

Highlights

SECTRACER: A Framework for Uncovering the Root Causes of Network Intrusions via Security Provenance

Seunghyeon Lee, Hyunmin Seo, Hwanjo Heo, Anduo Wang, Seungwon Shin, Jinwoo Kim

- We introduce network-level security provenance to analyze multihost intrusion behaviors.
- SECTRACER, built on SDN, uses probabilistic logic to automate root cause analysis of APTs.
- Our approach enables accurate attack forensics with low overhead and no system modifications.

SECTRACER: A Framework for Uncovering the Root Causes of Network Intrusions via Security Provenance^{*}

Seunghyeon Lee^{a,1}, Hyunmin Seo^{b,1}, Hwanjo Heo^c, Anduo Wang^d, Seungwon Shin^b and Jinwoo Kim^{e,*}

^aS2W, Pangyoeyeok-ro 192beon-gil, Seongnam-si, 13524, Republic of Korea

^bKAIST, 291 Daehak-ro, Yuseong-gu, Daejeon, 34141, Republic of Korea

^cElectronics and Telecommunications Research Institute (ETRI), 218 Gajeong-ro, Yuseong-gu, Daejeon, 34129, Republic of Korea

^dComputer and Information Science, Temple University, SERC 342, N 12th St Philadelphia, PA, 19122, USA

^eSchool of Software, Kwangwoon University, 20 Kwangwoon-ro, Nowon-gu, Seoul, 01897, Republic of Korea

ARTICLE INFO

Keywords:

Security Provenance
Software-Defined Networking
Network Forensic
Network Intrusion Detection
Root Cause Analysis

ABSTRACT

Modern enterprise networks comprise diverse and heterogeneous systems that support a wide range of services, making it challenging for administrators to track and analyze sophisticated attacks such as advanced persistent threats (APTs), which often exploit multiple vectors. To address this challenge, we introduce the concept of *network-level security provenance*, which enables the systematic establishment of causal relationships across hosts at the network level, facilitating the accurate identification of the root causes of security incidents. Building on this concept, we present SECTRACER as a framework for a network-wide provenance analysis. SECTRACER offers three main contributions: (i) comprehensive and efficient forensic data collection in enterprise networks via software-defined networking (SDN), (ii) reconstruction of attack histories through provenance graphs to provide a clear and interpretable view of intrusions, and (iii) proactive attack prediction using probabilistic models. We evaluated the effectiveness and efficiency of SECTRACER through a real-world APT simulation, demonstrating its capability to enhance threat mitigation while introducing less than 1% network throughput overhead and negligible latency impact.

1. Introduction

With the advent of several key research innovations (e.g., network virtualization and mobility management), enterprise networks are actively evolving into the next generation, enabling them to support a wider range of diverse and complex services (e.g., e-commerce and e-voting). However, this transformation not only brings advancements but also introduces a critical challenge: network intrusions. For example, several *advanced persistent threats (APTs)* targeting critical assets within enterprise networks have emerged. These threats often compromise reachable hosts, using them as entry points to infiltrate the network and steal or manipulate critical assets, a technique known as lateral movement or stepping-stone attacks [1].

Network administrators are aware of these threats and have made considerable efforts to reduce network attack surfaces. To this end, they deployed a variety of security mechanisms (e.g., network intrusion detection systems and firewalls) to detect specific attack attempts [2, 3] and even outsourced security services to external providers [4]. However, despite these efforts, sophisticated attacks persist and often bypass traditional defenses. Therefore, administrators frequently fail to trace the origins and progression of these

attacks (e.g., identifying the initial entry point), leaving networks vulnerable to future unforeseen threats.

To address this challenge, many *provenance-based intrusion detection systems (PIDS)* have been proposed [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. These systems collect data on system and network activities, construct provenance graphs, and analyze the root causes of security incidents. However, they have several limitations. First, most existing PIDS focus on collecting and analyzing host-level provenances, making it difficult to capture and establish causal relationships across multiple hosts. Second, their monitoring coverage is limited because achieving comprehensive visibility requires the deployment of agents on all hosts, which is both costly and operationally burdensome. Third, provenance graphs often contain a vast number of vertices and edges, making manual investigation highly time-consuming and labor-intensive. We suggest that overcoming these limitations is crucial for effective root cause analysis and forensic investigation of complex network intrusions in enterprise networks.

In this study, we introduced SECTRACER, a novel framework for comprehensive provenance-based intrusion detection through network-wide monitoring. SECTRACER provides administrators with clear visibility into network intrusions, including their origin and progression, by continuously monitoring and collecting relevant network and security events to enable the reconstruction, analysis, and mitigation of advanced threats. To achieve this, we introduced the concept of *network-level security provenance*, an extension of traditional provenance that offers visibility

^{*}This is the accepted manuscript of an article accepted for publication in *Computers & Security*.

^{*}Corresponding author

✉ sl@s2w.inc (S. Lee); shm9574@kaist.ac.kr (H. Seo);
hwanjo@etri.re.kr (H. Heo); adw@temple.edu (A. Wang);
claud@kaist.ac.kr (S. Shin); jinwookim@kw.ac.kr (J. Kim)

ORCID(s): 0000-0003-1303-8668 (J. Kim)

¹Co-first authors

into how specific network events are created, modified, and evolve, facilitating the efficient tracking of changes and precise identification of intrusion root causes. By leveraging network-level security provenance, SECTRACER uncovers the origins of enterprise security incidents, providing deeper insights into attack paths and strengthening network defense mechanisms.

At a high level, SECTRACER collects security and related audit data (e.g., network events) by utilizing software-defined networking (SDN). It considers packets and network snapshots—including network topology, security policies, configurations, and forwarding rules—as primitive provenance data and captures them in chronological order to provide sufficient forensic information to reconstruct the attack history. Using the collected data, SECTRACER constructs a *network-level security provenance graph* to represent the operational flow of attacks across hosts, illustrating not only the causal relationships of security events with their origins but also the forensic details that characterize attack behaviors. To generate the provenance graph, SECTRACER begins its analysis from a specific event or host defined by the network administrator, and performs a causality analysis to uncover the relationships between discrete security diagnostic results.

To streamline root cause analysis from complex provenance graphs, SECTRACER leverages probabilistic models [31]. It extracts information related to network security issues from a security provenance graph and structures it into known relationships using first-order logic. By applying predefined root cause analysis (RCA) rules, SECTRACER models the attacker behavior and prioritizes various root cause insights for each subject. Finally, it exports the constructed security provenance graph, enhancing administrators' ability to analyze suspicious activities with maximum security visibility. This approach significantly reduces the overhead of manual security data analysis and minimizes the risk of overlooking critical information, such as the starting point of an APT attack.

Contributions. The key contributions of this study are as follows:

- We proposed *network-level security provenance* enriched with diverse features collected from an enterprise network. This is enabled by SDN, which provides centralized control and fine-grained traffic monitoring.
- We designed a method for performing RCA on complex provenance graphs by leveraging probabilistic soft logic (PSL), which automatically identifies intricate attack patterns based on predefined rules.
- We developed a prototype system, SECTRACER, which implements network-level security provenance in modern enterprise networks, efficiently collecting comprehensive security and network audit data without requiring modifications to existing infrastructure or components.

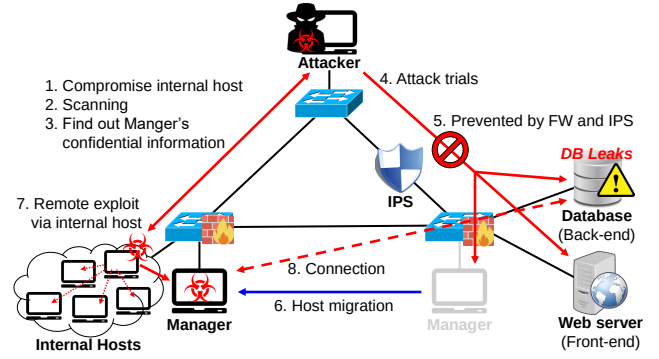


Figure 1: Example scenario where an attacker aims to steal confidential information through the manager's machine via an APT [1]. The dotted line indicates that the attacker controls compromised victims.

- We evaluated the effectiveness of SECTRACER by demonstrating its capability to trace the root causes of intrusion scenarios (i.e., an APT attack and botnet propagation) and assess its efficiency by measuring system overhead and performance.

The remainder of this paper is organized as follows: Section 2 details the problem statement, including a motivating example, our research goals, and the threat model. Section 3 discusses the related studies on provenance-based intrusion detection. The high-level design and architecture of SECTRACER are presented in Section 4. Subsequently, we describe its three core components in detail: provenance collection (Section 5), provenance analysis (Section 6), and RCA (Section 7). Section 8 presents a comprehensive evaluation of the proposed framework's effectiveness and performance. Finally, Section 9 discusses limitations and future work.

2. Problem Statement

2.1. Motivating Example

Figure 1 illustrates a typical APT scenario targeting an enterprise network. In this scenario, the attacker compromises the database and web servers. ① (**Initial Access**) The attacker first attempts to directly attack an internal host and successfully gains access. ② (**Discovery and Reconnaissance**) Next, the attacker conducts port scanning on other internal hosts to gather intelligence about the network. ③ (**Enumeration and Credential Access**) After multiple scanning attempts, the attacker uncovers details about the network infrastructure. ④ (**Lateral Movement and Privilege Escalation Attempts**) Using this information, the attacker attempts to directly compromise the database server, the web server, and the manager's machine. ⑤ (**Defense Evasion and Access Control**) This initial attempt is blocked by the intrusion prevention system (IPS) and firewall rules configured by the administrator, which restrict remote access to critical assets. At this stage, the internal systems remain well-protected. ⑥ (**Configuration Drift and Security Gap**)

Table 1

Mapping of scenario steps to the MITRE ATT&CK framework, based on an attack scenario designed to reflect realistic conditions of an APT incident.

Step	MITRE ATT&CK Tactic	Technique ID & Name
①	Initial Access	T1190 – Exploit Public-Facing Application T1566 – Phishing
②	Discovery	T1046 – Network Service Discovery T1018 – Remote System Discovery
③	Credential Access / Discovery	T1083 – File and Directory Discovery T1555 – Credentials from Password Stores
④	Lateral Movement / Privilege Escalation	T1021 – Remote Services T1078 – Valid Accounts
⑤	Defense Evasion (<i>Prevented</i>)	T1562 – Impair Defenses (<i>attempt failed</i>)
⑥	Defense Evasion	T1078 – Valid Accounts T1562 – Impair Defenses
⑦	Lateral Movement	T1021 – Remote Services T1550 – Use Alternate Authentication Material
⑧	Privilege Escalation / Impact / Exfiltration	T1068 – Exploitation for Privilege Escalation T1005 – Data from Local System T1041 – Exfiltration over C2 Channel

Exploitation) However, after migrating the manager machine to another switch, the administrator neglects to update security configurations. ⑦ (**Exploitation and Lateral Movement**) Exploiting this misconfiguration, the attacker gains access to the internal host and subsequently moves laterally to the manager machine. ⑧ (**Privilege Escalation, Impact, and Data Exfiltration**) After attacking the manager machine, the attacker escalates privileges, breaching the database server and gaining access to sensitive records. To enhance the relevance and clarity of the scenario, each step of the attack was mapped onto the MITRE ATT &CK [32] framework, as summarized in Table 1.

Although all the aforementioned attack attempts could be prevented, either by maintaining the system up-to-date (e.g., patching the kernel) or by deploying advanced security measures, the primary challenge for administrators lies in the vast number of possible attack scenarios. The sheer size of an attack surface makes it extremely difficult to anticipate and mitigate potential threats. Furthermore, when an attack occurs, administrators may lack the necessary tools to detect the theft of confidential information. Security defense mechanisms may subsequently flag unusual database access patterns or document leaks; however, linking these alerts to the full sequence of attack events remains a significant challenge. In other words, although security mechanisms can capture observable signs of attack attempts, they often fail to explain their root causes, such as *host migration*, or correlate historical events that combinedly constitute a successful attack. In this context, no clear or comprehensive solution exists.

2.2. Research Goal and Limitations

As demonstrated in the aforementioned example, understanding how an attacker exfiltrates confidential information is extremely challenging from an administrator's perspective. Therefore, our goal is to systematically analyze such complex attacks using *RCA*, a method for identifying the underlying origins of a problem or event. In the context of security, *RCA* involves a structured sequence of security

diagnostics, including (i) threat recognition and impact assessment, (ii) investigation to trace the origins of security incidents, and (iii) security enhancement by addressing root causes. However, achieving these objectives in current enterprise networks is hindered by three fundamental limitations (L1–L3):

L1. Limited Monitoring Coverage. To minimize the deployment overhead, many enterprise administrators position security devices at network choke points [33, 34], balancing concerns such as performance degradation (e.g., deep packet inspection) and network complexity (e.g., flow management). However, selective security monitoring also introduces critical vulnerabilities. As illustrated in Figure 1, key security events, such as ①, ②, and ⑦, are essential for detecting and preventing information leakage attacks. However, because of the attacker's ability to bypass the coverage of the IPS and firewalls, these crucial events may go unnoticed by administrators. Administrators can enhance the monitoring coverage by deploying security agents across all hosts. However, these agents continuously collect additional security-related data, which imposes resource overhead on the host and potentially degrades system performance [13, 14]. Furthermore, this method is inherently limited to scenarios in which users operate devices without a preinstalled security agent because the behavior of such devices remains undetected [35]. Consequently, detecting security incidents at the network level is crucial because it enables the identification of threats irrespective of host-specific configurations and provides a network-wide perspective on potential security breaches.

L2. Limited Network-wide Causality Analysis. Attackers often employ various techniques to execute multiple attack attempts to achieve their objectives. Although traditional attack detection systems, such as security information and event management (SIEM) and intrusion detection systems (IDS), can effectively identify individual attack attempts, they are limited in determining the root cause. To overcome this limitation, PIDS have been studied extensively

[5, 6, 7, 9, 10, 11]. These systems construct provenance graphs by capturing system events, such as process creation, file modifications, and network communications, offering a comprehensive view of malicious activities. The provenance graph enables a fine-grained causality analysis, allowing administrators to trace attack chains, analyze data dependencies, and pinpoint root causes. However, the existing PIDS face challenges in detecting attacks that span multiple hosts. The lack of network-wide correlation prevents them from reconstructing multihost attack paths and effectively detecting lateral movements in distributed network environments.

L3. Overwhelming Security Alerts. Administrators must prioritize security alerts based on their severity and impact on enterprise infrastructure. However, according to Trend Micro's threat reports [36], 70% of security teams are emotionally overwhelmed by security alert volumes, which makes it difficult to triage all potential threats. Large and complex enterprise environments generate enormous volumes of alerts, which often lead to alert fatigue. Despite existing efforts to prioritize security events [37, 38], administrators may struggle to identify the most critical alerts amid noise or assess their broader impact on infrastructure. For instance, in Figure 1, an administrator may overlook security alerts related to early attack attempts (①), even though these events could lead to severe consequences, such as the leakage of confidential information.

2.3. Assumptions and Threat Model

Assumptions. We assume an SDN-enabled enterprise network in which an administrator defines high-level security requirements and specifies which assets must be protected. To satisfy these requirements, the administrator deploys network security mechanisms using either dedicated security devices (e.g., NIDS/network intrusion prevention systems (NIPS)) or SDN controllers, thereby enabling flow-level access control and monitoring. However, owing to the dynamic nature of enterprise networks—such as topology changes or policy conflicts across devices—we assume that misconfigurations or human errors may lead to discrepancies between the intended high-level requirements and their actual enforcement, potentially enabling APTs by adversaries.

Threat Model. We consider adversaries who attempt to gain unauthorized access to internal assets in an enterprise network with malicious intent, such as stealing sensitive information, disrupting services, or escalating privileges, to attack protected resources. These adversaries may employ a range of penetration techniques, including remote exploitations, bypassing security devices, leveraging stepping stones, or compromising network components such as switches and links. To focus on the core objectives of SEC-TRACER, we assume that adversaries cannot compromise or manipulate SEC-TRACER itself (e.g., its internal logic or

execution), nor can they tamper with the trusted components responsible for collecting provenance data. However, we assume that adversaries can influence the network environment (e.g., via ARP spoofing), thereby affecting the accuracy or completeness of the provenance data collected by SEC-TRACER.

3. Related Work

Provenance-based forensic and IDS are increasingly applied across various domains, extending beyond traditional networks to environments such as IoT and cloud computing [5, 6, 7, 8, 9, 10, 11]. These systems operate by gathering host events, such as system calls [8, 12, 13, 14], and analyzing attack attempts based on the causal relationships between the collected events. This approach facilitates the effective clarification of the relationships between events, thereby enhancing the capability of detecting more intricate attack patterns. These PIDS identify attacks with one of the two primary types: (i) anomaly-based or (ii) rule-based attacks.

Anomaly-based. Anomaly-based provenance systems focus on the construction and analysis of provenance graphs that encapsulate the relationships and interactions between events occurring in the host [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. Using advanced graph machine learning algorithms, these systems analyze provenance graphs to detect anomalies that signify deviations from established normal behavior, indicating potential intrusions. This approach is particularly effective in identifying zero-day exploits and sophisticated malware that may not match known signatures or rules.

Rule-based. By contrast, rule-based provenance systems employ a methodology in which collected host events are meticulously matched against an established set of rules that define the characteristics of attacks [15, 16, 17, 18]. These rules model APT attacks as sequences of malicious actions, thereby allowing for the systematic detection of similar attack patterns within the network. Moreover, these systems enrich the detection process by correlating identified attacks with comprehensive attack-related databases such as the MITRE ATT&CK framework. This correlation not only facilitates a more detailed understanding of attack tactics, techniques, and procedures but also enhances the contextualization of security alerts, making them more actionable for security analysts.

Recent innovations in this domain have integrated network event monitoring to provide holistic security overviews. For example, P4CONTROL [41] is a pioneering tool that leverages eBPF for the collection of host events and extends its surveillance capabilities to observe network events between hosts within programmable switches. By applying rule-based analytics, this tool enhances the ability to detect and respond to complex attacks across multiple hosts in both host and network layers. PivotWall [42]

Table 2

Comparison of previous provenance-based forensic and IDS and SEC TRACER.

(● Fully applicable, ● Partially applicable, and ○ Not applicable)

Type	Work	Target	SDN-based Design	Network-level Provenance	Automatic RCA
Forensic Systems	Hi-Fi [5]	OS Kernel	○	○	○
	SPADE [6]	OS Kernel	○	○	○
	LPM [7]	OS Kernel	○	○	○
	CamFlow [8]	Cloud	○	●	○
	Clarion [9]	Microservice	○	●	○
	Alastor [10]	Serverless	○	●	○
Anomaly-based Detection	Poirot [19]	Enterprise Host	○	○	●
	Nodoze [20]	Enterprise Host	○	○	●
	Unicorn [21]	Enterprise Host	○	○	○
	ProvDetector [22]	Malware	○	○	○
	Atlas [23]	Enterprise Host	○	●	●
	ThreaTrace [24]	Enterprise Host	○	○	○
	ShadeWatcher [25]	Enterprise Host	○	○	○
	Prographer [26]	Enterprise Host	○	●	●
	Magic [27]	Enterprise Host	○	○	○
	R-CAID [28]	Enterprise Host	○	○	●
	Kairos [29]	Enterprise Host	○	●	●
	Flash [30]	Enterprise Host	○	●	●
Rule-based Detection	SLEUTH [15]	Enterprise Host	○	●	●
	Holmes [16]	Enterprise Host	○	●	●
	MORSE [17]	Enterprise Host	○	●	●
	RapSheet [18]	Enterprise Host	○	●	●
	ProvTalk [11]	NFV	○	●	●
	ForenGuard [39]	SDN Attacks	●	●	○
	ProvSDN [40]	SDN Attacks	●	●	●
	P4Control [41]	Enterprise Network Intrusions	○	●	●
	SEC TRACER (our work)	Enterprise Network Intrusions	●	●	●

proposed a security framework that integrates information flow tracking with SDN to detect and mitigate APTs and stepping-stone attacks within enterprise networks.

Comparison with SEC TRACER. In response to the evolving landscape of network security, SEC TRACER was specifically designed for enterprise networks, distinguishing itself from traditional systems that primarily focus on single-host events. In contrast to these conventional approaches, SEC TRACER conducts comprehensive network-wide monitoring across multiple hosts using an SDN to facilitate an in-depth RCA. This capability enables a more precise understanding of network dynamics and anomalies, allowing administrators to trace the origins of security incidents accurately. By offering a specialized approach to network event analysis, SEC TRACER significantly enhances threat detection and mitigation through SDN infrastructure, thereby strengthening the overall security posture of these critical networks. Table 2 summarizes our analysis results, which compare SEC TRACER with previous solutions in terms of (i) SDN-based centralized monitoring, (ii) network-level security provenance, and (iii) automatic RCA, which are the main design considerations.

4. SEC TRACER Design

In this section, we present the high-level approach of SEC TRACER and describe its architecture and key components.

4.1. High-level Approaches

SEC TRACER enables administrators to perform RCA to identify potential security issues in enterprise networks by assessing their impact on the infrastructure. To this end, SEC TRACER considers three high-level approaches (A1–A3):

A1. SDN-based Centralized Monitoring. SEC TRACER aims to achieve information-lossless forensics to ensure that no security events or issues are overlooked, thus enabling administrators to accurately reconstruct the events occurred during a security incident. To achieve this, SEC TRACER leverages the centralized visibility and flow-level monitoring capabilities of SDN [34, 43, 44], enabling consistent tracking of host-to-host communication across complex enterprise environments. Although SDN traffic data provide broad coverage, they are insufficient to fully explain the cause, method, and location of

security incidents. To address this, SECTracer collects a variety of *security snapshots*, including security policies, network configurations, host profiles, network topology (including host locations), forwarding rules, and provenance monitoring data captured at specific points in time. These datasets, collectively termed as *security provenance data*, serve as the foundation for security diagnosis, enabling the identification of security causalities and providing a comprehensive view of attack progression.

A2. Network-wide Provenance Graph Construction.

SECTracer introduces a *network-level security provenance graph* that captures interhost security events in an enterprise network. Traditional host-level causality delineates the causal relationships among events associated with the host system resources, including processes, files, and network sockets. We generalize this concept to the network level, enabling the representation of causal dependencies between network security events across diverse hosts within an enterprise environment, such as servers, databases, employee workstations, and administrative systems. Consequently, an entire attack sequence over a network is represented as a series of historical security causalities, illustrating how each host is affected and how the attack progresses over time. For example, in Figure 1, the red arrows represent causalities, providing administrators with critical insights into the flow of an attack and enabling them to recognize potential information leaks.

A3. Automatic RCA. SECTracer prioritizes endangered network assets in enterprise networks through *rule-based root cause analyses*, thereby enabling administrators to quickly identify and inspect the most critical assets in response to security incidents. To achieve this, we leveraged *PSL*, a framework that is widely used in areas such as natural language processing, social network analysis, and knowledge graphs [45]. Our approach organizes various network and security events into a relational form and applies PSL to determine which hosts are attacked, which are the potential targets, and what types of high-risk incidents have occurred. Prioritization is guided by predefined rules to ensure an efficient incident response. For example, in Figure 1, the internal host controlled by the attacker is classified as both a *target* and a *suspect*, which helps administrators quickly identify critical assets at risk. This approach differs significantly from traditional solutions, which primarily assess security events in isolation based on their severity scores [37, 38] without considering their broader context or relational impact.

4.2. SECTracer Overview

Figure 2 illustrates the architecture of SECTracer, which was designed based on the proposed approach. At a high level, an administrator must provide SECTracer with three key operational requirements: (i) *security policy*, which defines a routing policy specifying which hosts should pass through specific security devices and be

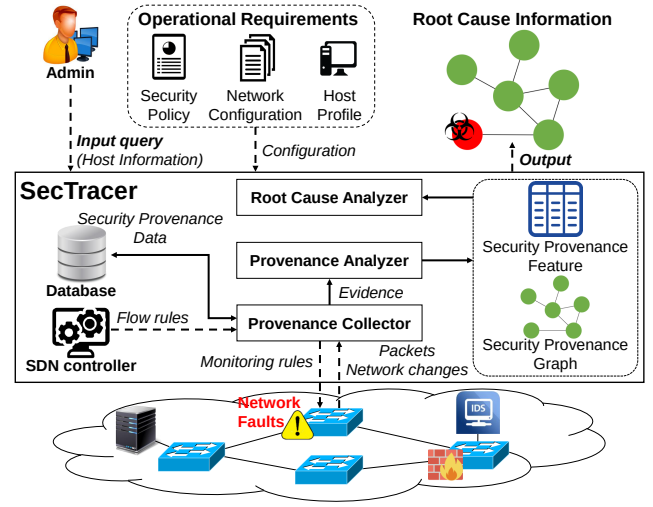


Figure 2: Overall architecture of SECTracer. The SECTracer is hosted over a wide enterprise infrastructure above the SDN control plane. The solid and dotted lines indicate an internal and external event, respectively.

subject to access control rules, (ii) *network configurations*, which include details about the locations of security devices or middleboxes within the network, and (iii) *host profile*, which specifies which security provenance data should be monitored for certain hosts and outlines the security policies governing them. Based on these inputs, SECTracer constructs a *security provenance graph* enriched with root cause information to aid the security analysis.

SECTracer comprises three main modules: (i) a *provenance collector*, (ii) a *provenance analyzer*, and (iii) a *root cause analyzer*. The *provenance collector* gathers the security provenance data, generates supporting evidence, and publishes both to a database. For provenance monitoring, SECTracer leverages SDN to efficiently collect network-related information, such as topology and packet traces. With SDN-centralized network management, the provenance collector tracks network changes by monitoring SDN control messages and issues monitoring flow rules to capture packets corresponding to installed or removed flow rules using the SDN control plane¹. The *provenance analyzer* receives relevant evidence from the provenance collector based on the input host information and constructs a security provenance graph by identifying security causalities among the collected provenance data. Finally, the *root cause analyzer* processes the provenance graph to pinpoint the root cause of security incidents.

Next, Section 5, Section 6, and Section 7 present the details of SECTracer for the provenance collector, provenance analyzer, and root cause analyzer, respectively.

¹We used OpenFlow v1.3 as the southbound SDN protocol [46].

5. Provenance Collection

The main purpose of the *provenance collector* is to collect security provenance data across an enterprise infrastructure and create evidence with the corresponding security snapshots. This approach incorporates three major components: *packet monitoring*, *security snapshot recording*, and *evidence generation*.

5.1. Packet Monitoring

The provenance collector reads the configurations specified in the host profile provided by the administrator, detects flow-rule events from network applications running on the SDN controller, and issues provenance-monitoring rules to capture raw packets in the data plane.

Two-Phase Packet Capture. To enhance packet tracking, we proposed a novel method called *two-phase packet capture* (2PC) that is designed to monitor both the integrity and reachability of packets. This approach enables monitoring at two distinct strategic points to maximize packet visibility. It operates by duplicating packets at the egress and ingress switches using monitoring rules and forwarding both copies to the provenance collector for further analysis.

Figure 3 illustrates how SEC-TRACER leverages SDN techniques, specifically OpenFlow commands² to capture packets at both the ingress and egress points of the flow. In this scenario, *H1* sends a packet to *H2*, and switches *S1* and *S2* have flow rules in place to forward the packet from *H1* to *H2* (Rules #1 and #4). The packet monitor is assumed to have preinstalled monitoring rules for 2PC using OpenFlow commands (Rules #2, #3, #5, and #6). When a packet arrives at the ingress switch, *Rule #2* duplicates it using two actions: output and goto_table. Subsequently, *Rule #3* applies the push_vlan action to tag a duplicated packet and forwards it to the evidence generator over an isolated network. Similarly, when the egress switch (*S2*) receives a packet forwarded by *S1*, its monitoring rules operate in the same manner as those of the ingress switch, ensuring comprehensive packet tracking across the flow path.

Deploying Monitoring Rules. To deploy monitoring rules with 2PC, the provenance collector maintains a list of hosts based on the host profile specified in the operational requirements³. Whenever a network application or administrator issues flow rules⁴, the packet monitor validates whether the IPv4 address ranges specified in the host profile match those specified in the issued flow rules. If a match is found, the current topology information is retrieved to identify the ingress and egress switches of the host pairs, as well as the edge switch connected to the evidence generator. To locate these switches, SEC-TRACER leverages a network-invariant checker to determine the end-to-end paths [47, 48] and considers packet detouring through network middleboxes,

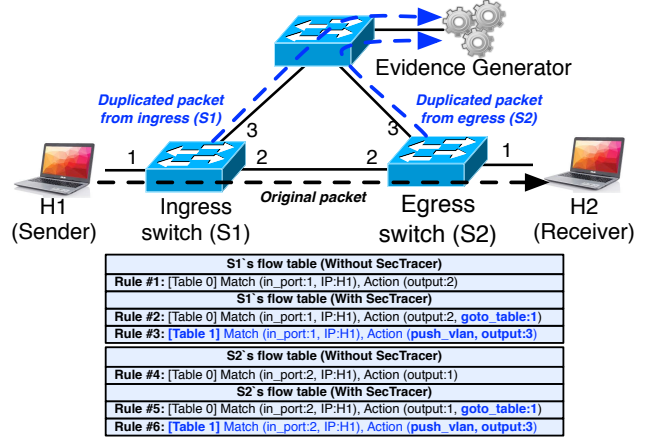


Figure 3: Working scenario of 2PC when *H1* sends a packet to *H2*. Monitoring parts are highlighted in blue.

eID	sID	timestamp	rPacket	mPacket	results
rID	SP	AC	reached	LIntegrity	Plntegrity
				exploitable	IDSMsg

Figure 4: Format of a single SEC-TRACER evidence. The gray box represents index fields for unique identifiers, and the white box is a list of feature fields. The text in bold represents features from the security devices, and the others in the white box are generated from enterprise-wide security inspections.

as specified in the network configuration, within the operational requirements⁵. When the relevant switches are identified, the provenance collector issues monitoring rules to enable packet duplication and forwarding, thereby ensuring comprehensive security provenance tracking.

5.2. Security Snapshot Recording

Changes in the operational requirements generate *configuration change* events that notify the provenance collector of updates for security snapshots. The network changes track the topology modifications (e.g., switch and port up/down), host location changes, and flow table updates for each switch. Because popular SDN controllers, such as ONOS [49] and OpenDaylight [50], maintain local states for topology information, host locations, and internal flow tables per switch, the provenance collector reads these local states to monitor and track network changes effectively. The provenance collector maintains historical security snapshots to maintain the internal states while monitoring the packets. Technically, whenever any security policies, network configurations, host profiles, topology including host locations, flow entries in each switch, and packet monitoring information are changed at a particular point in time, provenance collector publishes an updated snapshot of SEC-TRACER's database with a unique identifier specifying its version.

²We use the `flow_mod` action to install the flow rules.

³Each host profile is defined using CIDR notation.

⁴For flow rules installed or removed by network applications, SEC-TRACER monitors the `flow_mod` and `flow_removed` messages.

⁵The network configuration defines the locations of middleboxes.

5.3. Evidence Generation

The provenance collector generates evidence by analyzing a monitored packet and its corresponding security snapshot. Figure 4 illustrates the evidence format used by SECTRACER. Upon receiving two mirrored packets⁶, SECTRACER assigns unique identifiers, *eID* and *sID*, to the evidence and the associated security snapshot, respectively. It records the monitoring time in the *timestamp* field and then performs a security diagnosis on the provenance data to populate the feature fields. To ensure high-quality security diagnosis results, that is, accurate feature extraction, SECTRACER employs two key strategies: (i) *leveraging security devices* and (ii) *network-wide security inspections*. Details of these strategies are discussed as follows.

Leveraging Security Devices. To ensure compatibility with enterprise networks in which security devices have already been deployed, SECTRACER integrates existing security mechanisms, including network intrusion detection and prevention systems (NIDS/NIPS) and anomaly detection systems, as provenance sources. Specifically, the provenance collector leverages a software-based NIDS [51, 52] by forwarding monitored packets (i.e., packet streams) to an IDS instance for analysis. When an alert is triggered, the provenance collector assesses whether the detected activity corresponds to an exploitable attack⁷. If the attack is deemed exploitable, the *exploitable* field is set to *true*; otherwise, it is set to *false*. Finally, the raw alert messages are stored in *IDSMsg*, providing users with detailed insights into the detected threats.

Network-wide Security Inspections. Existing security devices have limited coverage because of their strategic deployment at network choke points, which restricts them from inspecting only a subset of the network flows [33, 34, 53]. Even when administrators have full control over individual host machines, they lack visibility of network-wide activities. To enable comprehensive network-wide monitoring, SECTRACER evaluates two key security aspects.

First, it monitors the *reachability* to determine whether an attacker's attempt successfully traverses the network under the current configuration. In particular, reachability is assessed by identifying a pair of packets that originate from the same source and comparing the immutable fields in the monitored packets [54]. By leveraging 2PC, a provenance collector receives one or two packets per attempt. If a matching pair is observed, the *reached* field is set to *true*. Otherwise, if no corresponding packet is found, the *reached* field is set to *false*, which indicates that the packet is dropped in the network.

Second, it checks for *policy violations* to ensure that the administrator's high-level security requirements are correctly enforced without errors or misconfigurations.

⁶If a packet is dropped, a single copy is forwarded to SECTRACER by the ingress switch.

⁷We used Snort *classtype* to assess an attack's exploitability. If the *classtype* of the Snort alert is *high*, SECTRACER classifies the attack as exploitable.

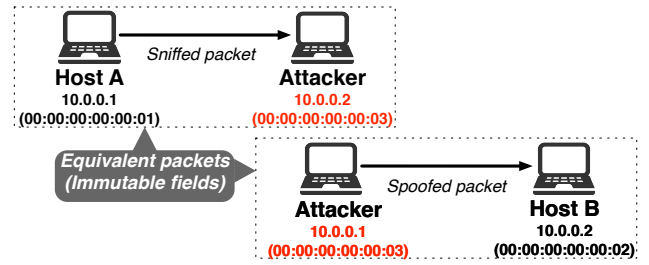


Figure 5: Example scenario of a local integrity violation between Hosts A and B, where an attacker intercepts a packet from Host A, spoofs its IP address to appear as Host A, and sends it to Host B.

To achieve this, SECTRACER examines whether any monitored packet violates the security policies defined in the operational specifications. Because the monitoring data include a list of middleboxes traversed by each packet, SECTRACER compares the observed path with the intended policy to verify the compliance. It also verifies whether a packet has been delivered to unauthorized hosts by validating it against the access control policy. If a violation is found in either the security routing policy or the access control policy, the fields *SP* and *AC* are set as the identifiers of the corresponding violated policies⁸.

Tamper-Resistant Provenance Collection. When SECTRACER collects provenance data, an attacker may attempt to tamper with it by manipulating packet headers such as IP addresses (Section 2.3). To ensure tamper-resistant provenance collection, SECTRACER detects integrity violations categorized into *physical* and *logical* types. A *physical integrity violation* occurs when an attacker manipulates a packet by compromising physical network components, such as links or switches. A packet is considered physically violated if (i) it is not dropped, and (ii) the monitored versions of the packet contain differing headers or payloads; in such cases, the *PIegrity* flag is set to *true*. In contrast, a *logical integrity violation* involves intercepting and relaying packets between hosts, as in man-in-the-middle attacks. Figure 5 illustrates an example scenario⁹ in which an attacker intercepts the packets, modifies the header fields, and replays them. A logical integrity violation is detected if (i) two packets within a connection share the same immutable fields, and (ii) a single MAC address is associated with two different IP addresses. When both conditions are satisfied, *LIntegrity* is set to *true*, and the attacker's identifier (*eID*) is stored as a reference ID (*rID*) in the corresponding evidence record. For both violation types, the provenance collector stores the original and manipulated packets in the *rPacket* and *mPacket* fields, respectively.

⁸Each policy entry is assigned a unique identifier.

⁹SECTRACER currently supports ARP spoofing as a concrete example of logical integrity violations. Other MITM-style attacks, such as DNS and IP spoofing, follow similar patterns and can be supported.

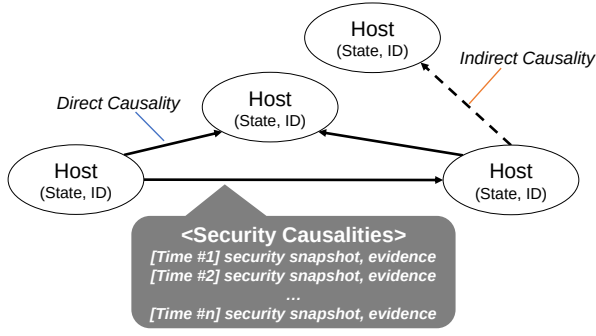


Figure 6: Example security provenance graph, which represents a series of security causalities, from a given host as an input. A vertex indicates a host with the unique identifier and the state. An edge represents security causalities between the hosts with corresponding evidence(s) and their security snapshot(s) in chronological order. A solid line represents a directed causality, and a dotted line represents an indirect causality.

6. Provenance Analysis

The *provenance analyzer* constructs a provenance graph that captures security causalities based on the collected evidence and generates security provenance features.

6.1. Security Provenance Graph

The provenance analyzer constructs a *network-level security provenance graph*, which is a specialized data structure that captures the complete sequence of attack procedures and provides valuable insights into the causes of security incidents across hosts. By analyzing this graph, administrators can effectively understand past security-related events without an extensive manual investigation. It offers a comprehensive view of host activities and operational behaviors, enabling administrators to analyze how an attack was executed and identify its root cause.

Figure 6 shows an example of a security provenance graph. Each vertex represents a host that serves as either a subject or an object within the graph. A vertex has two properties: (i) *state*, which indicates whether the host is a *victim* or an *attacker*, and (ii) *identification (ID)*, which consists of an *IP address* and a *MAC address*, serving as a unique identifier for the host. An edge represents a *security causality* between two hosts and contains security snapshot properties that store system snapshots at a specific time when the corresponding security causality is established. Multiple instances of security causality can occur between the same pair of nodes, resulting in multiple edges in the provenance graph. As each edge embeds temporal information, it can be leveraged for detailed attack reconstruction and forensic analysis.

The provenance analyzer constructs a provenance graph based on the IP address of the host, as specified by the administrator. It retrieves the evidence records in which the packet's IP address corresponds to a given input. Based on the retrieved evidence, it establishes security causalities as edges, connects the corresponding subject and object hosts

Table 3

Enumeration of supported security causalities. Star-notation(*) indicates *indirect causalities*.

Security Causality	Description
#1 connect to	Connection to arbitrary hosts
#2 attack to	Exploitable attack detected
#3 attack trial to	Attack trial detected
#4 attack blocked	Attack (trial) blocked
#5 p_insecure_connect to	Physical integrity violation
#6 l_insecure_connect to	Logical integrity violation
#7 violate integrity of	Subject of integrity violation
#8 violate ac	Access control policy violation
#9 violate sp	Security policy violation
#10 risky_connect to	Connection to an identified attacker
#11 implicit_violate_ac*	Potential access control policy violation
#12 implicit_violate_sp*	Potential security policy violation

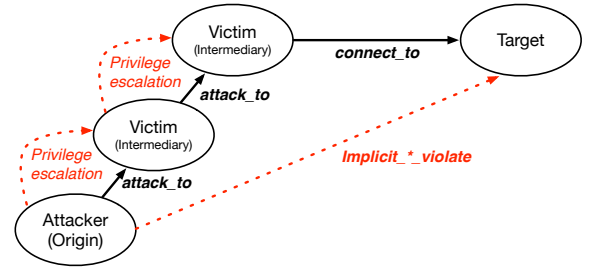


Figure 7: Example of security causality between attacker and target. The dotted line indicates an indirect security causality over intermediary victims.

as vertices, and assigns states to each host according to the defined security causalities (Section 6.2). After updating the security provenance graph, the provenance analyzer selects the next attack source and iteratively expands the graph until no further targets remain.

6.2. Security Causality

Direct and Indirect Causality. Security causality can be categorized into two types: direct and indirect. *Direct causality* is established when strong evidence links an action to its consequences. For example, the attack trials in Figure 1 illustrate the direct causality because they clearly depict the step-by-step progression of an attack. In contrast, *indirect causality* lacks definitive evidences to deterministically establish causality but suggests a potential impact on the target hosts. For instance, in our motivating example, the attacker has indirect causality with the database or web server because access is gained through the manager [1]. In total, SECTRACER supports 12 types of security causalities, as listed in Table 3.

Figure 7 illustrates the indirect causality involving intermediary victims whose privileges have been compromised by attackers. Although *attack_to* does not provide definitive evidence that victims have been fully compromised, the provenance analyzer flags these cases as potential risks, indicating that an attacker has attempted to penetrate the victim's system. The provenance analyzer follows a chain of *attack_to* relationships to identify all

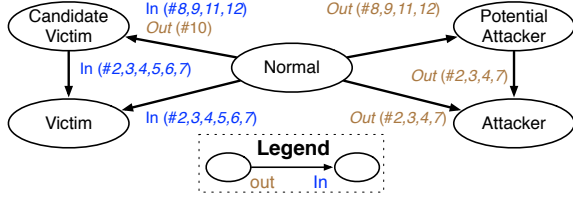


Figure 8: Conditions of state transition. Each number indicates the unique number of each security causality in Table 3.

intermediary victims and considers the leaf node of the chain to have potential access to the victim's associated hosts. If a connection between the leaf node and the victim's associated hosts violates security policies, the provenance analyzer generates *implicit_ac_violate* and *implicit_sp_violate* to indicate implicit access control and security policy violations, respectively. In addition, to warn about risky connections that may lead to further security breaches, it generates *risky_connect_to*, which signifies that a host has established a connection with a known attacker who has already triggered security causalities, leading to a state transition to the *Attacker*.

Vertex State Transition. Each *security causality* dynamically updates the vertex state as the security provenance graph evolves. As shown in Figure 8, the *state* of a node is determined by identified causality relationships. When security causality is detected from one node to another, the state of the affected node changes accordingly. The *Attacker* state indicates that a vertex is actively involved in malicious activities, such as initiating attack trails or causing integrity violations. The *Potential Attacker* state indicates that a vertex possesses the capability to violate security policies, although no explicit malicious behavior has yet been observed. The *Candidate Victim* state represents a host that is either accessible to a (*potential*) *attacker* without proper authorization or has established potentially vulnerable connections with an *attacker*. The *Victim* state is the most critical because it signifies the definitive evidence linking the host to a confirmed attack trial. In a security provenance graph, each node maintains two independent states—one for its role as an *attacker* and another for its role as a *victim*—because a host can act concurrently in both capacities. When a node assumes both the *Attacker* and *Victim* states, this indicates that the node has been exploited as a *stepping stone* that provides valuable context during the subsequent RCA phase.

6.3. Security Provenance Features

The provenance analyzer generates *security provenance features* (*SP features*) to support RCA (Section 7). These features capture various network behaviors, including packet transmissions between hosts, session details, IDS alert messages related to host communications, and the overall network environment status. After constructing the security provenance graph, the provenance analyzer retrieves relevant evidence from the graph and generates the SP features.

Table 4

Security Provenance Feature. Each feature has one or two variables, which represent the network assets (*H*: hosts and *S*: switches).

Category	SP Feature	Numeric	Boolean
Connection	Connection (H1, H2)		✓
	Blocked (H1, H2)		✓
	Number of packets (H1, H2)	✓	
	Packet bytes (H1, H2)	✓	
Session	Sessions (H1, H2)	✓	
	Flag SYN (H1, H2)	✓	
	Flag FIN (H1, H2)	✓	
	Session failure (H1, H2)	✓	
	Number of packets (H1, H2)	✓	
	Packet bytes (H1, H2)	✓	
Security Device	IDS result (H1, H2)	✓	
	Firewall drop (H1, H2)		✓
	IPS drop (H1, H2)		✓
Network Integrity	Physical spoofing (H1, H2)		✓
	Logical spoofing (H1, H2)		✓
	Switch connection (H, S)		✓
	Switch failure (S)		✓
	Link failure (S1, S2)		✓
Policy	AC violation (H1, H2)		✓
	SP violation (H1, H2)		✓
	Blacklist IP (H1, H2)		✓
	Blacklist protocol (H1, H2)		✓
	Blacklist port (H1, H2)		✓
	Blacklist service (H1, H2)		✓
Host	Internal (H)		✓
	Victim (H)		✓

These features are created on demand when an administrator initiates an RCA for a specific host, thereby minimizing the overhead. Table 4 lists the SP features currently supported by SECTRACER, which are categorized into six groups. Additionally, administrators can extend SECTRACER by defining new SP features to enhance RCA capabilities.

- *Connection*: It represents packet transmissions between hosts. This category includes network behaviors such as the success or failure of packet transmissions, packet size, and the number of packets exchanged between two hosts.
- *Session*: It describes TCP sessions between hosts. It includes session-related information such as the number of sessions, session failures, average number of packets per session, and average packet size.
- *Security Device*: It captures alert messages generated by security devices. These messages contain classifications and priority levels, which are converted into SP features.
- *Network Integrity*: It reflects changes in network conditions, such as switch or link failures and host migrations between switches. This category also includes packet integrity issues, such as spoofing or packet modifications.
- *Policy*: It represents network policies configured by administrators. It includes security policies, access control policies, and basic configurations such as allowed ports and protocols.

Table 5

RCA rules categorized by the types of malicious network behaviors that can be detected when the *root cause analyzer* applies these rules. The *weight* of each rule is determined empirically.

Category	Weight	Rules	Results
Basic Rules	1	Victim(A)	Target(A)
	1	Connect(A, B) & Victim(B)	Suspect(A), Target(A)
	1	Connect(A, B) & Suspect(A) & Internal(B)	Target(B)
Security Device Rules	20	Connect(A, B) & IDSResult(A, B) & !Internal(A)	Suspect(A), Target(B)
	20	Connect(A, B) & IDSResult(A, B) & Internal(A)	Suspect(A), Target(B)
	15	Blocked(A, B) & IDSResult(A, B)	Suspect(A), Target(A)
	10	Blocked(A, B) & FirewallDrop(A, B)	Target(B)
	10	Blocked(A, B) & IPSDrop(A, B)	Target(B)
Scanning	5	Connect(A, B) & FlagSYN(A, B) & SessionFailure(A, B)	Suspect(A)
	5	Connect(A, B) & FlagSYN(A, B) & BlacklistPort(A, B)	Suspect(A)
	5	Connect(A, B) & FlagFIN(A, B) & SessionFailure(A, B)	Suspect(A)
	5	Connect(A, B) & FlagFIN(A, B) & BlacklistPort(A, B)	Suspect(A)
Policy Violation	10	Connect(A, B) & ACViolation(A, B)	Suspect(A), Target(B)
	10	Connect(A, B) & SPViolation(A, B)	Suspect(A), Target(B)
Blacklist (external)	5	Connect(A, B) & BlacklistIP(A)	Suspect(A), Target(B)
	5	Connect(A, B) & BlacklistPort(A, B) & !Internal(A)	Suspect(A), Target(B)
	5	Connect(A, B) & BlacklistProtocol(A, B) & !Internal(A)	Suspect(A), Target(B)
	5	Connect(A, B) & BlacklistService(A, B) & !Internal(A)	Suspect(A), Target(B)
Blacklist (internal)	2	Connect(A, B) & BlacklistPort(A, B) & Internal(A)	Suspect(A), Target(B)
	2	Connect(A, B) & BlacklistProtocol(A, B) & Internal(A)	Suspect(A), Target(B)
	2	Connect(A, B) & BlacklistService(A, B) & Internal(A)	Suspect(A), Target(B)
Blacklist (blocked)	2	Blocked(A, B) & BlacklistIP(A)	Suspect(A)
	2	Blocked(A, B) & BlacklistPort(A, B) & !Internal(A)	Suspect(A)
	2	Blocked(A, B) & BlacklistProtocol(A, B) & !Internal(A)	Suspect(A)
	2	Blocked(A, B) & BlacklistService(A, B) & !Internal(A)	Suspect(A)
	2	Blocked(A, B) & BlacklistPort(A, B) & Internal(A)	Suspect(A)
	2	Blocked(A, B) & BlacklistProtocol(A, B) & Internal(A)	Suspect(A)
	2	Blocked(A, B) & BlacklistService(A, B) & Internal(A)	Suspect(A)
Spoofing	25	Connect(A, B) & PhysicalSpoofing(A, B)	Target(A), Target(B)
	25	Connect(A, B) & LogicalSpoofing(A, B)	Target(A), Target(B)
Risky Incident	10	Switch(A, S1) & Switch(A, S2) & (S1!=S2)	RiskyIncident(A)
	100	Switch(A, S1) & Switch(A, S2) & (S1!=S2)	HostMigration(A)
	10	SwitchFailure(S1)	RiskyIncident(S1)
	10	LinkFailure(S1, S2)	RiskyIncident(S1), RiskyIncident(S2)

- *Host*: It describes host characteristics. The *internal* feature identifies hosts belonging to the internal network, whereas the *victim* feature denotes hosts that have been attacked or are suspected of being targeted.

SP features are represented as soft truth values within the interval $[0, 1]$ and are classified into two data types: *numeric* and *Boolean*. The numeric type can assume any value between 0 and 1, whereas the Boolean type is restricted to 0 or 1. The choice of datatype depends on the nature of the feature; the Boolean type indicates the presence or absence of a specific behavior, whereas the numeric type quantifies its magnitude or frequency. When the numerical values exceed the $[0, 1]$ range, an appropriate normalization method is applied to scale them. To ensure that numeric features fell within this range, we computed the average value of the feature in constant time and normalized it accordingly. The transformation follows a predefined normalization equation to map the computed average to a value between 0 and 1. The normalized value of the feature v can be obtained using

$$v = 1 - e^{-ax}, \quad (1)$$

where x represents the average value of the feature, and a is a constant parameter that controls the rate of normalization.

7. Root Cause Analysis

RCA requires administrators to identify the initial attacker, detect infected hosts, and predict the attacker's next target by analyzing collected evidence. This involves extracting complex attack patterns from a provenance graph, a process that is often time-consuming and error-prone. To address this challenge, the *root cause analyzer* leverages PSL [55] to perform RCA by formalizing attack patterns as first-order logic. PSL relies on two key components: a *dataset* representing known relationships and a *ruleset* for inferring new relationships. In RCA, the dataset consists of previously generated SP features (Section 6.3).

The *root cause analyzer* requires *RCA rules*, which are the first-order logic rules used in PSL. The RCA rules are defined as follows:

$$Weight : SP_1 \& SP_2 \& \dots \& SP_n \rightarrow Result, \quad (2)$$

where *Weight* represents the significance of the rule and its impact on the resulting value, *SP* denotes the SP features combined using logical conjunction (&), and *Result* indicates the inferred outcome, which is classified into three categories—*Suspect*, *Target*, and *Risky Incident*—to represent the likelihood of a security event or condition occurring.

- *Suspect*: It represents a host that is either the initial attacker or has performed an attack. A higher suspect

value indicates a greater likelihood that the host initiated or executed an attack on other hosts.

- *Target*: It represents a host that has been infected or is at risk of being compromised. A high target value suggests a strong probability that the host is already infected or vulnerable to an attack.
- *Risky Incident*: It refers to unexpected network changes, such as host migration or failures of links or switches, that pose potential security threats to the network.

The results derived through the RCA represent the probabilistic classification of each node, indicating the likelihood of its role within the attack sequence. In contrast to the *state* assigned during the causality identification phase, which deterministically reflects the node's status based on the observed causal relationships, the categories produced by RCA express the probabilistic characteristics of each node. Specifically, these values estimate the likelihood that a given node is the initial point of compromise or the target of an attack. Notably, during the RCA phase, a single host may simultaneously exhibit high probabilities of being both a *suspect* and *target*, a condition that typically arises when a compromised host is under the attacker's control and is subsequently leveraged to launch further attacks against other hosts.

An example of the RCA rule is expressed as follows:

$$5 : \text{Connect}(A, B) \ \& \ \text{IDSTrojanActivity}(A, B) \ \& \ \text{Victim}(B) \rightarrow \text{Suspect}(A) \quad (3)$$

This rule derives the result by combining multiple SP features, indicating that host *A* is classified as a *suspect* when packet transmission occurs between host *A* and host *B*, which is identified as a Trojan activity, and host *B* is determined to be infected. If host *A* continues to compromise with additional hosts by exhibiting the same behavior, the likelihood of host *A* being a suspect increases. Table 5 summarizes the RCA rules currently supported by SECTRACER. Administrators can also create and modify the RCA rules based on specific security policies and network environments.

8. Evaluation

In this section, we will evaluate SECTRACER for addressing the following key questions:

- **Q1.** Can SECTRACER effectively assist administrators in performing RCA in real-world intrusion scenarios? (Section 8.3)
- **Q2.** What is the performance overhead introduced by SECTRACER's data collection and analysis? (Section 8.5)

8.1. Implementation

We implemented SECTRACER using approximately 10,000 lines of Java code. The provenance collector (Section 5) was built using ONOS v1.8.4 [49], which supports OpenFlow v1.3. Implemented as an ONOS application, it continuously monitors network changes, including flow rule creation/deletion, host migration, and topology modifications, to maintain updated network snapshots. For packet monitoring, dedicated monitoring flow rules are installed in a separate flow table to prevent interference with the existing flow rule tables used by other ONOS applications. This design allows network administrators to run any SDN application, including ONOS applications, without requiring modifications to integrate them with SECTRACER.

A provenance analyzer (Section 6) was implemented as a standalone application. The implementation leveraged several open-source software and libraries: OrientDB v2.2.23 [56] as the database, Blueprints v2.6 [57] for the database interface, Pcap4j v1.7 [58] for packet capture, Kryonet v2.21 [59] as a NIO server/client network library, Snort v2.9.9 [51] for intrusion detection and prevention, and Syslog4j v1.7 [60] as a syslog server/client library. The root cause analyzer (Section 7) was implemented using PSL [31], a framework designed for developing probabilistic models. The root cause analyzer translates the RCA rules into PSL rules and integrates the collected provenance data to perform root cause analysis.

8.2. Experimental Setup

We performed our experiments on an Intel Hexa-core Xeon E5-1650 machine with 64GB of RAM using Mininet [61] to emulate the network topologies. For the performance evaluation, we utilized two Intel Quad-core i5-6600K machines with 16GB of RAM and four hardware SDN switches, consisting of two Pica8 P3290 switches and two Pica8 P3297 switches. Each machine ran on Ubuntu. The Xeon machine hosted LXD containers to execute SECTRACER component instances, including OrientDB, Snort, and ONOS. We extended ONOS's default forwarding application to support security forwarding, enabling the integration of security middleboxes along forwarding paths, as well as access control functionalities. Malicious payloads such as remote shell exploits and botnet messages were synthetically generated with signatures provided to the IDS (Snort) for detection and analysis.

8.3. Attack Scenario Case Studies

Here, we will evaluate the effectiveness of SECTRACER under two attack scenarios: (i) APT and (ii) botnet propagation.

APT. To evaluate the effectiveness of SECTRACER, we demonstrate a scenario involving an APT, which is a prevalent attack in modern enterprise networks. APTs typically involve sophisticated techniques for infiltrating and compromising protected internal resources [62]. For this demonstration, we present our motivating example—an information leakage attack described in Section 2.1—as the APT

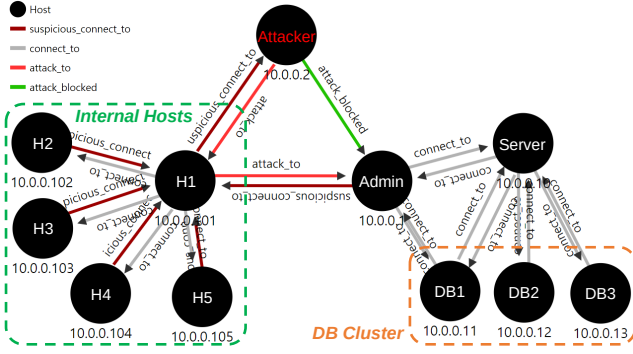


Figure 9: Security provenance graph of the APT attack scenario in Figure 1. Some of the security causalities are omitted for visibility.

scenario to showcase SECTRACER’s ability to construct a SP graph and perform RCA. We developed a script that generates the network topology shown in Figure 1 using Mininet [61] and executed the exact attack sequences. Upon running the script, SECTRACER collected and recorded the provenance data in its database and constructed a SP graph starting from the database server.

Figure 9 presents the SP graph generated using SECTRACER. The process begins by SECTRACER retrieving evidence from one of the databases (DB1), identified by its IP address (10.0.0.11), to establish security causalities with other machines. DB1 has a *connect_to* relationship with the administrator machine, which in turn connects to the server for management. The server also maintains *connect_to* relationships with all the databases for operational purposes. The attacker host initially attempted to deliver remote exploitation to the administrator machine; however, the firewall successfully blocked this attempt. Consequently, SECTRACER created an *attack_blocked* relationship between the attacker and the administrator machine, marking the attacker host as an *attacker*. However, the internal host (H1) subsequently succeeded in exploiting the administrator machine. Consequently, SECTRACER established an *attack_to* relationship between H1 and the administrator machine, classifying H1 as an *attacker*. This analysis reveals that the attack propagated through four internal hosts, traversing from the attacker to H1, then to the administrator, the server, and finally to the database.

Among the newly generated security causalities, *attack_to* has the highest priority, prompting SECTRACER to select H1 as the next attack source (see Section 6.1) and iteratively uncover additional security causalities. SECTRACER further revealed that H1 conducted a port scan on other internal hosts, leading to the identification of eight security causalities, including *connect_to* and *suspicious_connect_to*, linking H1 to internal hosts H2, H3, H4, and H5. In addition, two additional security causalities—*attack_to* and *suspicious_connect_to*—were identified between the attacker and H1, confirming that H1 was initially compromised by the attacker.

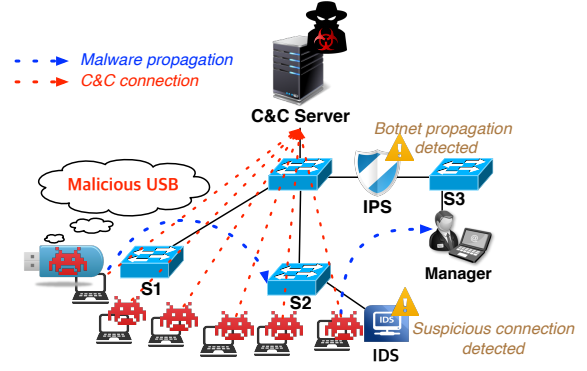


Figure 10: Scenario of enterprise botnet propagation by malware injection. One of the compromised hosts attempts to initiate access to the manager machine.

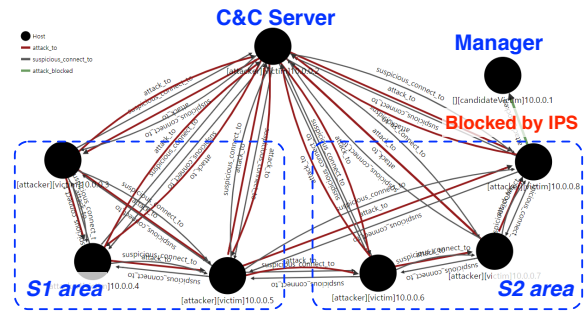


Figure 11: SECTRACER reveals the trace of botnet pandemic from its threat propagation analysis.

Botnet Propagation. Network attacks originating from internal hosts within an enterprise are challenging to detect and mitigate using conventional perimeter defense mechanisms. Although security middleboxes often generate frequent alerts, accurately assessing their relevance and severity remains a complex task. To evaluate the effectiveness of SECTRACER in this context, we consider an enterprise attack scenario in which a botnet is propagated from an internal host that is inadvertently infected via a USB drive containing malware. Figure 10 illustrates a scenario in which the security manager receives alerts of blocked botnet propagation from an internal IPS and suspicious C&C communication attempts detected by an internal IDS. Despite these alerts being generated by security systems, security administrators typically lack the contextual information required to accurately identify the attack origin or assess the security impact of such alerts.

Figure 11 shows the attack-history graph generated using SECTRACER. The botnet structure, consisting of a C&C server and six infected hosts, emerges as a strongly connected component within the graph because of the proliferation of *attack_to* and *suspicious_connect_to* edges derived from the collected attack evidence. Although the network traffic among hosts is complex and highly intertwined, the graph clearly shows that all communication paths converge toward the external C&C server. Furthermore, it captures events in which a compromised host attempts to access the

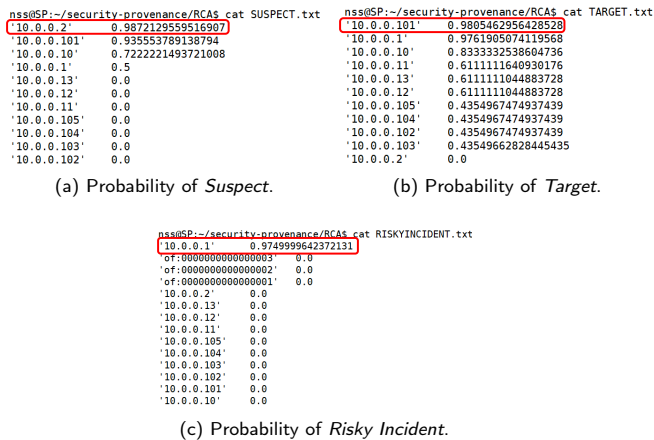


Figure 12: Results of applying RCA rules to the security provenance graph shown in Figure 9.

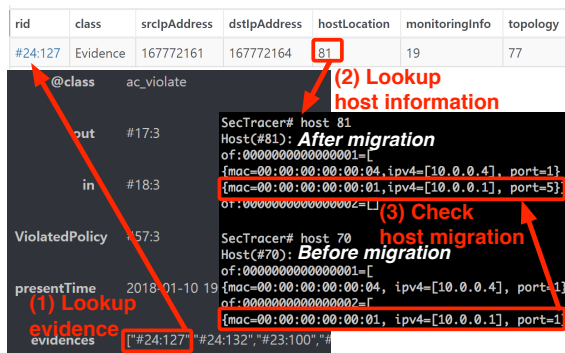


Figure 13: Example of manual inspection with the *ac_violate* security causality evidences.

manager’s machine during an attack. This capability enables SECTRACER to effectively visualize internal attacks that are otherwise difficult to detect, thereby providing a clear and comprehensive representation of attack propagation at the network level. In addition, the temporal information embedded in the edges can be leveraged to perform a RCA for deeper forensic investigations.

8.4. Effectiveness of Root Cause Analysis

We evaluated the RCA capability of SECTRACER for the two attacks demonstrated.

APT. After constructing the SP graph, SECTRACER performed a RCA using PSL. The root cause analyzer (Section 7) extracted SP features from the constructed provenance graph shown in Figure 9 and applied RCA rules to compute the probability of *Suspect*, *Target*, and *Risky incident* for the attacker host (10.0.0.2), administrator host (10.0.0.1), and internal host H1 (10.0.0.101). In this experiment, all the database servers in the DB cluster (10.0.0.11, 10.0.0.12, and 10.0.0.13) were designated as *victim* in the SP features (Table 4).

To infer the *Suspect* probability, Figure 12a shows that the attacker host (10.0.0.2) and the internal host H1 (10.0.0.101) have the highest suspicion levels, with

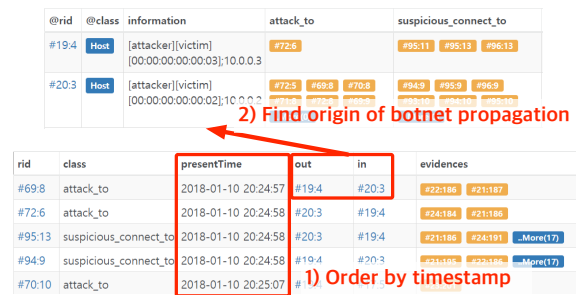


Figure 14: Chronological order of generated security relationships in SECTRACER dashboard GUI. The head of the list corresponds to the attack origin.

probabilities of 98.7% and 93.6%, respectively. Additionally, the server (10.0.0.10) and administrator machine (10.0.0.1) are identified as suspicious, with probabilities of 72% and 50%, respectively. This is because both machines have evidence marked as *reached* on their database servers, which are designated as *victim*. Notably, server (10.0.0.10) has a higher suspect value than the administrator machine (10.0.0.1) because of its access to a greater number of database servers, which resulted in more causalities.

To infer the *Target* probability, Figure 12b shows that the internal host H1 (10.0.0.101) and administrator machine (10.0.0.1) have the highest probabilities of being targeted at 98% and 97.6%, respectively. In addition, server (10.0.0.10) has a high probability (83.3%) because it is accessible to all database servers, making it a likely target for attackers. All database servers (10.0.0.11, 10.0.0.12, and 10.0.0.13) have a 61.1% probability of being targeted because they are designated as *victim*. Furthermore, all other internal hosts (10.0.0.102, 10.0.0.103, 10.0.0.104, and 10.0.0.105) have a 43.5% probability because they were scanned by H1 but did not experience further malicious activity.

Finally, to infer the *Risky Incident* probability, Figure 12c shows that the administrator machine (10.0.0.1) has the highest probability of 97.5%. An in-depth analysis was conducted to determine this cause. To support administrator inspection, SECTRACER provides both a GUI dashboard and a CLI tool. Figure 13 illustrates this process, which enables administrators to perform RCA effectively. During the investigation, an administrator detected an `ac_violation` causality originating from the SDN switch with DPID *of:0000000000000001*. However, further analysis revealed that the machine's current location was at a different SDN switch from the DPID *of:0000000000000002*, confirming that a host has migrated. This migration event could lead to access-control policy violations, thereby exposing potential security risks.

Botnet Propagation. In the botnet propagation scenario, identifying the root cause is comparatively more straightforward than in the APT scenario. As all connections in a botnet communicate directly with the C&C server, the root cause can often be determined without performing a

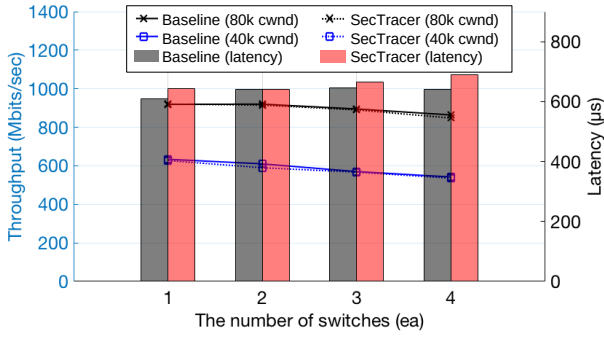


Figure 15: End-to-end latency and throughput with/without SEC TRACER.

full RCA, which relies solely on attack forensics to pinpoint the origin. The origin of the attack can be inferred from the chronological ordering of the identified security relationships. Figure 14 shows the SEC TRACER dashboard GUI, which lists security relationships in timestamp order. Tracing the initial entry in this sequence reveals the primary point of compromise, namely, the host infected via the malicious USB drive.

8.5. Performance Benchmarks

We evaluated SEC TRACER in terms of system overhead and performance. To assess the overhead of SEC TRACER’s provenance collection, we used `iperf3` [63] and `ping` [64] to measure its impact on network performance. Additionally, we assessed the processing time of the submodules in evidence generation and graph construction to comprehend their performance bottlenecks.

Network Overhead of SEC TRACER. We evaluated the end-to-end latency and throughput with and without provenance collection of SEC TRACER to assess its overhead. Our experimental environment consisted of two Intel i5 machines and four Pica8 SDN switches, and we configured a linear topology by incrementally adding the switches. Forwarding rules were installed to enable packet delivery between the two hosts, whereas monitoring rules were issued by SEC TRACER to the edge switches (or a single switch). We used `iPerf3` to measure TCP throughput with varying window sizes and `ping` to evaluate the latency. Figure 15 shows the evaluation results. As the number of switches increases, the throughput decreases slightly in both the baseline and SEC TRACER cases. However, SEC TRACER introduces less than 1% overhead when using TCP window sizes (cwnd) of 40k and 80k. In terms of latency, SEC TRACER exhibits negligible overhead compared with the baseline, demonstrating its efficiency in provenance collection without significant performance degradation.

Microbenchmark of Evidence Generation. To evaluate the performance overhead of provenance collection, we measured the processing time of each submodule in the evidence generation process of our tool, including integrity, reachability, and policy violation checks (Section 5.3).

Table 6

Elapsed time (μs) of the evidence generation. The star-notation (*) indicates cached results. (I: Integrity, R: Reachability, PV: Policy Violation, SP: Security Policy, SD: Security Device)

	I	R	PV I	PV II	SD	Total
<i>Benign</i>	2.50	4.45	134.31 (0.03)*	127.49 (0.14)*	-	268.75
<i>Malicious</i>	2.50	4.45	134.31 (0.03)*	127.49 (0.14)*	811.58	1,080.33

Table 6 presents the microbenchmark results for both benign and malicious cases. In both scenarios, integrity (I) and reachability (R) were completed within 5 μs . We assessed two cases of policy violation detection: (i) deploying 10 security routing rules (PV I) and (ii) deploying 10 access control rules (PV II). Checking for PV I and PV II violations requires 134.31 μs and 127.49 μs , respectively, as the processing time increases linearly with the number of rules. The primary bottleneck stems from SEC TRACER retrieving provenance data from a database through rule matching, which can be optimized by caching frequently accessed rules. With caching, the processing times for PV I and PV II are reduced to 0.03 μs and 0.14 μs , respectively, achieving a 99.9% reduction. In addition, we evaluated the processing time when SEC TRACER relies on existing security devices (SD), which requires parsing IDS logs to obtain evidence. Because security devices generate logs only upon detecting malicious activity, we measured this case separately. The results indicate that leveraging security devices accounts for over 75% of the total processing time because of the string parsing overhead required to interpret IDS logs.

Microbenchmark of Graph Construction. We evaluated the graph construction performance by measuring the processing time of each submodule. Using a similar experimental setup (i.e., an APT attack) on a Xeon machine running OrientDB and Snort, we obtained the results listed in Table 7. Overall, SEC TRACER completed the graph construction in approximately 31 s. It generated 32,155 pieces of evidence (EG) in approximately 1.57 s, accounting for 4.9% of the total processing time. In addition, SEC TRACER produced 32,155 security causalities (SR) in only 0.073 s and performed 32,114 state-transition operations (ST) in 0.006 s. The most time-consuming step was the database operation (DB Ops), which involved 39 read/write executions to retrieve packets and store vertices/edges for the SP graph. These operations accounted for approximately 94.75% of the total processing time. To enhance the performance of SEC TRACER, we plan to replace OrientDB with a high-performance database and leverage distributed computing environments [65] to efficiently distribute workloads.

Table 7

Processing time for the attack graph of the APT attack scenario. (EG: Evidence Generation, SC: Security Causality, ST: State Transition, DB Ops: DB Operations)

	EG	SC	ST	DB Ops
# of Operations	32,155	32,155	32,114	39
Per-Operation Time (μ s)	48.75	2.27	0.19	763,272.21
Total Time (s)	1.569	0.073	0.006	29.767
Ratio (%)	4.994	0.232	0.019	94.754

9. Discussion and Limitations

In this section, we discuss the additional design considerations and limitations of SECTRACER.

Real-time Attack Detection and Response. Because SECTRACER is primarily designed for forensic analysis to uncover the root causes of APTs in enterprise networks, its real-time detection and response capabilities are limited. However, this limitation can be mitigated by integrating traditional firewalls or NIPS to deploy filtering rules for suspicious hosts identified through RCA. Furthermore, in SDN-enabled networks—as assumed in Section 2.3—such responses can be efficiently implemented by installing blocking rules directly on SDN switches.

Storage Optimization for Security Logging. A security snapshot is generated only when changes occur in the corresponding policies or network configurations. As these elements do not change frequently, the overall size of the logging data remains relatively small. Furthermore, evidence records reference only the ID of the most recent snapshot, rather than the full snapshot, ensuring that data size does not become a concern. In addition, employing a delta-based storage approach, where only the differences between snapshots are stored, can further minimize the volume of logging data. Similarly, network packets can be aggregated using an optimized process, where redundant packets are deduplicated and only changes or differences are recorded before storage. Beyond these methods, various network log retention strategies can be applied to optimize storage efficiency and further reduce the size of the logging dataset.

Temporal-aware Causality in the Graph. Although SECTRACER embeds timestamps in security causality edges, as shown in Figure 6, it does not enforce the temporal order between vertices. This implies that an edge is created between entities when an incident occurs, regardless of their chronological order. To reconstruct an attack timeline, administrators must refer to the timestamps embedded in each causality edge. Although this design requires additional effort from administrators, it is motivated by the need to avoid the *dependency explosion* problem, where the provenance graph grows significantly in size if full temporal ordering among vertices is enforced. Thus, our approach prioritizes graph efficiency while preserving temporal

information within the edges.

False Positives in RCA. Despite our tamper-resistant provenance collection (Section 5.3), SECTRACER may incorrectly identify the root causes owing to inaccurate provenance data or labeling errors. Such inaccuracies stem from the use of the indirect causality mechanisms described in Section 6.2, including `risky_connect_to`, `implicit_violate_ac`, and `implicit_violate_sp`, which infer potential causality rather than rely on concrete evidence. However, the goal of SECTRACER is not to guarantee precise causal inference but rather to highlight potential threats based on predefined rules, enabling administrators to initiate manual investigations more efficiently. This approach helps reduce the burden of analyzing complex provenance graphs.

Comparison with Host-level Provenance Graphs. SECTRACER focuses on network-level SP to identify the root causes of APTs that span multiple hosts. Consequently, it does not provide the process-, user-, or file-level granularity offered by traditional host-level provenance graphs. We argue that RCA at the network level is critical in enterprise environments. Nevertheless, to track intrahost activities, SECTRACER can be integrated with host-level provenance systems to provide more fine-grained visibility. In this sense, SECTRACER is complementary to, rather than a replacement for, host-level provenance graphs.

10. Conclusion

Network intrusions in enterprise networks are becoming increasingly sophisticated, making their detection and analysis challenging. To address this problem, we introduce SECTRACER, a novel network-level provenance-based intrusion detection system that enables administrators to gain a clear understanding of operational scenarios and attack origins. SECTRACER systematically collects provenance data, constructs provenance graphs, and performs RCA using probabilistic models. Our evaluation demonstrates that SECTRACER effectively tracks the detailed operations of APT attacks and accurately identifies their root causes with minimal system overhead. These findings suggest that our approach is well-suited to securing modern network infrastructures, even in complex environments.

References

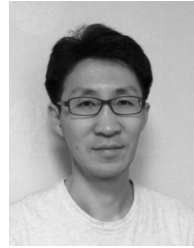
- [1] Yin Zhang and Vern Paxson. Detecting stepping stones. In *USENIX Security Symposium*, volume 171, page 184, 2000.
- [2] Richard Zuech, Taghi M Khoshgofaar, and Randall Wald. Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data*, 2(1):3, 2015.
- [3] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1), 2014.
- [4] Dana Larson. How to choose between outsourced vs in-house cybersecurity. <https://www.crowdstrike.com/en-us/cybersecurity-y-101/small-business/in-house-vs-outsourced-cybersecurity/>.

- [5] Devin J Pohly, Stephen McLaughlin, Patrick McDaniel, and Kevin Butler. Hi-fi: collecting high-fidelity whole-system provenance. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 259–268. ACM, 2012.
- [6] Ashish Gehani and Dawood Tariq. Spade: support for provenance auditing in distributed environments. In *Proceedings of the 13th International Middleware Conference*, pages 101–120. Springer-Verlag New York, Inc., 2012.
- [7] Adam M Bates, Dave Tian, Kevin RB Butler, and Thomas Moyer. Trustworthy whole-system provenance for the linux kernel. In *USENIX Security Symposium*, pages 319–334, 2015.
- [8] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eysers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 405–418, 2017.
- [9] Xutong Chen, Hassaan Irshad, Yan Chen, Ashish Gehani, and Vinod Yegneswaran. Clarion: Sound and clear provenance tracking for microservice deployments. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3989–4006, 2021.
- [10] Pubali Datta, Isaac Polinsky, Muhammad Adil Inam, Adam Bates, and William Enck. Alastor: Reconstructing the provenance of serverless intrusions. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2443–2460, 2022.
- [11] Azadeh Tabiban, Heyang Zhao, Yosr Jarraya, Makan Pourzandi, Mengyuan Zhang, and Lingyu Wang. Provtalk: Towards interpretable multi-level provenance analysis in networking functions virtualization (nfv). In *NDSS*, 2022.
- [12] Hassaan Irshad, Gabriela Ciocarlie, Ashish Gehani, Vinod Yegneswaran, Kyu Hyung Lee, Jignesh Patel, Somesh Jha, Yonghui Kwon, Dongyan Xu, and Xiangyu Zhang. Trace: Enterprise-wide provenance tracking for real-time apt detection. *IEEE Transactions on Information Forensics and Security*, 16:4363–4376, 2021.
- [13] Feng Dong, Shaofei Li, Peng Jiang, Ding Li, Haoyu Wang, Liangyi Huang, Xusheng Xiao, Jiedong Chen, Xiapu Luo, Yao Guo, et al. Are we there yet? an industrial viewpoint on provenance-based endpoint detection and response tools. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2396–2410, 2023.
- [14] R Sekar, Hanke Kimm, and Rohit Aich. eaudit: A fast, scalable and deployable audit data collection system. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3571–3589. IEEE, 2024.
- [15] Md Nahid Hossain, Sadegh M Milajerdi, Junao Wang, Birhanu Eshete, Rigel Gjomemo, R Sekar, Scott Stoller, and VN Venkatakrishnan. {SLEUTH}: Real-time attack scenario reconstruction from {COTS} audit data. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 487–504, 2017.
- [16] Sadegh M Milajerdi, Rigel Gjomemo, Birhanu Eshete, Ramachandran Sekar, and VN Venkatakrishnan. Holmes: real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1137–1152. IEEE, 2019.
- [17] 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)*, pages 1139–1155. IEEE, 2020.
- [18] Wajih Ul Hassan, Adam Bates, and Daniel Marino. Tactical provenance analysis for endpoint detection and response systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1172–1189. IEEE, 2020.
- [19] Sadegh M Milajerdi, Birhanu Eshete, Rigel Gjomemo, and VN Venkatakrishnan. Poirot: Aligning attack behavior with kernel audit records for cyber threat hunting. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1795–1812, 2019.
- [20] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. In *network and distributed systems security symposium*, 2019.
- [21] Xueyuan Han, Thomas Pasquier, Adam Bates, James Mickens, and Margo Seltzer. Unicorn: Runtime provenance-based detector for advanced persistent threats. *arXiv preprint arXiv:2001.01525*, 2020.
- [22] Qi Wang, Wajih Ul Hassan, Ding Li, Kangkook Jee, Xiao Yu, Kexuan Zou, Junghwan Rhee, Zhengzhang Chen, Wei Cheng, Carl A Gunter, et al. You are what you do: Hunting stealthy malware via data provenance analysis. In *NDSS*, 2020.
- [23] Abdullah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z Berkay Celik, Xiangyu Zhang, and Dongyan Xu. Atlas: A sequence-based learning approach for attack investigation. In *Proceedings of the 30th USENIX security symposium (USENIX security 21)*, pages 3005–3022, 2021.
- [24] Su Wang, Zhiliang Wang, Tao Zhou, Hongbin Sun, Xia Yin, Dongqi Han, Han Zhang, Xingang Shi, and Jiahai Yang. Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning. *IEEE Transactions on Information Forensics and Security*, 17:3972–3987, 2022.
- [25] Jun Zengy, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *2022 IEEE symposium on security and privacy (SP)*, pages 489–506. IEEE, 2022.
- [26] Fan Yang, Jiachen Xu, Chunlin Xiong, Zhou Li, and Kehuan Zhang. Prographer: An anomaly detection system based on provenance graph embedding. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23)*, pages 4355–4372, 2023.
- [27] Zian Jia, Yun Xiong, Yuhong Nan, Yao Zhang, Jinjing Zhao, and Mi Wen. Magic: Detecting advanced persistent threats via masked graph representation learning. In *Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24)*, pages 5197–5214, 2024.
- [28] Akul Goyal, Gang Wang, and Adam Bates. R-caid: Embedding root cause analysis within provenance-based intrusion detection. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3515–3532. IEEE, 2024.
- [29] Zijun Cheng, Qiuqian Lv, Jinyuan Liang, Yan Wang, Degang Sun, Thomas Pasquier, and Xueyuan Han. Kairos: Practical intrusion detection and investigation using whole-system provenance. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*, pages 3533–3551. IEEE, 2024.
- [30] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*, pages 139–139. IEEE Computer Society, 2024.
- [31] Psl. <https://github.com/linqs/psl>.
- [32] MITRE ATT&CK. <https://attack.mitre.org/>.
- [33] Seyed K Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. Bohatei: flexible and elastic ddos defense. In *Proceedings of the 24th USENIX Conference on Security Symposium*, pages 817–832. USENIX Association, 2015.
- [34] Seunghyeon Lee, Jinwoo Kim, Seungwon Shin, Phillip Porras, and Vinod Yegneswaran. Athena: A framework for scalable anomaly detection in software-defined networks. In *Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on*. IEEE, 2017.
- [35] Nima Zahadat, P Blessner, T Blackburn, and Bill A. Olson. Byod security engineering: A framework and its analysis. *Comput. Secur.*, 55:81–99, 2015.
- [36] Trend Micro. 70% of soc teams emotionally overwhelmed by security alert volume, 2021. <https://newsroom.trendmicro.com/2021-05-25-70-Of-SOC-Teams-Emotionally-Overwhelmed-By-Security-Alert-Volum>e.
- [37] Snort. Snort 3 rule writing guide. <https://docs.snort.org/rules/>.
- [38] NIST. Vulnerability metrics. <https://nvd.nist.gov/vuln-metrics/cvss>.
- [39] Haopei Wang, Guangliang Yang, Phakpoom Chinprutthiwong, Lei Xu, Yangyong Zhang, and Guofei Gu. Towards fine-grained network security forensics and diagnosis in the sdn era. In *Proceedings of*

- the 2018 ACM SIGSAC conference on computer and communications security, pages 3–16, 2018.
- [40] Benjamin E Ujich, Samuel Jero, Anne Edmundson, Qi Wang, Richard Skowrya, James Landry, Adam Bates, William H Sanders, Cristina Nita-Rotaru, and Hamed Okhravi. Cross-app poisoning in software-defined networking. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 648–663, 2018.
 - [41] Osama Bajaber, Bo Ji, and Peng Gao. P4control: Line-rate cross-host attack prevention via in-network information flow control enabled by programmable switches and ebpf. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 147–147. IEEE Computer Society, 2024.
 - [42] TJ OConnor, William Enck, W Michael Petullo, and Akash Verma. Pivotwall: Sdn-based information flow control. In *Proceedings of the Symposium on SDN Research*, pages 1–14, 2018.
 - [43] Seungwon Shin, Lei Xu, Sungmin Hong, and Guofei Gu. Enhancing network security through software defined networking (sdn). In *2016 25th international conference on computer communication and networks (ICCCN)*, pages 1–9. IEEE, 2016.
 - [44] Jinwoo Kim, Minjae Seo, Seungsoo Lee, Jaehyun Nam, Vinod Yegneswaran, Phillip Porras, Guofei Gu, and Seungwon Shin. Enhancing security in sdn: Systematizing attacks and defenses from a penetration perspective. *Computer Networks*, 241:110203, 2024.
 - [45] Angelika Kimmig, Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pages 1–4, 2012.
 - [46] Open Networking Foundation. Openflow switch specification version 1.3.0. <https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf>.
 - [47] Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P Brighten Godfrey. Veriflow: Verifying network-wide invariants in real time. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI 2013)*.
 - [48] Peyman Kazemian, George Varghese, and Nick McKeown. Header space analysis: Static checking for networks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI 2012)*.
 - [49] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. ONOS: towards an open, distributed SDN OS. In *Proceedings of ACM HotSDN 2014*.
 - [50] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *2014 IEEE 15th International Symposium on WoWMoM*, 2014.
 - [51] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, 1999.
 - [52] Suricata. Suricata open source ids / ips / nsm engine. <https://suricata.io/>.
 - [53] Seungwon Shin and Guofei Gu. Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In *IEEE International Conference on Network Protocols (ICNP)*, 2012.
 - [54] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazières, and Nick McKeown. I know what your packet did last hop: Using packet histories to troubleshoot networks. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 71–85, 2014.
 - [55] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research (JMLR)*, 2017.
 - [56] Orient db. <https://orientdb.org/>.
 - [57] Blueprints. <https://github.com/tinkerpop/blueprints/>.
 - [58] Pcap4j: A java library for capturing, crafting, and sending packets. <https://www.pcap4j.org/>.
 - [59] Kryonet. <https://github.com/EsotericSoftware/kryonet>.
 - [60] Syslog4j: Complete syslog implementation for java. <http://www.syslog4j.org/>.
 - [61] Mininet: An instant virtual network on your laptop (or other pc). <http://mininet.org/>.
 - [62] Symantec. Advanced persistent threats: A symantec perspective. In *White Paper*, 2011.
 - [63] iperf - the ultimate speed test tool for tcp, udp and sctp. <https://iperf.fr/>.
 - [64] ping(8) Linux man page. <https://linux.die.net/man/8/ping>.
 - [65] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 2010.



Seunghyeon Lee is the Product Director of the AI Product Department and Co-Founder at S2W. He received his Ph.D. degree in Information Security from KAIST. His research interests include ontology modeling, knowledge graph construction, and intelligent information systems. His current work focuses on applying these research areas to develop AI products that address complex real-world problems.



Seungwon Shin is an Associate Professor in the School of Electrical Engineering at KAIST and an Executive Vice President at Samsung Electronics. He received his Ph.D. degree in Computer Engineering from the Electrical and Computer Engineering Department, Texas A&M University, and his M.S. and B. S. degrees from KAIST, both in Electrical and Computer Engineering. His research interests include software-defined networking security, dark web analysis, and cyber threat intelligence.



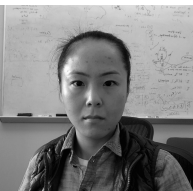
Hyunmin Seo is a Ph.D. student in the School of Electrical Engineering at KAIST. He received a B.S. degree in Electrical Engineering from KAIST. He received his M.S. degree in Electrical Engineering from KAIST. His research interests include programmable network data planes, cyberattacks, and cloud security.



Jinwoo Kim is an Assistant Professor in the School of Software at Kwangwoon University, Seoul, Republic of Korea. He received his Ph.D. degree from the School of Electrical Engineering at KAIST, his M.S. degree from the Graduate School of Information Security at KAIST, and his B.S. degree from Chungnam National University in Computer Science and Engineering. His research focuses on security and privacy issues with software-defined networks, cloud systems, and VR.



Hwangjo Heo is a Principal Researcher at the Electronics and Telecommunications Research Institute (ETRI). He earned his Ph.D from KAIST and his Master's degree from Purdue University. His research interests include the security of computer systems, encompassing blockchain, computer networks, and artificial intelligence systems.



Anduo Wang is an Associate Professor at Temple University, where she specializes in improving network state management using formal methods, databases, knowledge representation and reasoning, and logic programming. She received her Ph.D. from the University of Pennsylvania in 2013.