A Principled Approach for Detecting APTs in Massive Networks via Multi-Stage Causal Analytics

Jiaping Gui[†,*], Mingjie Nie[†], Jinyao Guo[†], Futai Zou[†,*], Mati Ur Rehman[‡], Wajih Ul Hassan[‡,*]
[†]Shanghai Jiao Tong University, [‡]University of Virginia
Email: {jgui, derek1998, 384606266, zoufutai}@sjtu.edu.cn, {wkw9be, hassan}@virginia.edu

***Abstract*—Detecting Advanced Persistent Threats (APTs) in large enterprise networks with conventional Network Intrusion Detection Systems (NIDS) is challenging due to the stealthy, multi-stage, and long-running nature of APTs. This paper introduces NETGUARDIAN, a novel NIDS utilizing a comprehensive methodology to correlate anomalies across APT stages. By merging real traffic with simulated APT scenarios, NETGUARDIAN creates a detailed training dataset for enhanced anomaly detection. NETGUARDIAN implements custom models for each APT stage, extracting specific traffic features, such as periodicity and failed connections, to identify anomalies. These anomalies are then correlated to reconstruct attack paths. Our system leverages these paths to assign threat scores based on interconnected anomalies matching known APT progression, effectively prioritizing suspicious paths. Evaluation on a large dataset of enterprise network traffic merged with simulated APTs along with the DARPA OpTC dataset shows that NETGUARDIAN detects various APT stages with high accuracy and low false positives, outperforming state-of-the-art (SOTA) NIDS.**

***Index Terms*—NIDS, APT attacks, causal analytics**

I. INTRODUCTION

Advanced Persistent Threats (APTs) stand at the forefront of cybersecurity challenges for enterprise infrastructures in the era of digital transformation, as evidenced by the cyber attack against Russian Scientific Computing Center [1] and the SolarWinds supply chain attack [2]. APTs navigate through four pivotal stages - Initial Intrusion, Establish Foothold, Lateral Movement, and Data Exfiltration - each leaving distinct footprints in network logs which serve as valuable indicators for timely detection and mitigation in enterprise environments.

To tackle APTs, Network Intrusion Detection Systems (NIDS) act as critical components within organizational cybersecurity infrastructures. SOTA NIDS continuously parse through network traffic logs, generating threat alerts upon identifying patterns and signatures indicative of APT attacks. Security analysts rely heavily on these alerts, scrutinizing them to ascertain their validity, identify true attacks, and subsequently initiate the necessary incident response and attack remediation processes.

However, despite their critical role, current NIDS [3], [4] face several significant limitations that hinder their effectiveness in battling the sophisticated strategies employed by APTs. First, relying mainly on signature-based detection, these systems find it difficult to identify APTs that consistently change their tactics and use unique malware. Additionally, the narrow focus of current NIDS on singular events or individual APT attack stages significantly undermines their detection efficacy. The architecture and operation of these systems lack the capability to consider the causal relationships and progression across the multiple stages of APTs, which is crucial for comprehensive threat identification and understanding. The failure to account for the interconnectedness of APT stages results in a fragmented view of ongoing threats, leading to a high volume of false alerts or "alert fatigue" [5]. This condition not only overwhelms security analysts but also leads to a slow response to genuine, often sophisticated, threats embedded within the multitude of false positives. This scenario underscores the need for a threat alert triage mechanism capable of prioritizing alerts based on their severity and potential impact to facilitate more effective and timely responses by analysts.

Moreover, the lack of holistic attack reconstruction capability in conventional NIDS compounds the challenges faced by security professionals. Without a mechanism that cohesively correlates information across different APT stages, these systems fall short of providing a comprehensive understanding of the threat landscape. This deficiency hinders accurate attack reconstruction and impairs the subsequent analysis.

Finally, lack of scalability emerges as a third significant limitation [3], [4]. With the exponential growth in network traffic and the complexity of modern network architectures, traditional NIDS struggle to efficiently and accurately process and analyze the voluminous data, inevitably impacting their performance and reliability in detecting and responding to threats in real-time. This scalability issue is further exacerbated during peak traffic periods, limiting the system's ability to provide timely and accurate threat detection and analysis.

Addressing these challenges, we design NETGUARDIAN, an effective NIDS designed for optimal performance in large-scale network environments. We introduce the notion of *Multi-stage Causal Analytics*, which enables the system to effectively correlate different APT stages from network logs and generate causal paths between those stages. We demonstrate that these causal paths enhance our system's detection accuracy by providing a more comprehensive view of the unfolding threat landscape. NETGUARDIAN consists of four modules, which we describe below.

First, NETGUARDIAN uses the Traffic Data Merging module for creating a realistic training environment. It merges authentic network flows with simulated APT scenarios, crafting

*Corresponding authors.
This work was supported by the Natural Science Foundation of Shanghai (23ZR1429600).

a dataset that reflects the complexity of genuine network activities while embedding simulated attack patterns. The merging ensures that the learning model is exposed to and learns from a traffic environment where normal and malicious patterns coexist, capturing the intricacies of genuine network activities while integrating crafted attack patterns. This comprehensive training dataset provides a realistic traffic pattern exposure, entwining normal activities with malicious behaviors. Then NETGUARDIAN leverages the Historical Information Extractor module to analyze historical network data and distinguish between routine and potentially threatening activities, establishing valuable behavior baselines. This analysis adds depth and context to real-time data, enhancing system accuracy.

During runtime, the APT Stage Detector module actively extracts features indicative of anomalies during the critical stages of APTs. This module is composed of four detectors, each optimized for identifying a specific stage of an APT. To build these detectors, we leverage a novel adaption of SOTA methods [6]–[9]. First, the Phishing-Email Detector leverages machine learning techniques trained on meticulously preprocessed data from reputable datasets. With a discerning feature set extracted from each email and a trained SVM model, it efficiently identifies phishing attempts, flagging the originating IPs of these malicious emails as initial compromise indicators. Next, the Command and Control (C2) Communication Detector relies on scrutinizing periodic connections and DNS resolutions. It precisely identifies potential C2 communications by analyzing timestamp intervals and data sizes within identified IP pairs, alongside differentiating the nuances of DNS resolutions between benign and C2 traffic. Subsequently, the Network-Scanning Detector identifies lateral movements within the network by focusing on both vertical and horizontal scanning behaviors. Through a detailed analysis of failed connections on various ports and calculated anomaly scores, it meticulously flags internal IPs involved in anomalous scanning activities, providing insights into potential lateral movements within the network. Lastly, the Data-Leakage Detector employs a multifaceted approach by calculating and analyzing factors related to the volume of outbound traffic, the number of connections to external networks, and domains accessed. With a dynamic anomaly scoring system, it efficiently flags nodes with unusual traffic patterns as potential data leakage points, warranting immediate attention.

Once the individual stage detection is done, NETGUARDIAN's APT Causal Path Reconstructor module creates causal paths encapsulating network activities and sequences of anomalies, offering a coherent view of APT progression. This causal path not only aids in the detection process but also facilitates alert triaging and expedites the investigative process by providing a clear overview of the attack's progression. After that, we score and rank causal paths generated in the previous step such that paths related to true APT attacks are ranked higher than false alarms.

We evaluated NETGUARDIAN, showcasing its advanced APT detection capabilities compared to SOTA NIDS ARGUS [10] and PIKACHU [4]. Our assessment utilized two datasets: anonymized campus traffic and DARPA OpTC [11]. In comparison with these SOTA NIDS, NETGUARDIAN exhibited significantly lower false positive and false negative rates, reinforcing its efficacy in real-world scenarios. Remarkably, our system has a data processing rate of 1.35× and over 170× faster than ARGUS and PIKACHU, respectively, when analyzing extensive network logs. A distinctive feature of NETGUARDIAN lies in its consistent high performmance across varying network sizes, which we demonstrated by expanding background traffic from one to eight days without affecting its performance. This resilience stems from our system's reliance on attack features and causal dependency for stage detection and path reconstruction, which remain effective regardless of the network's scale.

Moreover, NETGUARDIAN's ability to discern key stages of APT attacks offers a substantial benefit. Its stage detection models streamline the typically demanding process of stage identification, reducing the need for extensive domain expertise. Our system successfully identified all stages in various APT scenarios, as detailed in the case study of DARPA OpTC_0923 attack. This level of interpretability greatly assists security analysts in comprehending and responding to threats.

The main contributions of our paper are as follows.

- The development and validation of a novel data merging technique, enabling the embedding of simulated APT scenarios within genuine network data.
- The design and implementation of SOTA APT stage detection models, utilizing a harmonized approach of statistical, historical, and graph analytics.
- The introduction of a sophisticated attack causal path reconstruction algorithm, adeptly analyzing candidate sequences, and proficiently illuminating the interconnected progression of APTs.
- A robust evaluation of NETGUARDIAN, demonstrating its capability to reconstruct multi-stage cyber-attacks with precision within expansive enterprise networks.

II. BACKGROUND

In this section, we first discuss the limitations of Host Intrusion Detection Systems (HIDS) to underscore the necessity of NIDS solutions for detecting APT attacks. Then, we use a real-world APT attack scenario to discuss the usage of NIDS and the problems existing NIDS solutions face in practice.

A. Limitations of HIDS

HIDS monitors the process activities (e.g., file, process and network sockets) [12]–[14] in system logs to detect malicious activities on an individual host. However, HIDS faces significant practical limitations in a massive network, especially against APT attacks. We explain each of these limitations of HIDS in more detail below.

Graph construction: Netflow objects in system logs capture data at the level of system events corresponding to individual network packets. Directly analyzing packet-level data, as Netflow logs require, is resource-intensive and inefficient due to the voluminous amount of data generated, particularly

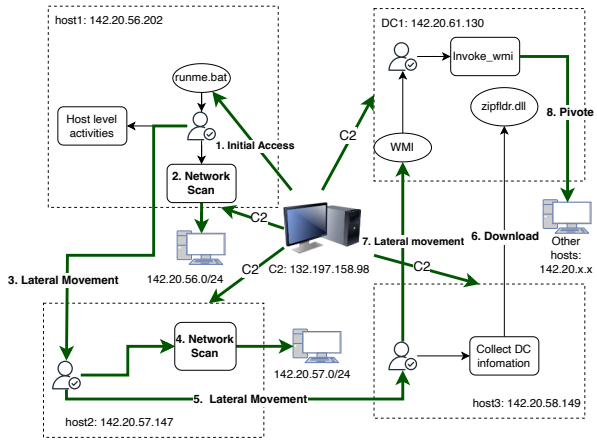


Fig. 1: Workflow of an APT attack in DARPA OpTC.

in large network environments (e.g., enterprise or campus). Constructing communication graphs across the entire network poses significant practical challenges for HIDS. In contrast, it is more feasible and efficient to create these graphs by monitoring traffic flows in the network.

Log monitoring: In a large-scale network comprising over 10,000 devices, instrumenting every device within such a network for data collection is not only impractical but also less reliable. Should data collection fail on any device, it would lead to a complete loss of traffic data from that device, adversely impacting the analysis and overall system performance. The maintenance cost is also high, since devices may be equipped with diverse operating systems and utilize varying log formats. In contrast, collecting network logs at the gateway circumvents these issues, offering a more viable and efficient solution for large-scale environments.

Resilience to zero-day attacks: HIDS solutions (e.g., RapSheet [15]) can only detect attacks that leave specific footprints in system logs. Consequently, reliant on known patterns within these logs, HIDS solutions are inherently limited to detecting known threats, rendering them less effective against zero-day attacks, which exploit unknown vulnerabilities and do not match existing patterns. In contrast, network logs offer a broader scope, capturing a wider range of interactions and traffic patterns, potentially including anomalous behaviors indicative of zero-day exploits. Nevertheless, zero-day attacks that evade HIDS solutions are unlikely or at least difficult to conceal their traces in network logs.

We argue that network logs contain sufficient attack traces to detect multi-stage APT attacks. However, existing NIDS solutions fail to achieve this due to various challenges that we will elaborate on below.

B. Challenges in Existing NIDS

We leverage the "Plain PowerShell Empire" attack that was launched in a real-world scenario in the DARPA OpTC dataset [11], as our motivating example. Figure 1 shows the attack process, comprising multiple stages. The adversary first connects to the victim device `host1` and downloads the malicious PowerShell Empire stager into a batch file. Then, he escalates access privilege, gathers credentials using Mimikatz, and edits the registry to establish persistence. Finally, the adversary conducts a network scan against the /24 network and utilizes WMI to pivot to `host2`, from where he pivots to `host3` using the same scanning method. On `host3`, the adversary attempts multiple system-level activities, collects information on the domain controller `DC1`, and pivots to `DC1` with WMI, and so on.

TABLE I
COMPARISON OF NETGUARDIAN AGAINST SOTA NIDS.

Systems	Single-stage detection	Multi-stage detection	Attack investigation	Massive networks
[16] [19] [24] [25] [26]	✓	×	×	×
[3] [9] [18] [27] [28]	✓	×	×	✓
[20] [21] [22] [23] [29]	✓	✓	×	×
[4] [10] [30]	✓	✓	×	✓
[31]	✓	×	✓	×
[32]	×	×	✓	×
NETGUARDIAN	✓	✓	✓	✓

* Generic anomaly detection is also considered as multi-stage detection.
* A network with over 10,000 hosts is considered a massive network.

However, it is challenging for existing NIDS to detect the aforementioned attack. Unlike HIDS whose system logs capture intricate details of system behaviors, NIDS suffer from network traffic encryption (e.g., TLS). Consequently, many network logs contain little information about encrypted payloads, making it harder for NIDS to detect multi-stage attacks. Despite the progress, a large number of NIDS [3], [16]–[19] focus on single-stage detection (e.g., lateral movement), which fails to provide a holistic view of the attack. Those that support multiple-stage detection [10], [20], [21] identify anomalous behaviors that do not directly map to specific stages, and these anomalies are isolated, leading to a fragmented view of ongoing threats. For example, in the attack scenario depicted in Figure 1, existing NIDS might detect anomalies on individual hosts, yet security analysts are left with limited insights into the attack's evolution throughout its duration. Additionally, the sheer scale of networks can be another barrier preventing the practical application of many NIDS [22], [23]. To better understand these limitations, we thoroughly investigate recently published NIDS solutions and summarize our findings in Table I.

III. PRELIMINARIES

In this paper, we consider network communication as a directed graph, with unique IPs as nodes[1] and flow sessions as edges, each with timestamps. Within this graph, a host or node can be either the source or the destination of a directed edge, depending on whether it is the source or destination IP in a flow. This approach allows for a more detailed and accurate representation of network activities, crucial for detecting subtleties in APT attacks. Formally, the communication graph can be represented as: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}_E)$, where $\mathcal{V}$ denotes the set of nodes whose type is determined by whether the IP is private, $\mathcal{E}$ denotes the set of edges for traffic flows, and $\mathcal{X}_E$ denotes the set of edge attributes (e.g., timestamps).

[1]In the rest of this paper, we will use IP, host, or node interchangeably.

A traffic flow is denoted as a directed edge $e(u, v)$, where $u \in \mathcal{V}, v \in \mathcal{V}, e \in \mathcal{E}$, and the direction of the edge represents the direction of network connection (i.e., connecting from u to v). In addition, the edge records the start ($e.st$) and end ($e.et$) time of the flow. Given two nodes n_1 and n_2, n_2 has a casual dependency on n_1 if there exist two edges $e_1(n_1, u_1)$ and $e_2(u_2, n_2)$ such that $u_1 = u_2$ and $e_1.st \leq e_2.et$. For an APT attack, we define its inter-stage casual path as $\mathcal{L} = \{v_1, v_2, ..., v_n\}$, where $v_k \in \mathcal{V}, \forall k \in \{1, ..., n\}$, representing the node in a specific APT stage, and v_k has a causal dependency on its preceding nodes, i.e., $v_i, \forall i \in \{1, ..., k-1\}$.

IV. Threat Model & Assumptions

NetGuardian targets accurate identification of multi-stage APT attacks and aims to provide exhaustive threat vector coverage while minimizing false positives, given the notable consequences of false detections in Security Operations Centers (SOCs) [5]. Tailored for organizations with constrained analyst resources, NetGuardian produces prioritized, actionable alerts and ranks potential attack paths by inherent risk, enabling refined alert management strategies. The system's success is gauged by its ability to alert any APT stage that leaves network footprints, prompting organizations to employ advanced forensic methodologies for further analysis [5], [33]. This work envisages a large enterprise environment under threat from well-funded, remote attackers executing swift, disruptive assaults. Assumptions include network log integrity, as per existing research [3], [34], with a network log collector facilitating a transparent view of enterprise-wide network activities. This paper does not consider attacks via hardware layer, physical vectors, or side channels. While the secure storage of network logs and accessibility only to authorized users are crucial for user privacy, the enhancement of privacy-preserving mechanisms in our system is left for future work.

V. Design

Figure 2 presents the architecture of NetGuardian, which consists of four modules: Traffic Data Merging, Historical Behavior Analysis, Stage Detector, and APT Causal Path Reconstructor. The following subsections delve into the design and operational details of these modules.

A. Traffic Data Merging

To simulate APT attacks in large networks (e.g., campus), we need both large-scale background traffic ($\mathcal{B}_\mathcal{T}$) and attack traces ($\mathcal{A}_\mathcal{T}$) from the real world. Background traffic is collected by deploying a monitor at the network gateway, which captures all external and most internal communications. This method enables the profiling of internal nodes' historical traffic patterns, assisting in aligning nodes with attack traces. We simulate APT attacks in a lab environment, following typical settings in related works [35], [36], using the MITRE adversary emulation library [37]. By manually launching each APT stage, we control the attack progression, which ensures the duration and time sequence captured are stable for each APT stage. This simulation yields labeled attack traces, serving as

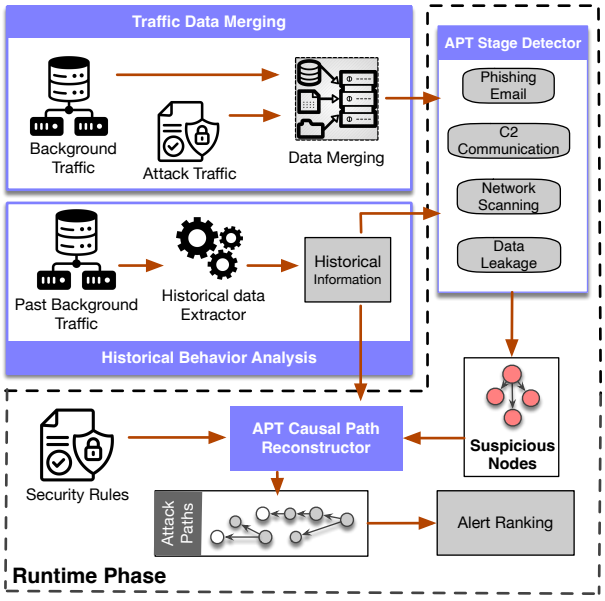


Fig. 2: Overall NetGuardian architecture.

ground truth to evaluate NetGuardian. For each APT attack stage, we record the time and methodology. Specifically, we log the start time of each stage to create a timestamp sequence, $T = t_1, t_2, ..., t_n$. Here, n is the stage count and t_i is the start time of the i-th stage in $\mathcal{A}_\mathcal{T}$. We then extract t_{max} and t_{min} from $\mathcal{B}_\mathcal{T}$. Data merging for the i-th attack stage between t_i and t_{i+1} follows specific rules for inserting attack flows into the background traffic.

The key challenge behind data merging is to retain the behavior of each attack stage while ensuring rational mapping during data merging. To handle it, we design a merging process that involves adjusting attack timestamps to fit into the background traffic timeline and maintaining the stealth and persistence of APT behaviors. Specifically, the timeline of $\mathcal{A}_\mathcal{T}$ is proportionally mapped to that of $\mathcal{B}_\mathcal{T}$. For instance, if a network flow in the i-th stage starts at t_h, the updated timestamp post-merging is $t'_h = \frac{t_{max} - t_{min}}{t_n - t_1} * (t_h - t_1) + t_{min}$. IP mapping considers both source and destination IPs. This process requires no change in the behavior of simulated attack stages. IPs are selected based on the attack methodology in each stage. For flows in stages, such as initial compromise or data leakage, one IP is external. In stages, such as lateral movement, both IPs are internal. After determining the IP type and the updated timestamp t'_h, we select IPs in $\mathcal{B}_\mathcal{T}$ active near t'_h as the mapped IPs. This approach assumes that it is unreasonable to map a flow in $\mathcal{A}_\mathcal{T}$ to nodes that do not communicate in $\mathcal{B}_\mathcal{T}$.

B. Historical Behavior Analysis

This module analyzes nodes' historical behavior, providing a basis for anomaly detection in subsequent phases. It records daily large-scale background traffic, extracting crucial node statistical information for profiling, and is divided into Connection Feature Extraction and Node Clustering components.

Connection Feature Extraction. Malicious (infected) internal nodes often show abnormal network behavior, manifesting as

increased one-way connections or significant upstream and downstream traffic disparities. We extract: (1) data volume transmitted in both directions, (2) the total number of domains accessed, (3) overall connections established, (4) the count of one-way connections (where requests are rejected or ignored), and (5) connections displaying significant traffic disparities (with a ratio over 20 between data volumes) for each node. This statistical information allows us to detect anomalies based on the behavioral differences exhibited by nodes over time.

Node Clustering. Besides statistical traffic features, we cluster internal nodes to identify abnormal behaviors. We do this based on the observation that in a large network, the normal behavior of most internal nodes is similar (except for special nodes such as servers), while a minority of nodes fall into a tiny cluster exhibiting abnormal behavior. To cluster nodes, we first use the statistics extracted in the prior component as features and normalize these features using min-max normalization. Then, we leverage spectral clustering algorithm [38] to cluster all internal nodes in the daily network traffic. After clustering, a main class that consists of the majority of nodes is formed, with a smaller class representing the minority. We consider the nodes in the smaller class to have a higher probability of exhibiting abnormal behavior on the same day.

C. APT Stage Detector

The next step is detecting the stages of APT attacks. After reviewing a number of APT reports [39]–[41], we summarize network-related behaviors that correspond to each of the stages in the lifecycle of APT attacks. In the end, we consider the four most representative stages whose communication activities can be captured through network monitoring. These stages are 1) phishing emails, 2) connection to C2 servers, 3) network scanning during lateral movement, and 4) data leakage. Our key insight is that by identifying individual stages and causally linking them together to reconstruct the whole attack path, we can detect APT attacks with high accuracy. Notice that it is possible that an APT attack has a different attack vector, but it does not invalidate our methodology. Instead, our system has a flexible architecture that supports the detection of new attack vectors. In Section VI, we demonstrate NETGUARDIAN's capabilities against various APTs that span across all or a subset of four stages. Below we explain the feature extraction concerning each of these steps.

1) Phishing-Email Detector: According to reports [42], many APT groups employ phishing emails to achieve initial compromise. A detected phishing email in the network is a strong signal indicating an adversary launches a potential APT attack. To detect phishing emails, we train a detection model through machine learning techniques. To do it, we first download the normal and phishing-email datasets from Enron email [43] and Phishing Corpus [44] dataset, respectively. We leverage various features in our detectors, such as the number of URLs included, the frequency of keywords (e.g., login or upgrade), and whether a reference/script is embedded. Taking into account these features, we extract a 12-dimension vector for each email, and normalize the numerical features using min-max normalization. Finally, we train an SVM model utilizing the normalized features from half of the data in both datasets. Testing the other half shows that our trained model achieves an accuracy of 93%. For emails in real network traffic, we extract the same set of features and use the trained model to determine if they are phishing emails. Upon detection of a phishing email, the corresponding sender's IP is flagged, signifying a potential initial compromise.

2) C2-Communication Detector: C2-communication detection hinges on two observable features: the presence of periodic connections between the C2 server and the infected host, and the characteristics of DNS resolutions for C2 domains. For the periodicity detection, we scrutinize the timestamp intervals and data sizes of communication flows within identified IP pairs. The coefficient of variation for these sequences is meticulously computed, and IP pairs that exhibit coefficients surpassing predetermined threshold values are flagged as potential C2 communications. In the realm of DNS context detection, the approach discerns the nuanced differences between the DNS resolutions associated with C2 traffic and regular, benign traffic. Typically, C2 traffic exhibits isolated DNS resolutions with no dependent relationships in their context. The methodology involves recording the frequency of specific domain resolutions by internal IPs, subsequently analyzing the surrounding DNS context for each resolution instance. The detection algorithm calculates and utilizes two metrics, namely the average number of other domains resolved concurrently and the maximum count of subsets containing a specific domain, to effectively discriminate between C2 and normal traffic.

3) Network-Scanning Detector: During the lateral movement phase of APT attacks, adversaries engage in network scanning, which is indicative of potential lateral movements. Scanning behaviors can be categorized as vertical, where a single internal IP is scanned at multiple ports, and horizontal, involving multiple IPs scanned at a single port. In a **Vertical Scan**, ports are classified as sensitive (e.g., 80, 443, 3389) and non-sensitive, with the number of failed connections at each being recorded as s_1 and s_2, respectively. An anomaly score is calculated using $SC_1 = \omega_1 \cdot s_1 + \omega_2 \cdot s_2$, with ω_1 and ω_2 as weights assigned to sensitive and non-sensitive ports [6]. For **Horizontal Scans**, we analyze connections initiated by an internal host to others, using two sequences representing the number of connections and the volume of data transmitted. This leads to a normalized entropy-based anomaly score SC_2. If either SC_1 or SC_2 surpasses certain thresholds, the internal IP is flagged for scanning behavior.

4) Data-Leakage Detector: In detecting data leakage, the Data-Leakage Detector employs three calculated factors derived daily for each internal node [9]. Each node h has an associated feature vector $\mathbf{x}_t = (x_t^1, x_t^2, x_t^3)$, encapsulating the total upstream traffic to external networks, total connections to external networks, and the number of accessed domains. The first factor, f_t^1, is the Euclidean distance between $\mathbf{x}_t$ and $\mathbf{cv}_t$, the center vector of the feature space for all nodes, providing a measure of deviation from the norm. The second factor, f_t^2,

represents the degree of deviation of $\mathbf{x}_t$ from its historical values, offering insight into unusual activity over time. The third factor, f_t^3, evaluates the direction of the deviation in the feature space, helping identify anomalous trends in the traffic patterns of all internal nodes. The final anomaly score for a node is computed as the product $f_t = f_t^1 \times f_t^2 \times f_t^3$. Nodes with an anomaly score exceeding a predetermined threshold are flagged as potential data leakage points, ensuring immediate attention and action.

D. APT Causal Path Reconstructor

This module aims to reconstruct APT progression paths, utilizing threat scores to identify and prioritize the most suspicious paths indicative of APT attacks.

Inter-Stage Behavioral Analysis. Different from the stage detector (Section V-C) focusing on detection models, we first aim to identify anomalous behaviors that span across stages. It helps facilitate the reconstruction of APT attack chain in this module based on the different impacts of the same behavior on different stages. For example, malicious payload downloading could be launched in the initial compromise or lateral movement stage. Intuitively, this downloading behavior has a higher threat impact on the earlier stage (i.e., initial compromise). For another example, a mining domain accessed by a host indicates the host is a potential candidate for foothold establishment. In a real APT attack, this host could later launch the lateral movement activity in the subsequent step. To identify inter-stage behaviors related to network traffic, we reviewed a number of APT reports [39]–[41] and summarize all inter-stage behaviors that are applied to NETGUARDIAN.

Threat Scores. For each of the behaviors summarized, we define a threat score that reflects the degree of correlation between the behavior and the attack stage. Specifically, for each internal node h, we first sum the threat scores as the stage score for all four stages. Then, we map the node to a stage if the node's stage score is over the threshold in that stage for each of the stages. That said, the node h is more likely to perform abnormal behavior in one or more stages compared to other stages. Finally, we obtain a set of nodes for each of the stages, labeled as U_1, U_2, U_3, and U_4.

Next, we reconstruct attack paths by correlating suspicious nodes in the $U_1 - U_4$ sets. To do it, we first convert traffic flows into a directed graph, where each node represents an individual IP, and each edge the flow between two nodes. The direction of edges is consistent with that of flows. The edge contains information such as the start time and ports of the corresponding flow. If there is a directed path from node h_1 to node h_2, and each edge on the path satisfies a temporal dependency relationship (i.e., increasing timestamps), then h_2 is considered reachable for h_1.

Accordingly, we can reconstruct the attack path based on the reachability relationship between nodes. To do it, we perform temporal depth first search (T-DFS) [45] with each node in $U_1 - U_4$ as the root node. The time and space complexity of the T-DFS algorithm are $O(|\mathcal{E}| + |\mathcal{V}|)$ and $O(|\mathcal{V}|)$, respectively. Unlike standard DFS, T-DFS considers whether the timestamp of the next edge and the current edge satisfy temporal dependencies when performing searches. In addition, standard DFS does not consider the presence of multiple edges, which are common in our input data (e.g., h_1 communicates with h_2 multiple times). To address this problem, we use the earliest timestamp when h_1 connects to h_2 to represent all communications from h_1 to h_2 during search. In this way, we ensure all reachable paths can be included in the searching results.

After performing T-DFS, we obtain a set of reachable nodes for each node. Then, we take each node in $U_1 - U_4$ as the root node, recursively traverse its set of reachable nodes, and sequentially add the traversed nodes to form possible paths. Finally, we calculate the threat score for each of all paths as follows. For a path $H = \{h_1, h_2, ..., h_m\}$, where m is the number of nodes in the path, and h_i $(i \in [1, m])$ is the i-th node. We determine the threat score of path H as $SC = \frac{\sum TS(h_i)}{m} + 0.1 * length^{1.1}$, where $TS(h_i)$ is the threat score of node h_i, and $length$ the path length. If H's sub-paths have a higher score than H, we update H's score to the average of the highest score among sub-paths and H's score. We rank all paths based on their threat scores as the final output of the detection model.

VI. EVALUATION

We answer the following research questions (**RQs**) in our evaluation: **RQ1:** How does our system's APT detection accuracy compare to existing methods on real enterprise data? **RQ2:** Can our system maintain detection precision as network size increases? **RQ3:** How effectively does our model differentiate between APT attack stages? **RQ4:** What is the system's efficiency in terms of computational and memory resources? NETGUARDIAN is executed on a server with Intel(R) Xeon(R) CPU E5-2630 v4 (2.20GHz), 188GB RAM running 64bit CentOS Linux release 7.9.2009 (Core).

A. Baseline Comparison

In our comparative analysis, we selected the most recent works ARGUS [10] and PIKACHU [4] as baselines due to their comprehensive nature in APT detection. Contrasting with systems like Euler [3], which are tailored to specific APT stages (such as lateral movement), ARGUS and PIKACHU encompass the entire spectrum of attack stages, making them an appropriate benchmark for NETGUARDIAN's multi-stage detection capability. This focused approach to comparison allows for a more equitable and insightful evaluation of NETGUARDIAN in the context of holistic APT detection systems. It is important to note that neither ARGUS nor other related works support the reconstruction of causal paths in APT scenarios. We do not evaluate against system log-based provenance analysis approaches, such as ProGraPher [46] and RapSheet [15], since they are HIDS solutions, a different scope from ours, as outlined in Section II.

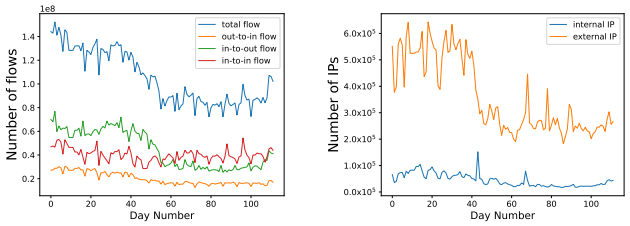
B. Evaluation Dataset

Two datasets were employed for our evaluation: the Anonymous Campus Network Traffic and DARPA OpTC [11]. Note

TABLE II
STATISTICS OF TRAFFIC DATASETS AND NETGUARDIAN'S PERFORMANCE RESULTS.

Attack	Statistics of traffic datasets							Performance of different phases of NETGUARDIAN							
	Historical traffic (daily average)		Background traffic		Attack traffic			Historical Information Extractor		Data Merging		Stage Detector		Attack Path Reconstructor	
	# flows (M)	# int./ext. IPs	# flows (M)	# int./ext. IPs	# flows	# int./ext. IPs	# stages	Exec. Time (s)	Mem. (GB)	Exec. Time (s)	Mem. (GB)	Exec. Time (s)	Mem. (GB)	Exec. Time (s)	Mem. (GB)
menuPass	117.38	46K / 404K	104.53	28K / 25K	21	1 / 2	3	661	0.41	1,550	1.93	802	1.60	1,525	0.60
Sandworm	82.55	20K / 255K	82.69	29K / 228K	587	1 / 4	3	564	0.35	1,323	1.71	632	1.32	1,450	0.58
Wizard Spider	94.42	40K / 276K	89.01	27K / 226K	366	1 / 3	3	595	0.36	1,392	1.75	680	1.49	1,490	0.69
OpTC 0923	33*	316 / 993	38.35*	325 / 1,002	3,667*	1 / 3	3	273	1.49	3,290	2.73	602	3.70	2,818	0.51
OpTC 0924	33*	316 / 993	34.47*	321 / 992	401*	2 / 3	4	273	1.49	3,579	2.85	614	11.34	16,154	0.87
OpTC 0925	33*	316 / 993	23.13*	315 / 1,025	563*	2 / 3	3	273	1.49	1,543	1.68	535	3.25	3,996	0.54

* In the table, "int./ext." denotes internal/external, "Exec." Execution, "Mem." Memory, and "*" flows only from Zeek sensors in the OpTC dataset.



(a) Flow statistics (b) IP statistics

Fig. 3: Statistics of background traffic for about four months from 05/01/2023 to 08/20/2023.

that we exclude the LANL 2015 dataset [47] because it discards most of the contextual information (e.g., filiation relationship) which impedes fine-grained feature engineering.

Campus Network Traffic. To investigate APT attacks in large networks, we monitored the gateway of an anonymous campus network for over 110 days using Zeek [48] to capture daily traffic flows. This traffic includes both internal and external communications, with noticeable variations in external traffic between the semester and the summer break. Figure 3 shows the number of flows (3a) and IPs (3b) that have traffic captured, respectively over the monitoring duration (i.e., 05/01/2023 - 08/20/2023). With an average of *∼1.05 billion* total traffic flows (over 2.38 TB of traffic data transmitted) and 45,078 active internal IPs daily, this data is representative of substantial networks. For evaluation, we select a random day's data for background traffic and analyze the preceding ten-day traffic for historical information extraction.

Simulated attacks capture the important traits of different stages in the lifecycle of APT attacks. Specifically, we performed three APT attacks [49]: `menuPass`, `Sandworm`, and `Wizard Spider` that represent complex scenarios in the real world. For `Sandworm` and `Wizard Spider`, we performed the failed scenarios where the malware was unresponsive and the lateral movement to the domain controller failed, respectively. Both scenarios could evade traditional solutions that rely on monitoring external data transfer to trigger alerts. Note that we do not include attacks such as *illegal storage* and *backdoor download* that are evaluated by related work [50], since these attacks are either executed only on the host with no network activities or have one-step simple logic to achieve the malicious goal. Nonetheless, our solution is complementary to existing ones that work on such attacks.

DARPA OpTC. The DARPA OpTC dataset [51], an ideal candidate for APT detection research [52], is also included in our study. This dataset comprises system and network audit logs from 1,000 Windows-10 hosts. We developed a tool to parse these logs, which have all internal traffic captured by host sensors directly included in system logs, not in the Zeek-captured network logs. To address this, FLOW objects in system logs were parsed and mapped to external traffic using IP addresses. Our evaluation utilizes the OpTC dataset, containing *∼0.53 billion* Zeek-sensor traffic flows and *∼8.76 billion* host-level sensor traffic events. It is crucial to note that NETGUARDIAN, in the OpTC dataset, relies on traffic events rather than flows for detecting anomalies like internal network scanning. The OpTC dataset encapsulates both benign and malicious activities, documented through network and host-level logs. Following an initial period of benign record generation, the dataset incorporates multi-stage APT attacks executed by a red team, occurring alongside continuous benign traffic. For our evaluation, we profiled nodes based on five days of benign traffic (09/18 - 09/22) and detected APT attacks within three days of malicious traffic (09/23 - 09/25).

Ground Truth Labeling for the Attacks. Simulated attacks on our hosts captured the start and end times of each stage and related traffic, excluding benign traffic during attack execution for accurate labeling. In the OpTC dataset, attack flows were identified and mapped using source/destination IPs/ports and timestamps, verified by detailed red-team documentation. We manually checked both datasets for labeled flow completeness.

Table II presents the traffic statistics for both merged and OpTC datasets, including flow numbers and internal/external IP counts for all traffic types, with daily average statistics provided for historic traffic.

RQ1: Detection of APT Attacks

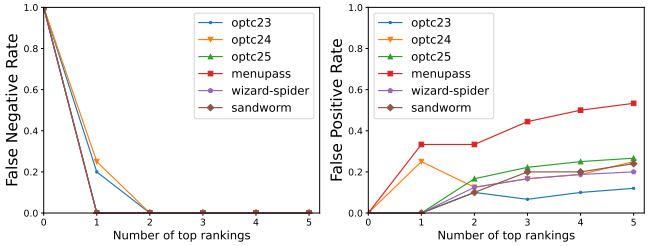
To demonstrate the effectiveness of NETGUARDIAN in detecting APT attacks, we investigate the ranking of the

TABLE III

THE RANKING OF GROUND-TRUTH ATTACK PATH IN THE FINAL RECONSTRUCTED PATHS.

Attack	menu Pass	Sand worm	Wizard Spider	OpTC_ 0923	OpTC_ 0924	OpTC_ 0925
Ranking	1	1	1	3	1	1

* Underlined numbers represent the ranking in juxtaposition.



(a) False negative rate (b) False positive rate

Fig. 4: Statistics of attack nodes that are misclassified or missing in top-ranked paths.

ground-truth attack path among those reconstructed by NETGUARDIAN. Table III lists the results. We can see that the ground-truth paths exhibit the highest threat scores for nearly all attacks, with three rankings in juxtaposition. The only exception is `OpTC_0923`, where the ground-truth path ranks third. The reason is that the top-one path is a subset of the ground-truth path, while the second-highest scoring path lacks a host for lateral movement that is present in the ground-truth path. The features of lateral movement on this absent host are not as salient as those of other nodes, primarily due to the lack of network scanning, which is verified by the red-team report. Nonetheless, the result of `OpTC_0923` is still promising for real-world attack investigations. For two failed attacks, NETGUARDIAN can also accurately identify the attack paths among the top rankings, with one achieving the highest score and the other ranking second.

We further measure the false positive rate (FPR) and the false negative rate (FNR) among the top-ranked paths reconstructed by NETGUARDIAN. The FPR and FNR are defined as the number of benign nodes that are misclassified as malicious, and the number of attack nodes that are missing over the number of all nodes in reconstructed paths. We show the results in Figure 4 where n on the x-axis represents all top n paths. Consistent with aforementioned results, no attack node is missed with a small number of nodes misidentified in the top two paths among all attack cases. Due to the limited number of nodes in each path, the number of misidentified nodes is small despite the relatively high FPR rate (i.e., over 20%) in several attacks. For example, in the top three paths, `menuPass` has the highest FPR (0.44), but the actual number of benign nodes (i.e., misidentified) is 4. This indicates NETGUARDIAN is highly effective in identifying all attack nodes while keeping a low number of false positives.

TABLE IV

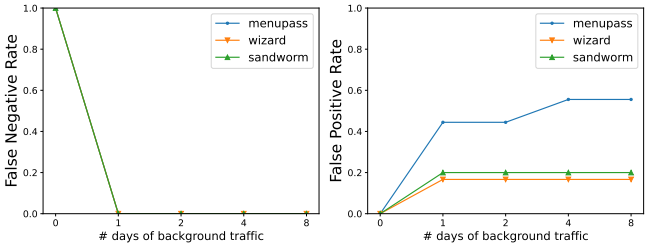
COMPARISON OF DETECTION PERFORMANCE.

Attack	FPR (%)			TPR (%)		
	PKC	AGS	NG	PKC	AGS	NG
menuPass	7.3	2.1	**1.45E-03**	92.7	90.5	**95.6**
Sandworm	7.5	2	**2.62E-03**	92.5	90.3	**96.1**
Wizard Spider	7.6	2.1	**2.03E-03**	93.0	90.1	**96.4**
OpTC_0923	4.2	0.05	**7.48E-03**	98.4	82.6	**98.6**
OpTC_0924	4.4	0.04	**5.57E-03**	98.2	82.7	**98.8**
OpTC_0925	4.2	0.06	**2.28E-03**	**98.5**	82.1	98.4
Average	5.9	1.1	**3.57E-03**	95.6	86.4	**97.3**

* PKC stands for PIKACHU, AGS ARGUS, and NG NETGUARDIAN.

To show NETGUARDIAN's superior performance compared to related works, we use ARGUS [10] and PIKACHU [4] for comparison from the perspective of network anomaly detection. Notice that there exists no related work that precisely aligns with our scope, as shown in Table I. Specifically, we evaluate the performance of ARGUS and PIKACHU on the same datasets as NETGUARDIAN (Section VI-B), and measure the data processing rate, FPR, and TPR, as summarized in Table IV, all pertaining to network edges. Among all three metrics, NETGUARDIAN achieves the best results, with a data processing rate of 45K edges per second. Notably, NETGUARDIAN is 1.35× and over 170× faster than ARGUS and PIKACHU, respectively, while maintaining an FPR orders of magnitude lower than both. In terms of TPR, NETGUARDIAN outperforms ARGUS and PIKACHU by an average of 12.6% and 1.7%, respectively. All these results demonstrate the efficiency and effectiveness of NETGUARDIAN in real-world scenarios.

In our system, we ensure a low false positive rate by using features specifically tailored to be indicative of APT attacks. These features, including patterns of periodic connections in C2 communications or network scanning behaviors, are highly unlikely to manifest under normal network conditions. This specificity significantly reduces the chances of benign activities being erroneously flagged as malicious. Moreover, our sophisticated threshold-setting mechanism, based on the analysis of historical network data, establishes accurate baselines for normal activities, enabling dynamic adjustment of thresholds as network behaviors evolve. Additionally, NETGUARDIAN's comprehensive detection framework, integrating both intra- and inter-stage behavioral analyses, offers further validation of anomalies across various APT stages. Through this combination of highly indicative connection features, dynamic and well-calibrated threshold settings, and a comprehensive detection framework, NETGUARDIAN ensures that the incidence of false positives is minimized.



(a) False negative rate (b) False positive rate

Fig. 5: FNR and FPR in the top three paths under different network sizes.

RQ2: Selection of Network Sizes

In the real world, APT attacks can persist over an extended period. Hence, in this RQ, we aim to investigate whether NETGUARDIAN can sustain its exceptional performance while expanding the scale of network traffic. Specifically, we increase the time span of the background traffic from one day to two, four, and eight days, respectively, and merge simulated attack traces into this expanded background traffic. We then conduct the same evaluation as in RQ1. The results show the ranking of the ground-truth path is unchanged as the network size grows, with all attack paths consistently ranked first among the reconstructed paths across the three simulated attacks. Figure 5 shows the FPR and FNR results, corresponding to the value of three on the x-axis in Figure 4. We can see that NETGUARDIAN's performance remains virtually unchanged as the number of days increases. Notably, the FPR is zero when only the top-ranked path is considered. These results are unsurprising since NETGUARDIAN relies on attack features and casual dependency for stage detection and path reconstuction. Essentially, the effectiveness of NETGUARDIAN is unaffected by the network size, both spatially and temporally.

RQ3: Revealing APT-Attack Stages

A key advantage of NETGUARDIAN is its capability to explain the key stages of ATP attacks thanks to the application of stage detection models. This is particularly helpful since the identification of these stages requires domain expertise and can be labor intensive. Existing solutions (e.g., those based on system logs [50]) demand further analysis to identify the attack stages even though the path is detected. Below we present one representative attack to demonstrate NETGUARDIAN's competence on revealing APT-attack stages.

`OpTC_0923`: Figure 1 shows the workflow of the APT attack conducted by red team on day one in the DARPA OpTC dataset. The number represents the network activity in the order of time, and bold green edges represent anomalous traffic that is detected by NETGUARDIAN. The ground-truth path is `C2` (S1) → `host1` (S2, S3) → `host2` (S3) → `host3` (S3) → `DC1` (S2), where the stage ID after each node is the one identified by NETGUARDIAN. Based on the red-team report from the OpTC dataset, NETGUARDIAN detects all stages correctly. Through these key stages, security analysts can easily understand the attack logic and take actions accordingly.

For all other attacks, NETGUARDIAN achieves the same results as `OpTC_0923`, demonstrating its superior performance on the interpretability of attack paths.

RQ4: System Performance

We measure the performance of NETGUARDIAN for each of its modules. The results are shown in Table II. All statistics in the table are calculated from datasets on a daily basis. Particularly, for the historical information extractor, we first recorded the execution time, memory usage, and the number of flows for each dataset (one-day span), and then took the average value over ten days for our monitored traffic and five days for the OpTC dataset. Note that we measure the data preprocessing costs as the performance of the data merging module for the OpTC dataset. In the high level, it takes about ten minutes for each of the first three modules to complete the analysis of one-day data. It takes about 20 minutes for the attack path reconstructor module to generate detection results for our monitored dataset. The memory usage is stable across all modules except `OpTC_0924` that has significantly more data to process compared to other OpTC attacks. Overall, NETGUARDIAN demands a reasonable amount of system resources to run, making it practical in real-world scenarios.

VII. RELATED WORK

In this section, we delve into existing NIDS, whose research falls into the following distinct categories.

Signature-based NIDS. Some studies detect a specific type of anomalies through behavioral pattern analysis, such as botnet [53], cryptojacking malware [19], and APT-related malware [54]. In recent years, researchers leverage programmable switches [16], [22] to achieve network intrusion detection. There are also some works focusing on a specific stage of APT attacks, such as lateral movement [34], periodical communication between C2 servers and controlled hosts [55], and data leakage [9] in the task-completion phase. However, all these NIDS lack NETGUARDIAN's ability to causally correlate anomalies, thus fail to provide interpretable detection results for multi-stage attacks like APTs.

Machine Learning-based NIDS. Some NIDS leverage traditional machine learning algorithms to learn normal traffic behavior and detect anomalies. For example, researchers [56] use clustering algorithms to detect outliers that are inconsistent with normal traffic behavioral patterns. Some works employ deep learning techniques [18], [24] or graph neural networks [3], [4], [10], [20] to detect traffic anomalies. Although the aforementioned NIDS can effectively detect anomalies, they focus on detecting a single attack stage, or lack analysis of the correlation between various anomalies. In contrast, NETGUARDIAN uses statistical models targeting each APT attack phase individually with much less false positives, and reconstructs APT paths to facilitate more accurate and detailed investigation.

VIII. CONCLUSION

This paper introduces NETGUARDIAN, a network intrusion detection system designed to address the limitations of current NIDS in detecting sophisticated APTs. By leveraging multi-stage causal analytics, NETGUARDIAN precisely correlates the various phases of APTs by constructing attack progression graphs. Our system outperforms SOTA NIDS in identifying APT behaviors, prioritizing high-risk alerts, and enabling threat detection through efficient traffic analysis. We present a robust validation of NETGUARDIAN using real-world network data, demonstrating its capabilities to accurately reconstruct multi-stage attacks within complex enterprise environments.

REFERENCES

[1] "Cyberattack on russian scientific research center," https://news.yahoo.com/military-intelligence-cyberattack-russian-scientific-084313996.html.

[2] "The SolarWinds Cyber-Attack: What You Need to Know," https://www.cisecurity.org/solarwinds/.

[3] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," in *NDSS*, 2022.

[4] R. Paudel and H. H. Huang, "Pikachu: Temporal walk based dynamic graph embedding for network anomaly detection," in *NOMS*, 2022.

[5] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "NoDoze: Combatting threat alert fatigue with automated provenance triage," in *NDSS*, 2019.

[6] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation." in *USENIX Security*, 2007.

[7] R. Basnet, S. Mukkamala, and A. H. Sung, "Detection of phishing attacks: A machine learning approach," in *Springer Soft computing applications in industry*, 2008.

[8] X. Wang, K. Zheng, X. Niu, B. Wu, and C. Wu, "Detection of command and control in advanced persistent threat based on independent access," in *IEEE ICC*, 2016.

[9] M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for advanced persistent threat detection," *Computer Networks*, 2016.

[10] J. Xu, X. Shu, and Z. Li, "Understanding and bridging the gap between unsupervised network representation learning and security analytics," in *IEEE S&P*, 2024.

[11] "DARPA OPTC," https://github.com/FiveDirections/OpTC-data.

[12] J. Gui, D. Li, Z. Chen, J. Rhee, X. Xiao, M. Zhang, K. Jee, Z. Li, and H. Chen, "Aptrace: A responsive system for agile enterprise level causality analysis," in *IEEE ICDE*, 2020.

[13] J. Gui, X. Xiao, D. Li, C. H. Kim, and H. Chen, "Progressive processing of system-behavioral query," in *ACSAC*, 2019.

[14] W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, D. Wang, Z. Chen, Z. Li, J. Rhee, J. Gui *et al.*, "This is why we can't cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage," in *ACSAC*, 2020.

[15] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *IEEE S&P*, 2020.

[16] S. Yoo, X. Chen, and J. Rexford, "Smartcookie: Blocking large-scale syn floods with a split-proxy defense on programmable data planes," in *USENIX Security*, 2024.

[17] H. Zhou, S. Hong, Y. Liu, X. Luo, W. Li, and G. Gu, "Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches," in *IEEE S&P*, 2023.

[18] J. Piet, A. Sharma, V. Paxson, and D. Wagner, "Network detection of interactive ssh impostors using deep learning," in *USENIX Security*, 2023.

[19] E. Tekiner, A. Acar, and A. S. Uluagac, "A lightweight iot cryptojacking detection mechanism in heterogeneous smart home networks," in *NDSS*, 2022.

[20] C. Fu, Q. Li, and K. Xu, "Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis," *arXiv preprint arXiv:2301.13686*, 2023.

[21] H. Seo and M. Yoon, "Generative intrusion detection and prevention on data stream," in *USENIX Security*, 2023.

[22] Y. Dong, Q. Li, K. Wu, R. Li, D. Zhao, G. Tyson, J. Peng, Y. Jiang, S. Xia, and M. Xu, "Horuseye: A realtime iot malicious traffic detection framework using programmable switches," in *USENIX Security*, 2023.

[23] P. Rieger, M. Chilese, R. Mohamed, M. Miettinen, H. Fereidooni, and A.-R. Sadeghi, "Argus: Context-based detection of stealthy iot infiltration attacks," *arXiv preprint arXiv:2302.07589*, 2023.

[24] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "Insomnia: Towards concept-drift robustness in network intrusion detection," in *ACM AISec*, 2021.

[25] A. Corsini and S. J. Yang, "Are existing out-of-distribution techniques suitable for network intrusion detection?" in *IEEE CNS*, 2023.

[26] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "Packet-level signatures for smart home devices," in *NDSS*, 2020.

[27] J. Xing, W. Wu, and A. Chen, "Ripple: A programmable, decentralized link-flooding defense against adaptive adversaries," in *USENIX Security*, 2021.

[28] H. Griffioen, K. Oosthoek, P. van der Knaap, and C. Doerr, "Scan, test, execute: Adversarial tactics in amplification ddos attacks," in *ACM CCS*, 2021.

[29] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *ACM CCS*, 2021.

[30] F. Pierazzi, G. Apruzzese, M. Colajanni, A. Guido, and M. Marchetti, "Scalable architecture for online prioritisation of cyber threats," in *IEEE CyCon*, 2017.

[31] B. E. Ujcich, S. Jero, R. Skowyra, A. Bates, W. H. Sanders, and H. Okhravi, "Causal analysis for software-defined networking attacks," in *USENIX Security*, 2021.

[32] A. Nadeem, S. Verwer, S. Moskal, and S. J. Yang, "Alert-driven attack graph generation using s-pdfa," *IEEE TDSC*, 2021.

[33] W. U. Hassan, D. Li, K. Jee, X. Yu, K. Zou, D. Wang, Z. Chen, Z. Li, J. Rhee, J. Gui *et al.*, "This is why we can't cache nice things: Lightning-fast threat hunting using suspicion-based hierarchical storage," in *ACSAC*, 2020.

[34] G. Ho, M. Dhiman, D. Akhawe, V. Paxson, S. Savage, G. M. Voelker, and D. A. Wagner, "Hopper: Modeling and detecting lateral movement." in *USENIX Security*, 2021.

[35] B. Stojanović, K. Hofer-Schmitz, and U. Kleb, "Apt datasets and attack modeling for automated detection methods: A review," *Elsevier Computers & Security*, 2020.

[36] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "Atlas: A sequence-based learning approach for attack investigation," in *USENIX security*, 2021.

[37] "Adversary emulation tool," https://github.com/center-for-threat-informed-defense/adversary_emulation_library.

[38] U. Von Luxburg, "A tutorial on spectral clustering," *Springer Statistics and computing*, 2007.

[39] "Blackorbird," https://github.com/blackorbird/APT_REPORT.

[40] "Aptnotes," https://github.com/aptnotes/data.

[41] A. Lemay, J. Calvet, F. Menet, and J. M. Fernandez, "Survey of publicly available reports on advanced persistent threat actors," *Elsevier Computers & Security*, 2018.

[42] I. Ghafir and V. Prenosil, "Advanced persistent threat and spear phishing emails," in *International Conference Distance Learning, Simulation and Communication*, 2015.

[43] P. S. Keila and D. B. Skillicorn, "Structure in the enron email dataset," *Springer Computational & Mathematical Organization Theory*, 2005.

[44] "Phishing corpus dataset," https://academictorrents.com/details/a77cda9-a9d89a60dbdfbe581adf6e2df9197995a.

[45] S. Huang, J. Cheng, and H. Wu, "Temporal graph traversals: Definitions, algorithms, and applications," *arXiv preprint arXiv:1401.1919*, 2014.

[46] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "Prographer: An anomaly detection system based on provenance graph embedding," in *USENIX Security*, 2023.

[47] M. J. Turcotte, A. D. Kent, and C. Hash, "Unified host and network data set," in *Data science for cyber-security*, 2019.

[48] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Elsevier Computer networks*, 1999.

[49] "APT Attack Groups," https://attack.mitre.org/groups/.

[50] P. Fang, P. Gao, C. Liu, E. Ayday, K. Jee, T. Wang, Y. F. Ye, Z. Liu, and X. Xiao, "Back propagating system dependency impact for attack investigation," in *USENIX Security*, 2022.

[51] M. M. Anjum, S. Iqbal, and B. Hamelin, "Analyzing the usefulness of the darpa optc dataset in cyber threat detection research," in *ACM SACMAT*, 2021.

[52] A. Zimba, H. Chen, Z. Wang, and M. Chishimba, "Modeling and detection of the multi-stages of advanced persistent threats attacks based on semi-supervised learning and complex networks characteristics," *Elsevier Future Generation Computer Systems*, 2020.

[53] C. Herley, "Automated detection of automated traffic," in *USENIX Security*, 2022.

[54] N. Virvilis and D. Gritzalis, "The big four-what we did wrong in advanced persistent threat detection?" in *IEEE ARES*, 2013.

[55] D. Tariq, B. Baig, A. Gehani, S. Mahmood, R. Tahir, A. Aqil, and F. Zaffar, "Identifying the Provenance of Correlated Anomalies," in *SAC*, 2011.

[56] C. Fu, Q. Li, K. Xu, and J. Wu, "Point cloud analysis for ml-based malicious traffic detection: Reducing majorities of false positive alarms," in *ACM CCS*, 2023.