represent complex state-of-the-art APT scenarios, as they were deliberately simple to set a low bar for detection. Our results in §IV show that even our simplest attacks are very difficult for PIDSs to detect, so this is not a significant limitation.

E. Ground Truth Labeling

After executing each attack, we identified the initial malicious process and its start time by querying our database for the payload filename, accessed files, and attacker IPs. We performed forward and backward tracking queries from this point to retrieve associated files, processes, and netflows. Then, using an in-house visualization tool we systematically examined these graphs to find all nodes involved in the attack chain, which we individually labeled with the relevant attack number.

We labeled the malicious process itself, files or processes created by the malicious process, and IP connections to the attacker as malicious, along with otherwise benign files or processes on the system that were a part of the attack chain. For example, processes that were used to download or unzip payloads, such as Outlook, Firefox, or 7-Zip, were labeled as malicious due to their interaction with the malicious payload. Notably, not all one-hop relatives of malicious nodes are labeled; if a malicious process reads `ntdll.dll`, `ntdll.dll`

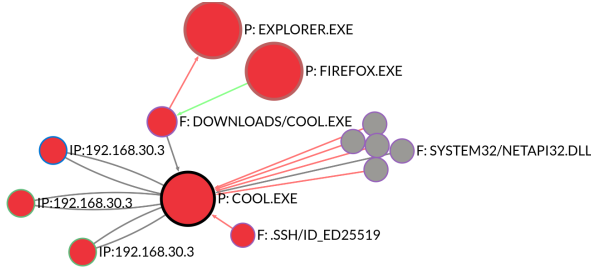


Figure 5: Malicious process visualization. Malicious nodes are shown in red and benign nodes in grey. 192.168.30.3 is the IP of the simulated attacker.

is not labeled as malicious, as it is not part of the causal chain of the attack. An example of these labels is shown in [Figure 5](#). Together these nodes represent the attack flow, and they serve as the ground truth for evaluating the detection performance of PIDS.

We conducted and labeled a total of 22 separate attacks, comprising 929 total malicious nodes. Labeling was performed by the attacker shortly after each attack. The attacker had full knowledge of all files, processes, and IP addresses associated with malicious activities. These labels were later verified by other researchers.

F. Detection Model Training and Deployment

All provenance-based intrusion detection systems evaluated in this study were trained and deployed on a dedicated compute server running Ubuntu 24.04 LTS. The server was provisioned with an AMD EPYC 9224 processor, 220 GB of system memory, and a single NVIDIA RTX A6000 GPU to accelerate both model training and inference.

For the core model implementations, we followed the baseline PIDSs provided by the PIDSMaker framework [51]. However, due to the scale of the generated event streams and the inherent resource constraints of processing complex provenance graphs, we introduced several performance and memory optimizations into the original PIDSMaker pipeline². These optimizations were essential to ensure that the detection models could scale effectively to the volume of data collected during our two-week evaluation period without exceeding our hardware limitations. These optimizations involved reducing the amount of data loaded into memory at once by streaming the dataset from disk rather than loading the entire dataset into memory, deleting fields that were not used by the model, and partitioning disconnected subgraphs to fit them in GPU memory for GNN-based models that did not implement their own batching strategies beyond splitting time windows. Even with these optimizations, we were still unable to run every model due to our VRAM constraints. Bilot et al. [51] were unable to evaluate all PIDSMaker models in their work even with 1 TB of RAM and an A100 GPU with 80GB VRAM.

²We intend to submit a pull request to share these optimizations with the community.

IV. EVALUATION

A. Dataset Summary

Table III: DOLLHOUSE dataset summary statistics. Weekly counts include nodes that appear in both weeks, so their sum exceeds the total count.

	Type	Week 1	Week 2	Total
Nodes	Subject	191,465	445,778	635,719
	File	1,488,931	2,317,221	3,258,836
	Netflow	1,283,566	1,465,807	2,218,745
	Total	2,963,962	4,228,806	6,113,300
Edges	Subject ↔ Subject	361,013	869,054	1,230,067
	Subject ↔ File	106,230,664	26,064,858	132,295,522
	Subject ↔ Netflow	32,815,919	37,844,833	70,660,752
	Total	139,407,596	64,778,745	204,186,341

The DOLLHOUSE dataset was collected from our simulated enterprise environment as described in [§III-A](#) over a period of two weeks from April 6th, 2026 to April 20th, 2026. The earliest event was logged at 05:18 UTC on the 6th and the last event was logged at 04:59 UTC on the 20th. Our entire two-week data collection produces a provenance graph with over 6 million nodes and 200 million edges, detailed in [Table III](#). The amount of data collected varies between the two weeks. This is in part due to activity generally increasing over time as workstations were rarely rebooted, and in part due to the deployment of a Python web server during the first week that repeatedly reloaded files on disk, causing a single process to produce 89 million file-read events over the course of a few hours, as shown in [Figure 6](#). There were multiple instances where we experienced complete or partial network outages that prevented new events from being logged from some or all hosts; however, none of these outages interrupted ongoing attacks.

In addition to our DOLLHOUSE dataset, we also evaluate against the CARBANAKv2 dataset [32]. CARBANAKv2 was conducted in an environment similar to that of the DOLLHOUSE dataset with multiple users and workstations; however, it ran a single, long-running APT attack simulation over multiple days in contrast to the multiple shorter attacks in DOLLHOUSE. CARBANAKv2 only includes a total of 17,143,304 events, much lower than the 204,186,341 we observed in our own engagement. From the distribution of events over time in [Figure 7](#), it appears that workstations were shut off overnight and on weekends, whereas our workstations were rarely rebooted and automatic sleep was disabled. Additionally, CARBANAKv2 utilized a commercial EDR framework to collect logs while we used syscall-level logs from ETW and Auditd, leading to higher event volume. Both datasets represent reasonably realistic scenarios with diverse characteristics, allowing us to evaluate the generalization capabilities of the different PIDS. We do not present an evaluation on the popular DARPA TC benchmarks because these implementations have already been evaluated against those benchmarks in a prior replication study [29].

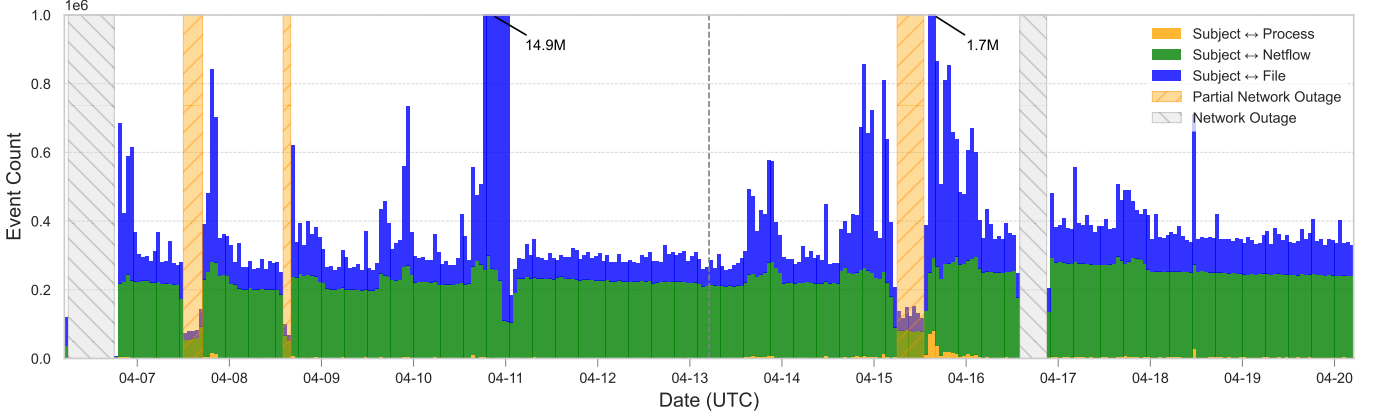


Figure 6: Events collected each hour during the two weeks of DOLLHOUSE data collection. Counts of different event types are stacked. Clipped values ($> 1,000,000$) are annotated with the peak hourly value reached across sequential outliers.

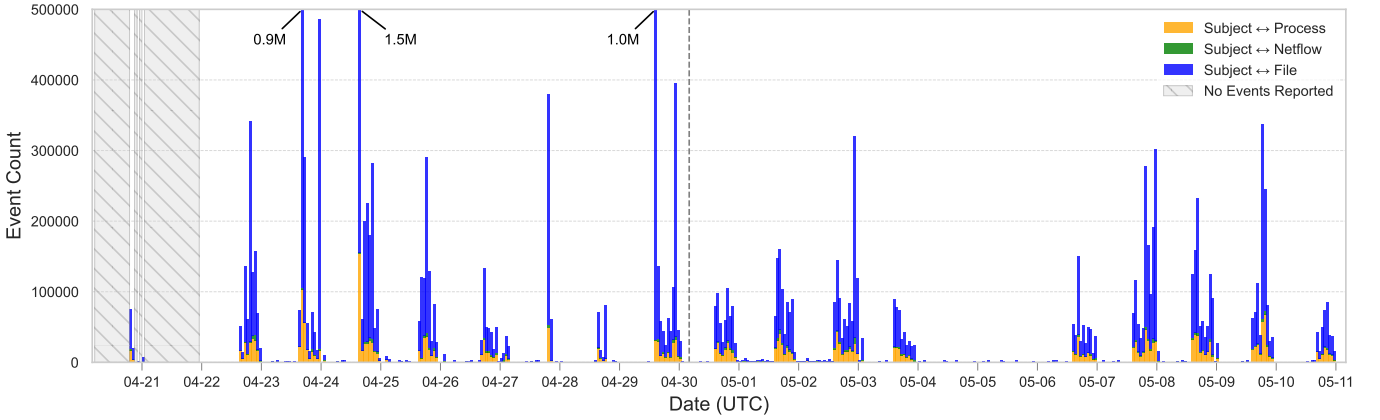


Figure 7: Events collected each hour during CARBANAKv2 data collection. Counts of different event types are stacked. Clipped values ($> 500,000$) are annotated with the peak hourly value reached across sequential outliers.

B. Metrics

Threshold-dependent metrics. In the literature, it is common for PIDS to be evaluated as binary classifiers (attack vs benign) on whatever granularity they operate (*e.g.*, edges, nodes, graphs, batches, *etc.*). Resolving anomaly scores to a concrete set of binary predictions requires picking a concrete detection threshold, which limits the scope of the evaluation while also increasing variance and reducing transparency [29]. For completeness, we report node-level threshold-dependent metrics (precision, recall, true positive count, false positive count) for each PIDS using the default thresholding scheme provided by its implementation.

Threshold-free metrics. To focus our evaluation on the intrinsic detection capabilities of the evaluated PIDS, we primarily focus on threshold-free metrics. ADP (Attack Detection-Precision) [29] is a summary statistic over the proportion of attacks detected at a given precision value over all possible thresholds.

$$ADP = \int_0^1 D(p)dp,$$

where $D(p)$ is the percentage of detected attacks at precision p . We also propose a new evaluation metric FP_n^* , which is the minimum number of false positives required to detect n attacks. This FP^* metric traces the Pareto-frontier of the trade-off between the number of detected attacks and the number of false positives as the threshold decreases. Importantly, the interpretation of FP^* is incredibly intuitive; it is simply the number of alerts an analyst would need to manually investigate to find each attack. FP_1^* is the number of false positives if the model’s threshold were set at the highest value that still detects one malicious node from one attack. FP_{max}^* is the number of false positives if the model’s threshold were set to detect at least one malicious node from every attack. For comparison with prior work, we also report node-level AUROC and AUPRC.

We additionally report an incident-level AUPRC to account for certain shortcomings with the ADP metric. While ADP effectively summarizes model performance in an unbalanced dataset and rewards models for detecting any node in each attack rather than prioritizing node-level recall, it still rewards models for node-level precision, and produces very small

scores for models with very low precision, which makes them difficult to compare. For example, in the CARBANAKv2 dataset there is a single attack containing 215 nodes. If a model achieved $FP_1^* = 2$ but assigned all but 1 malicious node very small anomaly scores, it would have an ADP score of ≈ 0.34 . A second model with $FP_1^* = 215$ that assigned the same anomaly score to all 215 nodes (i.e. it detected the entire attack at the same threshold) would have an ADP score of ≈ 0.5 . The ADP scores indicate that the second model is superior, however an analyst would have to investigate 100x more alerts to discover the attack with the second model than with the first. Incident-level AUPRC addresses this issue by only considering the first malicious node detected in each attack in the AUPRC calculation. Other malicious nodes are dropped from the calculation altogether. This better reflects real-world usage where an analyst can uncover an attack from a single alert in the attack chain. When there is only one attack, incident-level AUPRC is equal to $1/(FP_1^* + 1)$, so the first model in the example above would score 0.33 and the second would score 0.005.

C. Results

We trained each PIDS available in PIDSMaker [29] on the DOLLHOUSE and CARBANAKv2 datasets using the default hyperparameters provided by the framework which were tuned for DARPA datasets. We do not do our own hyperparameter tuning as the methods for doing so involve test-set snooping, making dataset-specific hyperparameter tuning an unrealistic option for real-world deployment. We assign the first 5 days of the DOLLHOUSE dataset to the training set, the next 2 days to the validation set, and the remaining 7 days to the test set. Attacks were only carried out in the last 7 days of the dataset; the training and validation data are fully benign. CARBANAKv2 provides a train/validation/test split with their data distribution for PIDSMaker. We use this split as-is; it consists of 8 days of benign training data, 2 days of benign validation data, and 7 days of test data with attacks.

The results of our experiments on DOLLHOUSE are presented in Table IV. We report the threshold-dependent metrics using the default thresholds provided by each model, as well as the threshold-free AUROC, AUPRC, incident-level AUPRC, ADP, and FP^* metrics. Figure 8 summarizes the full range of FP_n^* values for each model alongside estimated scores for a random baseline (the expected performance of randomly assigning anomaly scores to all nodes).

The results on CARBANAKv2 are reported in Table V. The FP^* curve is not shown as there is only a single labelled attack in the dataset.

D. False Positive Analysis

Table VI breaks down the false positives (evaluated at the FP_1^* threshold) on the DOLLHOUSE dataset by node category and rule. The regular expression for each rule is listed in Table VIII in the appendix. For each rule, we report the raw

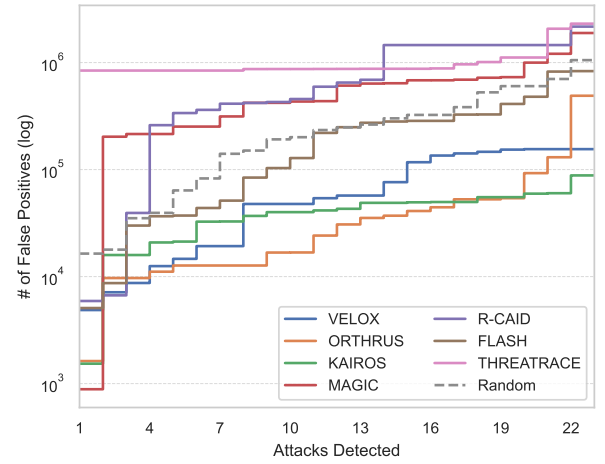


Figure 8: FP^* curve for DOLLHOUSE. Lower is better.

count and the *Risk Ratio* (RR) of the rule with respect to false positives:

$$RR = \frac{P(FP \mid \text{rule})}{P(FP \mid \neg \text{rule})}$$

A Risk Ratio greater than $1\times$ indicates that nodes matching the rule are disproportionately likely to receive a false-positive alert compared to nodes that do not; values exceeding $10\times$ (shown in **bold** in the table) indicate an extreme concentration.

Table VI reveals that each PIDS is sensitive to an entirely different class of benign activity, reflecting the distinct representations of normality learned during training. VELOX produces a relatively small total false positive count (4,869) but concentrates them sharply: JetBrains Rider-related *file* nodes account for 3,051 FPs with an RR of **215** $\times$ and loopback *Netflow* nodes contribute 308 FPs at an RR of **364** $\times$. Similarly, R-CAID's 5,916 false positives are exclusively Netflow nodes, with loopback traffic receiving an RR of **22** $\times$. This sensitivity to loopback traffic is particularly problematic in modern development environments; of the 114,382 loopback Netflow nodes in the DOLLHOUSE test set, 92,380 ($\approx 81\%$) are associated with a Google Antigravity process (typically the language server), whose routine inter-process communication is misidentified as anomalous. KAIROS raises most of its 1,535 false positives on Git *process* nodes (1,049; RR **22** $\times$) and on cache-related *file* nodes (79; RR **100** $\times$), suggesting that version-control and build-cache activity common in a developer workload falls well outside the distribution of behavior that KAIROS considers normal. MAGIC has the fewest false positives of any PIDS (886), but over a quarter of them are Windows scheduled task files (241; RR **591** $\times$), along with smaller concentrations on loopback *Netflow* nodes (10; RR **17** $\times$) and updater *process* nodes (90; RR 4.7 $\times$). In contrast, THREATTRACE generates a much larger total false positive count (842,146), but spreads them more broadly across categories. THREATTRACE shows only slight elevations for log, *LaTeX*, and Rider *file* nodes (RRs of 2.0 $\times$, 2.8 $\times$, and

Table IV: PIDS detection results for DOLLHOUSE with node-level ground truth labels. NODLINK results are not available due to memory constraints.

Model	Precision	Recall	TP	FP	AUROC	AUPRC	Inc. AUPRC	ADP	FP_1^*	FP_{max}^*
VELOX	0.0006	0.0043	4	7.2k	0.68	4.1×10^{-4}	2.1×10^{-4}	0.0007	4.9k	155.3k
ORTHRUS	0.0000	0.0000	0	148	0.62	4.9×10^{-4}	4.1×10^{-4}	0.0009	1.6k	489.8k
KAIROS	0.0000	0.0000	0	74	0.66	1.0×10^{-3}	2.9×10^{-4}	0.0024	1.5k	88.2k
MAGIC	0.0002	0.5145	478	2.1M	0.49	2.1×10^{-4}	7.1×10^{-5}	0.0003	886	1.9M
NODLINK	-	-	-	-	-	-	-	-	-	-
R-CAID	0.0000	0.0000	0	218	0.51	4.6×10^{-4}	3.9×10^{-5}	0.0005	5.9k	2.2M
FLASH	0.0001	0.6243	580	3.9M	0.64	4.5×10^{-4}	8.4×10^{-5}	0.0006	5.1k	832.1k
THREATTRACE	0.0000	0.0463	43	961.6k	0.50	2.2×10^{-4}	1.2×10^{-5}	0.0003	842.1k	2.3M

Table V: PIDS detection results for the CARBANAKv2 dataset. R-CAID results are not available due to memory constraints.

Model	Precision	Recall	TP	FP	AUROC	AUPRC	Inc. AUPRC	ADP	FP_1^*
VELOX	0.0000	0.0000	0	0	0.31	2.6×10^{-4}	7.6×10^{-4}	0.0036	1.3k
ORTHRUS	0.0000	0.0000	0	11	0.84	7.9×10^{-4}	8.0×10^{-4}	0.0013	1.2k
KAIROS	0.0000	0.0000	0	0	0.89	7.7×10^{-4}	1.9×10^{-2}	0.0189	52
MAGIC	0.0001	0.3907	84	955.5k	0.26	9.9×10^{-5}	5.1×10^{-4}	0.0007	2k
NODLINK	0.0004	0.3163	68	183.3k	0.52	2.3×10^{-4}	3.7×10^{-5}	0.0005	26.8k
R-CAID	-	-	-	-	-	-	-	-	-
FLASH	0.0000	0.0000	0	0	0.50	1.5×10^{-4}	6.8×10^{-7}	0.0017	1.5M
THREATTRACE	0.0020	1.0000	215	106.9k	0.98	1.1×10^{-2}	3.6×10^{-2}	0.1034	27

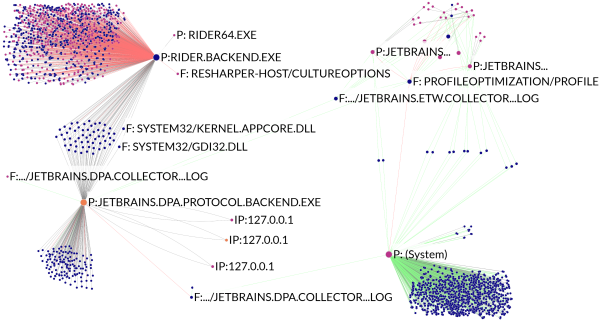


Figure 9: The provenance neighborhood of a JetBrains Rider process, color coded by number of PIDS alerts per node.

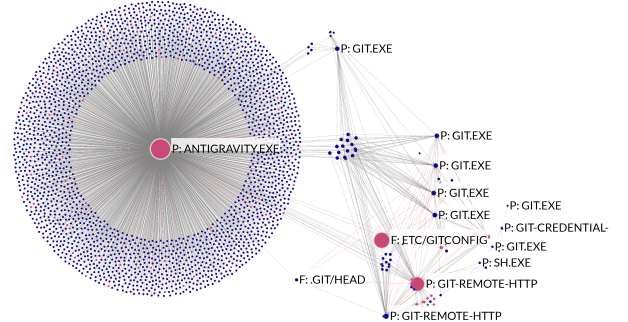


Figure 10: The provenance neighborhood of a GIT-REMOTE-HTTP process, color coded by number of PIDS alerts per node. This process has the highest anomaly score for KAIROS of any GIT process in the DOLLHOUSE dataset.

2.4×), and these relatively low Risk Ratios indicate a general over-fitting to a narrow notion of routine system activity rather than a specific sensitivity to individual program classes. This variance across systems confirms that each PIDS learns a distinct representation of benign behavior. The majority of programs that cause false alerts are present in substantial numbers in the training dataset, as reported in Table VII.

Figure 9 illustrates the provenance neighborhood of a JetBrains Rider process, which exhibits behavior that superficially resembles a data exfiltration tool. Specifically, the process establishes frequent connections to another process over the loopback interface; while this is a standard component

of Rider’s distributed architecture (which allows the backend to potentially run on a different host), it can be misidentified as unauthorized inter-process communication or network tunneling. Furthermore, the process’s routine operations involve loading many DLLs and producing a high volume of log files, creating a complex interaction pattern that multiple detection systems flag as anomalous. Both KAIROS and VELOX detect the Rider process as malicious when evaluated at their FP_1^* threshold.

Figure 10 illustrates a similar story for a different type of benign activity. KAIROS disproportionately flags GIT

Table VI: False positives (at threshold FP_1^*) by node type on the DOLLHOUSE dataset. Risk ratio (RR) values (in parentheses) that are $> 1\times$ indicate that the rule disproportionately predicts an alert.

Category	Rule	VELOX	ORTHRUS	KAIROS	MAGIC	R-CAID	FLASH	THREAT-TRACE
Netflow	Total	318	163	216	17	5,916	18	0
	Loopback	308 (364×)	3 (0.2×)	88 (8.1×)	10 (17×)	3,855 (22×)	0 (0.0×)	—
Subject	Total	852	445	1,231	176	0	5,010	0
	Git	38 (0.2×)	0 (0.0×)	1,049 (22×)	15 (0.4×)	—	1,312 (1.4×)	—
	Updater	17 (0.1×)	116 (1.6×)	3 (0.0×)	90 (4.7×)	—	869 (0.9×)	—
	Latex	213 (26×)	1 (0.2×)	25 (1.6×)	0 (0.0×)	—	29 (0.5×)	—
	Total	3,699	1,016	88	693	0	55	842,146
File	Log	39 (0.2×)	156 (2.8×)	0 (0.0×)	9 (0.2×)	—	0 (0.0×)	94,116 (2.0×)
	Cache	2 (0.0×)	36 (0.4×)	79 (100×)	6 (0.1×)	—	0 (0.0×)	83,096 (1.3×)
	JetBrains Rider	3,051 (215×)	0 (0.0×)	0 (0.0×)	1 (0.1×)	—	0 (0.0×)	42,349 (2.4×)
	Latex	0 (0.0×)	1 (0.0×)	0 (0.0×)	0 (0.0×)	—	0 (0.0×)	66,258 (2.8×)
	Scheduled Task	0 (0.0×)	0 (0.0×)	0 (0.0×)	241 (591×)	—	0 (0.0×)	0 (0.0×)
	Total FPs	4,869	1,624	1,535	886	5,916	5,083	842,146

Table VII: Occurrences of nodes matching the rules in Table VI within the training set of DOLLHOUSE.

Category	Rule	Matches
Netflow	Loopback	16,399
	Git	10,559
Subject	Updater	24,873
	Latex	37
File	Log	77,106
	Cache	184,717
	JetBrains Rider	58,227
	Latex	3,157
	Scheduled Task	1,987
	Total	842,146

processes due to their routine behavior of scanning local files and subsequently making network connections to remote repositories. In this case, the nearby GITCONFIG file was also flagged by THREATTRACE. Development and version control tools are more likely to be flagged by node-prediction based detectors due to their tightly coupled mix of files, processes, and network connections.

E. Attack Detection Performance

To make threshold-free comparisons of detection performance against specific attacks, we use a variant of the FP^* metric, defining $FP_{[k]}^*$ as the number of false positives that would be encountered before detecting a single node in the attack with ID k . All attack numbers are listed in Table IX.

Our attacks included both short-running and long-running attacks, designed with the intention of highlighting PIDS reliance on temporal proximity. However, we found no consistent difference between the two. For example, attacks 14 and 15 are slow versions of attacks 12 and 13 on the same hosts, yet the slow version was harder to detect for most PIDS on one host and easier on the other (see Table X).

Some models (KAIROS, ORTHRUS) performed worse on long-running variants of attacks, although this effect cannot be exclusively attributed to attack duration as the attacks occurred at different times so the background activity would have varied.

As a whole, PIDSs performed poorly on our attacks. The best performers by incident-level AUPRC were ORTHRUS, KAIROS, and VELOX (Table IV). MAGIC, R-CAID, and THREATTRACE performed similarly to random chance (AUROC from 0.49 to 0.51), and all three needed more false positives to detect every attack than randomly ordering every node in the graph would (FP_{max}^* of 1.1 million), as shown in Figure 8. Among the three best performers, the best result on a single attack was KAIROS on attack 17 with an $FP_{[17]}^*$ of 1,535, meaning an analyst would need to sift through over a thousand benign nodes before reaching a malicious node. The worst result among them was ORTHRUS on attack 15 (a long-running credential stealer worm) with an $FP_{[15]}^*$ of 489,800 (Table X).

Dong et al. [52] interviewed industry professionals who, on average, reported that they required < 0.1 alerts per day per host for a PIDS to be considered useful. The PIDS with the smallest FP_1^* when evaluated against DOLLHOUSE was MAGIC with $FP_1^* = 886$ on attack 5. This means, given a perfectly selected threshold to detect any malicious node, even the detector with the lowest FP_1^* still produces 10.5 alerts/day/host during our 7-day attack period, generating over 100x the number of alerts considered useful by industry professionals while still failing to detect 21 of the 22 attacks. PIDS remain unable to prioritize malicious nodes above the noise of benign anomalous activity, limiting their practical utility for analysts.

In contrast to the poor performance of all PIDSs evaluated against DOLLHOUSE, the best performing PIDS against CARBANAKv2 was THREATTRACE with an FP_1^* of only 27. Across their 6 hosts and 7-day attack period THREATTRACE

generates 0.64 alerts/day/host given a perfectly selected threshold to detect the attack, within an order of magnitude of the ~ 0.1 alerts/day/host industry requirement. This contrasts sharply with the catastrophically poor performance exhibited by THREATTRACE when evaluated against the DOLLHOUSE dataset, demonstrating the impact of dataset characteristics on PIDS performance.

V. LESSONS LEARNED

From our evaluation of the current PIDS landscape, we have identified the following lessons for the future of PIDS research. We group these along three axes: *datasets*, *evaluation design*, and *detector design*.

A. PIDS Datasets

Overreliance on specific benchmarks leads to structural overfitting. While the PIDS research community has made significant progress in improving attack detection rates on the DARPA datasets, our evaluation shows that this progress has not translated to generally strong detection performance on unseen datasets. Provenance dataset creators attempt to collect realistic data that accurately reflects production environments, but time and resource constraints result in unavoidable structural artifacts that exist only as a product of the constructed environment. Further, lacking a common framework for consistent data collection across benchmarks, different provenance datasets capture distinct subsets of system behaviors, with varying levels of granularity and semantic richness. Commendable progress in this direction has been made by [51], which attempts to create a common evaluation framework for PIDS, and should allow for the integration of additional provenance datasets in the future. We foresee a near-term need to produce an extensible open-source pipeline for provenance data collection to enable the community to generate and share provenance datasets more easily and consistently.

Benign data realism is the largest challenge in PIDS dataset construction. Prior criticisms of the DARPA PIDS benchmarks have focused on the predictable scripted behavior that constitutes the benign data [30]. Overly regular benign data leads to underestimation of false positive rates, particularly for anomaly-detection-based PIDS, which have continued to be the dominant paradigm. The gold standard for benign provenance data is logs from real users’ machines and the servers they interact with as they perform typical workloads unaware of supervision, but collecting such data faces logistical, ethical, and legal challenges (including privacy concerns). Further, to capture realistic corresponding malicious data, attacks must be conducted in the same environment that the benign data was collected in, which poses further ethical and legal constraints on the benign data collection environment. While our research faces these challenges directly, the PIDS research community will need to develop privacy-preserving mechanisms to streamline the collection and publication of realistic provenance datasets in the future.

Ground truth inconsistencies complicate PIDS comparison. Mapping traditional threat reports to precise node and

edge-level ground truth labels in a provenance graph is a labor-intensive and error-prone task. Even just considering the DARPA datasets, different works have used different mechanisms to produce ground-truth labels, which has been criticized in [28]; this is a symptom of DARPA not providing a canonical first-party ground truth labelset for their provenance graphs. To our knowledge, there is no community consensus on methods for accurately and consistently producing ground-truth labels for provenance datasets. While there is less ambiguity in labeling for edge-level data as compared to node-level data, producing edge-level labels is an order of magnitude more time-consuming than producing node-level labels. For example, if an otherwise benign Explorer.exe process is used to unpack a malicious archive, it is unclear whether that Explorer.exe process should be labeled as malicious; despite participating in the overall attack chain, that process did not perform any unusual behaviors. To support the production of new PIDS datasets, we foresee a need to develop automated edge-level ground-truth generation through instrumented attack payloads.

B. PIDS Evaluation

Evaluation metric interpretability under class imbalance. The challenge of accurate intrusion detection is intrinsically tied to the extreme class imbalance between benign and malicious system behaviors (often $1 : 10^5$ or worse, even in benchmark datasets which contain a high density of attacks by construction). This extreme class imbalance makes the practical utility of PIDS’s detections difficult to estimate because false negatives hold devastating consequences, but verifying alerts requires analysts to also inspect thousands or even hundreds of thousands of false positives, which quickly leads to alert fatigue [15]. Compounding with inconsistencies in labeling the ground truth attack components, research works in system provenance frequently define new metrics to highlight specific aspects of attack detection performance, but it is unclear how sensitive these metrics are to the specific contexts in which they are introduced [8, 25, 28, 29].

Data snooping. [29] found that several PIDS works tuned detection thresholds and feature extraction steps using data from the test set, leading to substantial inflation in reported performance metrics. As a result, no academically produced PIDS system provides a smooth “out-of-the-box” configuration for newly produced datasets. Determining good thresholds without data snooping remains a critical open challenge. To compare detection systems in isolation of their thresholding scheme, we recommend using threshold-free comparison methods that can tolerate extreme class imbalance. Alongside [29]’s ADP metric, we found incident-level AUPRC to be a useful threshold-free comparison scheme in the intrusion detection domain, as it does not reward a detector for finding many nodes from the same attack while avoiding the complexity of ADP. We investigate another, more verbose threshold-free comparison method in Table X in the appendix, which shows how many false positives were ranked above the most-highly-ranked malicious node in each attack chain.

C. IDS Design

Detecting long-running attacks. Broadly speaking, to detect long-running attacks, a PIDS can either rely on individual conspicuous events that are present in slow (but not covert) attacks, or correlate long event histories to determine that the aggregated behavior is malicious. In principle, the value of system provenance lies in the latter. However, this principle is undermined by stateless temporal segmentation or limited event batching. Counter to this intuition, our evaluation finds that even fully online PIDS that only consider a single edge at a time (*e.g.*, VELOX [29]) were able to detect (after many false positives) long-running attacks. The detectability of those attacks shows that there existed individual conspicuous actions that occurred over the course of the attack. A PIDS that sees more context should, in theory, be much more able to accurately classify these actions as malicious, but our evaluation found that modern PIDS still largely struggle. We believe that future works should consider two-stage PIDS design, with an initial streaming-based detection layer to identify “interesting” nodes and edges, followed by a more intensive historical analysis layer that tracks the causal history of those entities to provide additional context to a stricter alert generation layer.

Excessive resource consumption. Reproducibility of PIDS results is critical to research progress, but training overhead for many GNN-based PIDS can be out of reach for research groups with limited resources. In [29], the authors report needing over 400 days of computation on a powerful server with 1TB of RAM to run their reproducibility study, and even still had to omit some evaluation items due to resource constraints. We developed several performance optimizations for PIDSMaker [51] to enable us to run our evaluations on a server with 220GB of RAM and an NVIDIA RTX A6000 GPU with 48GB of VRAM, which is still a costly compute environment. Leaning into stream-based processing with stateful information persisted to disk is a promising design direction that is well-aligned with the current direction of PIDS research.

Beyond anomaly detection. Anomaly detectors are necessary for discovering previously unseen attacks, but they are unnecessarily limiting once the characteristics of an attack are known. Multi-stage threat detection research has been explored in the malware detection and classification domains with some success [4, 53, 54]. Similarly, we may be able to reduce the burden placed on provenance-based anomaly detectors by combining them with supervised detection models that target specific attack behaviors. This second-layer approach has been touched on in prior works (*e.g.*, HOLMES [6]), but has not been revisited outside of rule-based systems.

VI. DISCUSSION AND FUTURE WORK

The provenance research community is in a moment of consolidation, making now a pivotal point to set the direction for the next generation of PIDSs. We believe that establishing a stable foundation of fresh data collection will be critical to

advancing the field while avoiding pitfalls that were previously hidden by data scarcity.

Transparent provenance collection. Towards the broader direction of proliferating diverse provenance datasets that are easy to use, there is a need for developing common open-source provenance data collection frameworks. Current data collection methods primarily rely on proprietary software (*e.g.*, commercial EDR tools or in-house software with licensing complications) to extract streams of provenance events from host logs and to construct provenance graphs. By homogenizing the data collection pipeline in the public domain, we can reduce the overhead of translating between provenance data formats and improve seamless interoperability of PIDS on newly developed datasets. These developments would lay the foundation for the construction of reproducible data collection setups for industry and academic researchers to produce provenance data for modern applications and novel attacks. We intend to continue collecting data in our environment and to periodically release new versions of DOLLHOUSE with updated benign workloads and attacks.

Anonymization of provenance data. Several provenance works [6, 34, 41, 44, 55] collect, but *do not release* provenance data due to privacy concerns. The privacy implications of releasing graph-structured system log data are not well understood, which makes it exceedingly difficult to secure approval and consent to collect and release system provenance data, particularly when the activities to be captured are not highly scripted. By developing privacy analysis frameworks and anonymization techniques for system provenance data, the academic community can pave the way for broader releases of provenance data. In conjunction with transparent and accessible provenance collection frameworks, the ability to more freely share provenance data would largely solve the problem of provenance data scarcity that has resulted in the current myopic focus on the DARPA datasets.

Improving dataset diversity. One dataset cannot be a new standard benchmark for PIDSs; we do not intend to advocate for DOLLHOUSE as such a standard. Instead, we argue that PIDSs should be evaluated on several datasets with different characteristics. CARBANAKv2 [32] contains EDR logs of a single complex long-running APT scenario in an enterprise-like network, ATLASv2 [31] contains system logs of multiple CVE-based attacks against two Windows 7 hosts, and DOLLHOUSE contains simpler attacks executed repeatedly against a larger network of Windows 11 and Linux machines. Each of these captures different aspects of real environments and threats, and together they provide a better estimate of real-world detection performance than any one alone. Heavily relying on any single dataset for evaluation invites issues such as structural overfitting discussed earlier.

VII. CONCLUSION

In this work, we collected and analyzed a novel system provenance dataset, DOLLHOUSE, which captures two weeks of real-world benign user data, including well-documented attacks against those live user environments. Our work addresses

well-known limitations of the prevalent DARPA provenance datasets and combats data scarcity in the system provenance domain. Through a threshold-free replication study, we find that the steady progress of PIDS as measured by DARPA benchmark performance has not translated to general improvements in detection against unseen datasets, and is instead a result of structural overfitting to dataset construction artifacts. Rather than just providing another benchmark to optimize for, our findings are a call for the PIDS research community to invest effort into building dataset creation infrastructure and promoting data reuse to improve the robustness and generalizability of future security systems.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by National Science Foundation grant 2331424, and Eugene McDermott Graduate Fellowships 202208, 202504, and 202405. This work is supported in part by federal grant numbers: 69A3552344812 and 69A3552348317. It is based upon the work supported by the National Center for Transportation Cybersecurity and Resiliency (TraCR) (a U.S. Department of Transportation National University Transportation Center) headquartered at Clemson University, Clemson, South Carolina, USA. Any opinions, findings, conclusions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of TraCR, and the U.S. Government assumes no liability for the contents or use thereof. This research was conducted by University of Texas at Dallas (UTD) as a joint research project of Korea Institute of Science and Technology Information (KISTI)’s research and development project (Development of a zero-hacking system based on AI white-hat hackers) (K26L8M3C1, C26022).

VIII. ETHICAL CONSIDERATIONS

In this work, we collect and analyze system provenance data from active user environments, which inherently involves human subjects research. Furthermore, we execute live cyberattacks against these environments and make the resulting dataset available to researchers upon request. We carefully designed our study with awareness of the privacy and security implications of this methodology.

Our data collection, a two-week dataset involving active users and live attacks, was reviewed and approved by our Institutional Review Board (IRB) as human subjects research.

Stakeholder Analysis

We identify the key stakeholders affected by our study, the potential impacts on them, and the specific steps we took to mitigate any associated risks.

Research Participants (The “Users”):

Impact: As the individuals using the instrumented workstations, participants faced potential privacy risks from the collection and sharing of their system metadata, as well as potential disruption and data compromise from live cyberattacks.

Mitigation: To ensure informed consent and technical comprehension of the risks, the participants were deeply involved in the research process. Risks to privacy were mitigated by provisioning fresh workstations, strictly logging only metadata (no file or network payload contents), instructing participants to avoid personal tasks, and applying automated keyword sanitization prior to sharing the dataset. Attack disruptions were minimized by using non-destructive malware, and exfiltrated user data was deleted promptly after attack verification.

The Host Institution and IT Infrastructure:

Impact: Running vulnerable software and executing live malware poses a theoretical risk of lateral movement or infection spreading to the broader university network.

Mitigation: We strictly isolated the entire experimental testbed (workstations, shared servers, and routers) from the broader network using a VPN. All attack traffic and exfiltrated data were completely contained within this isolated environment. Furthermore, the research was conducted under the direct review and approval of the Institutional Review Board (IRB).

The Academic and Industry Security Communities:

Impact: This stakeholder group is the primary beneficiary of the research. Currently, researchers and developers are hampered by artificial, short-lived datasets that cause PIDSs to overfit and fail in real-world deployments.

Mitigation/Action: To maximize the positive impact for this group, we will make the sanitized dataset available to researchers upon request. This provides the community with the long-running, behaviorally diverse ground truth required to advance the state-of-the-art in intrusion detection.

General Public (Future System Users):

Impact: The general public benefits indirectly. By unblocking the stagnation in PIDS research, this work accelerates the development of robust security tools that will eventually be deployed to protect real-world users and enterprise environments from Advanced Persistent Threats (APTs). *Mitigation/Action:* No direct risks are posed to the general public, but their ultimate benefit relies on the responsible dissemination of DOLLHOUSE and findings to defensive practitioners.

Adversaries and Malicious Actors:

Impact: As with many defensive security datasets, there is a minor dual-use risk. Adversaries could theoretically analyze the dataset to study how PIDSs log specific attacks or to understand the statistical noise of benign user behavior, potentially using this knowledge to craft stealthier, evasion-optimized malware.

Mitigation: The attacks executed in DOLLHOUSE rely on well-known vulnerabilities rather than novel zero-day exploits. The defensive value of exposing current PIDS blind spots and providing a realistic benchmark for the defense community vastly outweighs the marginal intelligence an adversary might gain.

Respect for Persons

We respect participant autonomy and privacy through informed consent and data minimization.

Informed Consent and Privacy: To ensure maximum transparency and completely eliminate the risk of uninformed consent regarding data sharing, the five human participants operating the instrumented workstations were involved in the research throughout the process. As such, they possessed a complete technical understanding of the risks, including the internal details of the data collection process and the attacks executed against their machines.

Data Minimization and Anonymization: System provenance data can inadvertently capture sensitive behavior. To mitigate this, we implemented several strict technical and procedural boundaries:

Metadata Only: Our collection infrastructure only captures system metadata (e.g., filenames, program names, IPs, ports, timestamps). We explicitly do not capture or log the contents of files or the payloads of network communications.

Fresh Provisioning: We provisioned entirely new workstations specifically for this two-week experiment. This prevented the accidental capture of historical files or legacy data that might contain sensitive personal information.

Behavioral Scoping: Participants were explicitly instructed to avoid performing activities that could be considered highly personal or sensitive (e.g., financial transactions, medical consultations) on the instrumented machines during the collection period.

Keyword Sanitization: Prior to sharing DOLLHOUSE, participants provided a broad range of personal keywords (e.g., real names, usernames). These were used to automatically filter and anonymize any potentially sensitive strings that appeared in the captured filenames or process executions.

Beneficence

We carefully weighed the benefits of this research against the potential harms of live cyberattacks and data sharing.

Benefits: Current Provenance-based Intrusion Detection Systems (PIDSs) are stagnating due to an over-reliance on artificial datasets. By capturing real-world behavioral diversity alongside long-running attacks, DOLLHOUSE provides the security community with the necessary ground truth to develop practical, deployable defense mechanisms that can withstand real-world use.

Harms and Mitigation (Live Attacks): Executing live attacks on user workstations carries the risk of data loss and credential compromise. We mitigated these risks through strict containment. All custom malware was non-destructive. To prevent external leakage, the entire instrumented network—including the workstations, shared Linux servers, mail server, and routers—was strictly isolated from the broader university network via a VPN. Exfiltrated data never left the isolated lab network. Once the success of an attack was verified, all exfiltrated data was promptly deleted from the attack machines, and this deletion was personally verified by the participant whose machine was targeted.

Harms and Mitigation (Data Sharing): Sharing user provenance data poses a theoretical de-anonymization risk. However, because DOLLHOUSE is strictly limited to metadata,

collected on freshly provisioned machines, explicitly filtered for personal keywords, and devoid of deeply personal activities, we conclude that the privacy risks are thoroughly mitigated. Therefore, we determined that sharing DOLLHOUSE with researchers upon request is both ethically sound and highly beneficial to the community.

Respect for Law and Public Interest

The research was conducted in full compliance with institutional policies. The use of a VPN to isolate the testbed ensured that the vulnerable software required for our experiment—and the live attacks executed against them—posed zero risk to our university’s broader IT infrastructure, thus respecting public interest and institutional security policies.

Decision

We conclude that the significant benefits of providing a realistic, attack-laden provenance dataset to the security community heavily outweigh the mitigated risks. By utilizing researchers as participants, strictly isolating the network, and aggressively sanitizing the collected metadata, we have ensured that this research is conducted ethically and safely.

REFERENCES

- [1] Samuel T King and Peter M Chen. “Backtracking intrusions”. In: *Proceedings of the nineteenth ACM symposium on Operating systems principles*. 2003.
- [2] Samuel T King et al. “Enriching intrusion alerts through multi-host causality.” In: *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2005.
- [3] Yushan Liu et al. “Towards a Timely Causality Analysis for Enterprise Security”. In: *Network and Distributed System Security Symposium (NDSS)*. Network and Distributed System Security Symposium (NDSS). Feb. 2018. ISBN: 1-891562-49-5.
- [4] S. Forrest et al. “A sense of self for Unix processes”. In: *Proceedings 1996 IEEE Symposium on Security and Privacy*. 1996, pp. 120–128. ISBN: 9780818674172. DOI: [10.1109/secpri.1996.502675](https://doi.org/10.1109/secpri.1996.502675).
- [5] David Wagner and Paolo Soto. “Mimicry attacks on host-based intrusion detection systems”. In: *ACM conference on Computer and Communications Security (CCS)*. Nov. 2002, pp. 255–264. ISBN: 9781581136129. DOI: [10.1145/586110.586145](https://doi.org/10.1145/586110.586145).
- [6] Sadegh Momeni Milajerdi et al. “HOLMES - Real-Time APT Detection through Correlation of Suspicious Information Flows”. In: *IEEE Symposium on Security and Privacy (SP)*. 2019.
- [7] Sadegh M. Milajerdi et al. “POIROT: Aligning Attack Behavior with Kernel Audit Records for Cyber Threat Hunting”. In: *ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY (CCS)*. New York, NY, USA: Association for Computing Machinery, Sept. 2019, pp. 1795–1812. ISBN: 9781450367479. DOI: [10.1145/3319535.3363217](https://doi.org/10.1145/3319535.3363217).

- [8] Md Nahid Hossain, Sanaz Sheikhi, and R. Sekar. “Combating Dependence Explosion in Forensic Analysis Using Alternative Tag Propagation Semantics”. In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 1139–1155. DOI: [10.1109/SP40000.2020.00064](https://doi.org/10.1109/SP40000.2020.00064).
- [9] Michael Zipperle et al. “Provenance-Based Intrusion Detection Systems: A Survey”. In: *ACM Computing Survey* (June 2022). ISSN: 0360-0300. DOI: [10.1145/3539605](https://doi.org/10.1145/3539605). URL: <https://doi.org/10.1145/3539605>.
- [10] *SolarWinds Cyberattack Demands Significant Federal and Private-Sector Response (infographic) — U.S. GAO*. [Online; accessed 2026-04-16]. 2026. URL: <https://tinyurl.com/4bvk9dp9>.
- [11] *North Korea State-Sponsored Cyber Threat: Advisories — CISA*. [Online; accessed 2026-05-07]. URL: <https://www.cisa.gov/topics/cyber-threats-and-advisories/nation-state-cyber-actors/north-korea/publications>.
- [12] *WildPressure targets industrial in the Middle East*. <https://tinyurl.com/mr2n8hdu>.
- [13] *U.S. Said to Find North Korea Ordered Cyberattack on Sony*. <https://tinyurl.com/5da2h9bx>.
- [14] Anthony M Freed. *Inside the darkside ransomware attack on colonial pipeline*. 2021. URL: <https://tinyurl.com/y9puzen7>.
- [15] Wajih Ul Hassan et al. “NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage.” In: *Network and Distributed System Security Symposium (NDSS)*. Network and Distributed System Security Symposium (NDSS). Feb. 2019.
- [16] Qi Wang et al. “You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis”. In: *Network and Distributed System Security Symposium (NDSS)*. Feb. 2020.
- [17] Md Nahid Hossain et al. “SLEUTH: Real-time Attack Scenario Reconstruction from COTS Audit Data”. In: *USENIX Security Symposium (SEC)*. Vancouver, BC: USENIX Association, 2018, pp. 487–504. ISBN: 978-1-931971-40-9. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/hossain>.
- [18] Zijun Cheng et al. “Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance”. In: *IEEE Symposium on Security and Privacy (SP)*. Jan. 2024, pp. 3533–3551. DOI: [10.1109/sp54263.2024.00005](https://doi.org/10.1109/sp54263.2024.00005).
- [19] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. “Flash: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning”. In: vol. 00. *IEEE Symposium on Security and Privacy (SP)*. Institute of Electrical and Electronics Engineers (IEEE), May 2024, pp. 3552–3570. DOI: [10.1109/sp54263.2024.00139](https://doi.org/10.1109/sp54263.2024.00139).
- [20] Fan Yang et al. “PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding”. In: *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 4355–4372. ISBN: 978-1-939133-37-3. URL: <https://www.usenix.org/conference/usenixsecurity23/presentation/yang-fan>.
- [21] Akul Goyal, Gang Wang, and Adam Bates. “R-CAID: Embedding Root Cause Analysis within Provenance-based Intrusion Detection”. In: vol. 00. *IEEE Symposium on Security and Privacy (SP)*. Institute of Electrical and Electronics Engineers (IEEE), May 2024, pp. 3515–3532. DOI: [10.1109/sp54263.2024.00253](https://doi.org/10.1109/sp54263.2024.00253).
- [22] Zian Jia et al. “MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning”. In: *USENIX Security Symposium (SEC)*. Philadelphia, PA: USENIX Association, 20 2024, pp. 5197–5214. ISBN: 978-1-939133-44-1. URL: <https://www.usenix.org/conference/usenixsecurity24/presentation/jia-zian>.
- [23] DARPA I2O. *Transparent computing engagement 5 data release*. 2020. URL: <https://github.com/darpa-i2o/Transparent-Computing>.
- [24] FiveDirections. *Operationally Transparent Cyber (OpTC) Data Release*. 2020. URL: <https://github.com/FiveDirections/OpTC-data>.
- [25] Zhenyuan Li et al. “Incorporating Gradients to Rules: towards Online, Adaptive Provenance-based Intrusion Detection”. In: *IEEE Transactions on Dependable and Secure Computing* (2025).
- [26] Su Wang et al. “THREATTRACE: Detecting and Tracing Host-Based Threats in Node Level Through Provenance Graph Learning”. In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 3972–3987. DOI: [10.1109/TIFS.2022.3208815](https://doi.org/10.1109/TIFS.2022.3208815).
- [27] Zijun Cheng et al. “Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance”. In: *IEEE Symposium on Security and Privacy (SP)*. 2024.
- [28] Baoxiang Jiang et al. “ORTHRUS: achieving high quality of attribution in provenance-based intrusion detection systems”. In: *USENIX Security Symposium (SEC)*. USA: USENIX Association, Aug. 2025. ISBN: 978-1-939133-52-6.
- [29] Tristan Bilot et al. “Sometimes simpler is better: a comprehensive analysis of state-of-the-art provenance-based intrusion detection systems”. In: *Proceedings of the 34th USENIX Conference on Security Symposium*. Seattle, WA, USA: USENIX Association, 2025. ISBN: 978-1-939133-52-6.
- [30] Jason Liu et al. “What We Talk About When We Talk About Logs: Understanding the Effects of Dataset Quality on Endpoint Threat Detection Research”. In: *IEEE Symposium on Security and Privacy (SP)*. Institute of Electrical and Electronics Engineers (IEEE), 2025, pp. 112–129. DOI: [10.1109/sp61157.2025.00112](https://doi.org/10.1109/sp61157.2025.00112).
- [31] Andy Riddle, Kim Westfall, and Adam Bates. *ATLASv2: ATLAS Attack Engagements, Version 2*. 2023. arXiv: [2401.01341](https://arxiv.org/abs/2401.01341) [cs.CR]. URL: <https://arxiv.org/abs/2401.01341>.
- [32] Jason Liu et al. “How to Effectively Trace Provenance on Windows: Endpoint Detection & Response Teleme-

- try”. In: *Workshop on Attack Provenance, Reasoning, and Investigation for Security in the Monitored Environment (PRISM)*. 2026.
- [33] Xueyuan Han et al. “SIGL: Securing Software Installations Through Deep Graph Learning”. In: *USENIX Security Symposium (SEC)*. USENIX Association, Aug. 2021, pp. 2345–2362. ISBN: 978-1-939133-24-3. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/han-xueyuan>.
- [34] Xueyuan Han et al. “UNICORN: Runtime Provenance-Based Detector for Advanced Persistent Threats”. In: *Network and Distributed System Security Symposium (NDSS)*. 2020. DOI: [10.14722/ndss.2020.24046](https://doi.org/10.14722/ndss.2020.24046).
- [35] John Griffith et al. *Scalable Transparency Architecture for Research Collaboration (STARC)-DARPA Transparent Computing (TC) Program*. Tech. rep.
- [36] *Transparent Computing Engagement 3*. <https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md>.
- [37] *Transparent Computing Engagement 5*. <https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md>.
- [38] *FIVE DIRECTIONS*. <http://www.fivedirections.com/>.
- [39] *DARPA TRACE Program*. <https://www.darpa.mil/program/trace>.
- [40] Emaad Manzoor, Sadegh M. Milajerdi, and Leman Akoglu. “Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. URL: <https://dl.acm.org/doi/10.1145/2939672.2939783>.
- [41] Jun Zengy et al. “Shadewatcher: Recommendation-guided cyber threat analysis using system audit records”. In: *IEEE Symposium on Security and Privacy (SP)*. 2022.
- [42] Fan Yang et al. “PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding”. In: *USENIX Security Symposium (SEC)*. 2023.
- [43] Feng Dong et al. “DISTDET: A Cost-Effective Distributed Cyber Threat Detection System”. In: *USENIX Security Symposium (SEC)*. 2023.
- [44] Shaofei Li et al. “NODLINK: An online system for fine-grained APT attack detection and investigation”. In: *Proceedings 2024 Network and Distributed System Security Symposium*. San Diego, CA, USA: Internet Society, 2024. DOI: [10.14722/ndss.2024.23204](https://doi.org/10.14722/ndss.2024.23204).
- [45] Zian Jia et al. “MAGIC: Detecting Advanced Persistent Threats via Masked Graph Representation Learning”. In: *USENIX Security Symposium (SEC)*. 2024.
- [46] Qizhi Cai et al. “Next-Generation Datasets for Provenance-Based Intrusion Detection”. In: *Workshop on Attack Provenance, Reasoning, and Investigation for Security in the Monitored Environment (PRISM)*. USENIX Association, 2026. URL: <https://www.ndss-symposium.org/wp-content/uploads/prism2026-21.pdf>.
- [47] *The Linux audit framework*. <https://github.com/linux-audit/>.
- [48] *Event Tracing*. <https://docs.microsoft.com/en-us/windows/desktop/ETW/event-tracing-portal>.
- [49] *Metasploit — Penetration Testing Software, Pen Testing Security — Metasploit*. [Online; accessed 2026-05-07]. URL: <https://www.metasploit.com/>.
- [50] *Evaluation: Enterprise 2025*. <https://evals.mitre.org/enterprise/er7/>.
- [51] Tristan Bilot, Baoxiang Jiang, and Thomas Pasquier. *PIDSMaker: Building and Evaluating Provenance-based Intrusion Detection Systems*. 2026. arXiv: [2601.22983](https://arxiv.org/abs/2601.22983) [cs.CR]. URL: <https://arxiv.org/abs/2601.22983>.
- [52] Feng Dong et al. *Are we there yet? An Industrial Viewpoint on Provenance-based Endpoint Detection and Response Tools*. 2023. arXiv: [2307.08349](https://arxiv.org/abs/2307.08349) [cs.CR]. URL: <https://arxiv.org/abs/2307.08349>.
- [53] Steven A Hofmeyr, Stephanie Forrest, and Anil Somayaji. “Intrusion detection using sequences of system calls”. In: *Journal of computer security* 6.3 (1998), pp. 151–180.
- [54] Daniel Arp et al. “Dos and Don’ts of Machine Learning in Computer Security”. In: *USENIX Security Symposium (SEC)*. USENIX Association, Aug. 2021. DOI: [10.48550/arxiv.2010.09470](https://doi.org/10.48550/arxiv.2010.09470).
- [55] Kunal Mukherjee et al. “Evading Provenance-Based ML Detectors with Adversarial System Actions”. In: *USENIX Security Symposium (SEC)*. 2023.

APPENDIX

False Positive Rules

Table VIII lists the regular expressions used to group benign nodes into the categories in Table VI. We chose these categories by manually reviewing the benign nodes with the highest anomaly scores from each PIDS.

Table VIII: Rules used to categorize benign nodes in Table VI. Each rule is a case-insensitive regular expression matched against the node label (process command line, file path, or netflow src:port->dst:port).

Category	Rule	Regular expression
Netflow	Loopback	<code>127\.\0\.\0\.1:\d+-->127\.\0\.\0\.1:\d+</code>
	Git	<code>~*(c:[\//]program files[\//]git[\//]cmd[\//])?git[git(\.exe)?]*\$</code>
Subject	Updater	<code>unattended-upgrade apt apt-get snap snapd update-notifier dracut-install fwupdmg update upgrade install</code>
	Latex	<code>latex pdflatex xelatex lualatex</code>
File	Log	<code>log</code>
	Cache	<code>cache</code>
	JetBrains Rider	<code>rider</code>
	Latex	<code>latex pdflatex xelatex lualatex</code>
	Scheduled Task	<code>windows[\//]system32[\//]tasks</code>

Table IX: Successful attacks in DOLLHOUSE dataset in chronological order.

Attack ID	Attack	Mechanism	Target	User	Long Running	Protocol
1	File Stealer	Malware	provatk-ws3	efive	No	FTP
2	File Stealer	Malware	provatk-ws6	efour	No	HTTP
3	Credential Stealer	Malware	provatk-ws3	efive	No	FTP
4	Credential Stealer	Malware	provatk-ws6	efour	No	HTTP
5	File Snooper and Stealer	Metasploit reverse shell	provatk-ws3	efive	Yes	TCP
6	Document Stealer	Malware	provatk-ws3	efive	No	HTTP/FTP
7	SSH Pivot Snooper	Metasploit reverse shell	provatk-ws6	efour	No	TCP
8	File and Credential Stealer	C2	provatk-ws3	efive	No	HTTP
9	Credential Stealer	C2	provatk-ws4	eone	No	HTTP
10	Credential Stealer	C2	provatk-ws3	efive	No	HTTP
11	File Stealer and Credential Stealer	Malware	provatk-ws1	etwo	No	HTTP/FTP
12	Propagator and Credential Stealer	C2	provatk-ws4	eone	No	HTTP
13	Propagator and Credential Stealer	C2	provatk-ws3	efive	No	HTTP
14	Propagator and Credential Stealer	C2	provatk-ws4	eone	Yes	HTTP
15	Propagator and Credential Stealer	C2	provatk-ws3	efive	Yes	HTTP
16	Credential Stealer	Malware	provatk-ws1	etwo	Yes	HTTP
17	File Stealer	Metasploit reverse shell	provatk-ws3	efive	No	TCP
18	Propagator and Credential Stealer	C2 Worm	provatk-ws6	efour	Yes	HTTP
19	File Stealer	Malware	provatk-ws3	efive	Yes	HTTP
20	File Stealer	Malware	provatk-ws6	efour	Yes	HTTP
21	File Snooper and Stealer	Metasploit reverse shell	provatk-ws1	etwo	No	TCP
22	File Stealer	Metasploit reverse shell	provatk-ws1	etwo	No	TCP

Table X: Number of benign nodes detected before any malicious nodes in each attack were detected ($FP_{[k]}^*$) for each attack in Table IX. For CARBANAKv2, which has a single attack, see FP_1^* in Table V.

Attack	VELOX	ORTHRUS	KAIIROS	MAGIC	R-CAID	FLASH	THREATTRACE
1	54k (1.28%)	30.7k (0.73%)	40k (0.95%)	202.9k (4.81%)	594.9k (17.00%)	220.2k (5.22%)	870.4k (20.65%)
2	47.7k (1.13%)	9.7k (0.23%)	48.9k (1.16%)	609.1k (14.45%)	1.5M (41.66%)	823.8k (19.55%)	842.1k (19.98%)
3	141.5k (3.36%)	12.7k (0.30%)	15.9k (0.38%)	252.2k (5.98%)	428k (12.23%)	285.3k (6.77%)	870.2k (20.65%)
4	47.7k (1.13%)	16.8k (0.40%)	48.9k (1.16%)	432.5k (10.26%)	1.5M (41.66%)	273.8k (6.50%)	842.1k (19.98%)
5	155.3k (3.68%)	12.7k (0.30%)	49.4k (1.17%)	886 (0.02%)	1.5M (41.66%)	281.3k (6.67%)	871.6k (20.68%)
6	155.3k (3.68%)	12.7k (0.30%)	15.9k (0.38%)	721.6k (17.12%)	690.6k (19.74%)	285.3k (6.77%)	2.1M (49.15%)
7	7.1k (0.17%)	130.4k (3.09%)	20.8k (0.49%)	636.5k (15.10%)	1.5M (41.66%)	51.3k (1.22%)	881.5k (20.92%)
8	14.7k (0.35%)	12.7k (0.30%)	32.7k (0.78%)	313.8k (7.45%)	5.9k (0.17%)	479.4k (11.37%)	872.5k (20.70%)
9	76.3k (1.81%)	16.8k (0.40%)	21.2k (0.50%)	436.1k (10.35%)	361.5k (10.33%)	103.3k (2.45%)	869.7k (20.64%)
10	57.2k (1.36%)	52.7k (1.25%)	55.2k (1.31%)	999k (23.70%)	1.5M (41.66%)	832.1k (19.74%)	1.1M (26.46%)
11	19.3k (0.46%)	11.1k (0.26%)	43k (1.02%)	420.5k (9.98%)	39.3k (1.12%)	128.2k (3.04%)	874.9k (20.76%)
12	155.3k (3.68%)	37.1k (0.88%)	36.9k (0.88%)	1.9M (44.83%)	412.7k (11.80%)	84.2k (2.00%)	2.3M (54.81%)
13	4.9k (0.12%)	52.7k (1.25%)	55.2k (1.31%)	252.2k (5.98%)	337.6k (9.65%)	5.1k (0.12%)	842.1k (19.98%)
14	117.2k (2.78%)	92.6k (2.20%)	88.2k (2.09%)	640.8k (15.20%)	456.4k (13.04%)	43.8k (1.04%)	842.1k (19.98%)
15	153.7k (3.65%)	489.8k (11.62%)	60.1k (1.43%)	683.1k (16.21%)	6.7k (0.19%)	30k (0.71%)	842.1k (19.98%)
16	19.3k (0.46%)	24.2k (0.57%)	41.5k (0.99%)	420.5k (9.98%)	260.1k (7.43%)	248.9k (5.91%)	874.9k (20.76%)
17	12.5k (0.30%)	1.6k (0.04%)	1.5k (0.04%)	215k (5.10%)	650.6k (18.59%)	327.5k (7.77%)	1M (24.00%)
18	135.2k (3.21%)	41k (0.97%)	49.8k (1.18%)	692k (16.42%)	419.8k (12.00%)	37.2k (0.88%)	842.1k (19.98%)
19	57.2k (1.36%)	44.5k (1.05%)	40k (0.95%)	215k (5.10%)	2.2M (61.96%)	410.6k (9.74%)	875.1k (20.76%)
20	47.7k (1.13%)	9.7k (0.23%)	49.8k (1.18%)	730.7k (17.34%)	1.5M (41.66%)	326.6k (7.75%)	842.1k (19.98%)
21	146.6k (3.48%)	35.3k (0.84%)	32.6k (0.77%)	1.2M (28.67%)	1.5M (41.66%)	8.7k (0.21%)	1.1M (26.46%)
22	8.7k (0.21%)	54k (1.28%)	59.6k (1.41%)	683.3k (16.21%)	1.5M (41.66%)	36.6k (0.87%)	962.8k (22.84%)
FP_1^*	4.9k (0.12%)	1.6k (0.04%)	1.5k (0.04%)	886 (0.02%)	5.9k (0.17%)	5.1k (0.12%)	842.1k (19.98%)
FP_{max}^*	155.3k (3.68%)	489.8k (11.62%)	88.2k (2.09%)	1.9M (44.83%)	2.2M (61.96%)	832.1k (19.74%)	2.3M (54.81%)