DOS Classification, and Virtual Machine Detection via Passive Monitoring through Security Twins

Vincenzo Sammartino
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
vincenzo.sammartino@phd. unipi.it

Fabrizio Baiardi
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
f.baiardi@unipi.it

Salvatore Ruggieri
Dipartimento di Informatica
Università di Pisa
Pisa, Italy
salvatore.ruggieri@unipi.it

***Abstract*—The security twin (ST) paradigm is an emerging framework for dynamically modeling and defending complex cyber-physical systems. This paper extends NotLine, a fully automated, non-intrusive ST construction platform with three mechanisms. First, a lightweight statistical engine based on the numerically stable Welford algorithm maintains per-host behavioral baselines in $\mathcal{O}(1)$ time and space, enabling real-time Z-score standardization and denial-of-service classification across volumetric, resource-exhaustion, and application-layer attack categories via a similarity matrix framework. Second, a passive methodology identifies virtual machines by correlating deep packet inspection fingerprinting inconsistencies with SNMP-retrieved switch CAM table data, resolving both NAT and bridged VM topologies. Third, both mechanisms integrate into a unified, continuously updated ST that models structural topology, bandwidth saturation, and resource exhaustion within a single non-intrusive pipeline. Evaluations on ns-3 simulations and a physical testbed confirm zero misclassification across all three attack vectors and complete topology reconstruction, with negligible overhead compatible with real-time deployment.**

Keywords: Security Twin, Digital Twin, Passive Network Monitoring, Anomaly Detection, Denial-of-Service Classification, Virtual Machine Identification, Welford Algorithm, SNMP, Deep Packet Inspection, Z-Score.

I. Introduction

Modern networks and cyber-physical systems (CPS) are characterized by extreme heterogeneity of devices, services, and communication protocols. The convergence of information technology (IT) and operational technology (OT) has exponentially expanded the attack surface, rendering static defence mechanisms largely ineffective against automated, adaptive, and rapidly evolving threats. The ability to model and predict intrusions is therefore a fundamental prerequisite for designing scalable and proactive defense architectures.

a) Problem Statement: Current security monitoring frameworks suffer from three limitations in complex CPS environments. First, they apply active scanning and probing to maintain inventory accuracy. This is unsafe in production OT environments where injecting synthetic traffic can perturb legacy devices, trigger false alarms, or induce unexpected physical state transitions. Furthermore, they treat anomaly detection and topology modeling as decoupled concerns, failing to exploit the structural information embedded in network traffic for mutual enrichment. Lastly, they exhibit a critical blind spot with respect to virtualization: a virtual machine (VM) generates a network stack that is, at the protocol level, indistinguishable from a physical host, causing topology models to misrepresent the true hardware configuration.

b) Security Twin Paradigm: We address these limitations through the security twin (ST) paradigm. An ST is a continuously updated model of an ICT/OT infrastructure that maintains a structured inventory of hardware nodes, software modules, routing policies, and vulnerabilities, expressed as a directed graph where nodes model hosts and edges network flows. Formally, the ST is a tuple $\mathcal{ST} = (H, E, V, \mathcal{P})$, where H is the set of hosts, $E \subseteq H \times H$ is the set of directed communication flows, $V : H \to 2^{\text{CVE}}$ is a function mapping each host to its set of known vulnerabilities, and $\mathcal{P} : H \times H \to [0, 1]$ assigns conditional exploitation probabilities to edges. The ST serves as the substrate for an engine (SE) that estimates intrusion success rates and time-to-compromise by simulating thousands of intrusions with parallel attack path traversals, without interacting with the production system. The accuracy of the simulations requires that the ST is built in a continuous, fully automated mmanner by adapting the model to configuration changes, newly disclosed vulnerabilities, and structural topology shifts. To be non-intrusive, this builder relies exclusively on passive traffic observation.

c) Contributions: This paper presents three primary technical contributions extending NotLine [1], [2], a non-intrusive ST construction platform:

1) **Real-Time DoS Classification Engine**: A constant-time statistical engine based on the Welford online algorithm that incrementally maintains per-host behavioral baselines and classifies volumetric, resource-exhaustion, and application-layer denial-of-service (DoS) attacks via a formal similarity matrix framework, without requiring stateful flow tracking or deep payload inspection.
2) **Passive VM Topology Reconstruction**: The ST builder resolves obfuscated hardware topologies with high fidelity through a dual-mode methodology that identifies NAT-configured VMs through DPI fingerprinting inconsistency analysis and bridged VMs through SNMP-based polling of switch CAM tables.
3) **Unified Performance Model**: An integration of the above mechanisms into the ST builder that produces a

deterministic, continuously updated performance model of the monitored infrastructure, unifying structural topology modeling and behavioral anomaly detection within a single passive, non-intrusive framework.

d) Paper Organization: Section II surveys the state of the art in network intrusion modeling, flow-based passive monitoring, and digital twin construction. Section III formalizes the statistical anomaly classification model. Section IV describes the passive VM identification methodology. Section V presents the experimental evaluation. Section VI concludes the paper and outlines future directions.

II. Background and Related Work

The continuous construction and maintenance of a dynamic ST relies on the extraction of high-fidelity behavioral data from the underlying network [3]. This section establishes the theoretical foundations of modern network threats, the mechanics of passive flow analysis, and the architectural components of an ST, and situates our work within the existing literature.

A. Network Intrusions and Attack Graphs

An intrusion against a CPS is formally modeled as a sequence of discrete adversarial actions against a target, where each action may be an attack, but also collect information on the target system or exfiltrate information. In general, an action is characterized by a source address, a destination address, a set of preconditions over the attacker state, and a set of resulting postconditions [4], [5]. An attacker state is a pair of access rights and information on the target system. These sequences are mapped into an intrusion graph, where nodes represent the attacker state, and directed edges are labeled by the actions firing the actor state transitions. Failures are self-looping to a node. In the graph, attack paths connect an initial state to a target compromise one [6].

The predictive power of attack graphs critically depends on the accuracy of the underlying model of the target system. Noel and Jajodia [7] have shown that incomplete or stale host inventories result in entire attack families to remain invisible to simulation-based defenses because of false negatives. This motivates a continuous, passive inventory building [8], [9].

a) Denial-of-Service Taxonomy: Following the taxonomy of Mirkovic and Reiher [10], we classify DoS/DDoS attacks into three categories. **Volumetric attacks** saturate available bandwidth via high-rate connectionless flows (e.g., UDP floods), exhausting both capacity and router forwarding tables. **Resource exhaustion attacks** target stateful components: the canonical SYN flood [11] exhausts half-open connection tables without completing the three-way handshake. **Application-layer attacks** issue semantically valid but computationally expensive requests; Slowloris [12] holds HTTP sessions open at minimal rate, consuming server connection limits without a detectable bandwidth spike.

B. Flow Passive Monitoring and Deep Packet Inspection

Packet-level traffic analysis at wire speed is computationally prohibitive for high-throughput links and fails to provide a cohesive characterization of communication patterns [13]. The established alternative is flow-based monitoring: a unidirectional flow is formally defined by the five-tuple $f = (\text{src\_ip}, \text{dst\_ip}, \text{src\_port}, \text{dst\_port}, \text{proto})$. By aggregating packets sharing identical attributes, the monitor extracts comprehensive session statistics while reducing memory complexity from $\mathcal{O}(N_p)$ in raw packet capture to $\mathcal{O}(N_f)$, where $N_f \ll N_p$ is the number of active flows.

Bidirectional flows, formed by correlating forward and reverse unidirectional records, expose session-level statistics, including request-response latency and congestion dynamics. Flow exporters implementing IPFIX [14] emit these records to collectors for statistical analysis.

Deep Packet Inspection (DPI) complements flow metadata through content-aware payload analysis, typically applied only to initial connection packets. Even under pervasive TLS, the SNI extension in ClientHello reliably exposes target services [15]. Passive OS fingerprinting analyzes TCP/IP parameter selections (TTL, window size, DHCP option ordering) to identify OS kernels with accuracy exceeding 90% [16].

Passive OS fingerprinting is critical for our VM detection methodology. It analyzes the protocol parameter selection that characterizes each OS kernel: DHCP option ordering, TCP initial window sizes, TLS cipher suite preferences, and IP TTL values [16]. While no single indicator is definitive, their joint analysis via a trained classifier or rule-based system achieves identification accuracies exceeding 90% in controlled settings.

C. Digital and Security Twins: State of the Art

The digital twin concept, proposed by Grieves [17] for product lifecycle management, has been generalized as a continuously synchronized virtual replica of a physical infrastructure [18]. In the security domain, Eckhart and Ekelhart [19] introduced the security-oriented digital twin for ICS environments to test intrusion detection signatures without impacting production systems.

Within twin frameworks, the adoption of machine learning for anomaly detection [11], [20], [21], has achieved high detection rates but at the price of labelled attack data and significant overhead. Our approach operates without labeled samples, relying on statistical characterization of normal behavior—essential for zero-day scenarios. Dietz and Pernul [22] extended digital twin simulation into Security Operations Centers; our work complements this by enriching the twin model through traffic analysis. The challenge of topology obfuscation from virtualization remains underexplored. Pham et al. [23] require active scanning, inapplicable to safety-critical CPS. Our passive SNMP-DPI correlation [24] fills this gap.

III. Statistical Anomaly Classification Model

To enrich the ST with real-time behavioral intelligence, the ST builder maintains a continuous stream of statistical parameters per monitored host. This establishes a mathematical baseline of normal behavior in an initial observation to subsequently detect and classify deviations due to a DoS.

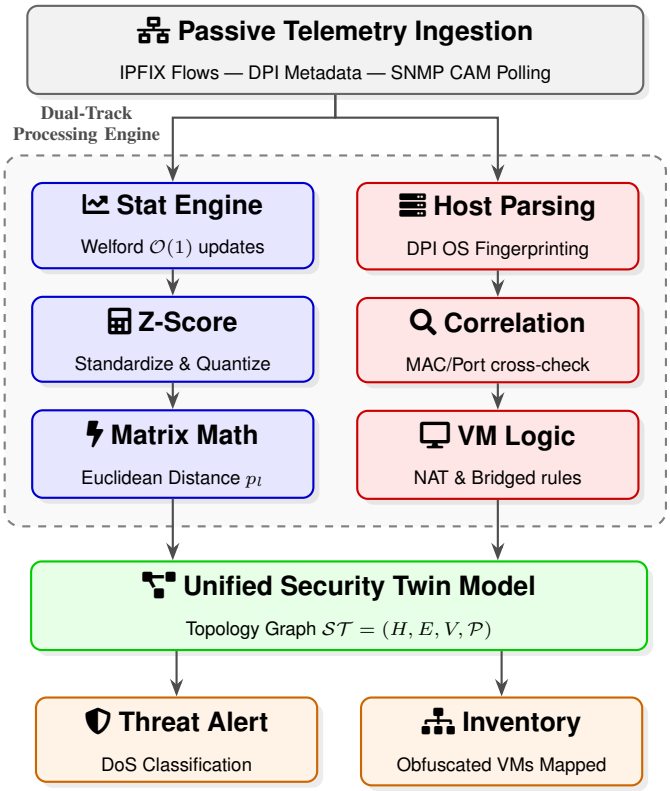

Fig. 1. The extended platform architecture. Raw passive telemetry splits into a dual-track processing pipeline: a constant-time statistical engine for behavior modeling (left) and an anomaly-driven heuristic engine for virtual machine discovery (right). Both tracks continuously enrich the unified Security Twin.

A. Metric Collection and Flow Representation

The ST builder collects network metrics as univariate time series $\{x_i\}_{i=1}^{n}$, where each observation x_i is a scalar measure at discrete time step i. Depending on the metric semantics, observations are either treated directly as gauge measurements (freely fluctuating quantities such as active flow counts) or differentiated as counter increments (monotonically increasing quantities such as byte totals, whose rate of change is the operationally meaningful quantity). This prevents long-term counter-saturation from distorting statistical estimates.

The full set of monitored metrics $\mathbf{m} = (m_1, m_2, \ldots, m_k)^T$ for each host forms a k-dimensional observation vector at each time step, from which the scalar Z-scores are later derived independently per dimension.

B. Numerically Stable Estimation: Welford Algorithm

Naïve two-pass computation of variance is computationally infeasible in a streaming setting. A single-pass accumulation of $\sum x_i^2$ and $\sum x_i$ for subsequent variance computation via the identity $\sigma^2 = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ is numerically unstable for large sample counts due to catastrophic cancellation [25].

We therefore implement the Welford online algorithm [26], which maintains numerical stability through incremental updates without requiring access to previous observations. Let μ_i denote the running mean and $M2_i$ the unnormalized variance accumulator after i observations. The updates at step i are:

$$\delta_i = x_i - \mu_{i-1} \tag{1}$$

$$\mu_i = \mu_{i-1} + \frac{\delta_i}{i} \tag{2}$$

$$\delta_i' = x_i - \mu_i \tag{3}$$

$$M2_i = M2_{i-1} + \delta_i \cdot \delta_i' \tag{4}$$

The sample variance and standard deviation are derived as:

$$\hat{\sigma}_i^2 = \frac{M2_i}{i-1}, \quad \hat{\sigma}_i = \sqrt{\hat{\sigma}_i^2} \tag{5}$$

The decomposition of the update into the two delta terms δ_i and δ_i' is the key to numerical stability: both represent the difference between x_i and a nearby mean estimate and ensure that the magnitudes of the product remain small and well-conditioned regardless of i. The algorithm achieves $\mathcal{O}(1)$ time and $\mathcal{O}(1)$ space per observation, maintaining only the triple $(\mu_i, M2_i, i)$ in memory. Fig. 2 shows the convergence of the running mean and standard deviation estimates during the initial learning of the baseline as a function of observation count. It confirms a rapid convergence within a few hundred samples for the tested metrics.

Welford Algorithm: Running Mean and Std. Dev. Convergence

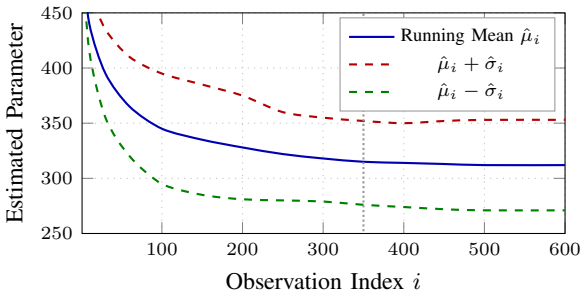

Fig. 2. Convergence of the Welford running mean $\hat{\mu}_i$ and its $\pm\hat{\sigma}_i$ envelope for the *Incoming Flow Count* metric. The baseline stabilizes and is considered reliable for Z-score computation after approximately 350 observations.

C. Z-Score Standardization and Severity Quantization

After convergence, each new observation is standardized using the Z-score transformation:

$$Z_j^{(t)} = \frac{x_j^{(t)} - \hat{\mu}_j}{\hat{\sigma}_j} \tag{6}$$

where $j \in \{1, \ldots, k\}$ indexes the metric dimension and t denotes the current time step. Under the central limit theorem, for a sufficiently large observation count, $Z_j^{(t)}$ follows an approximately standard normal distribution during periods of normal behavior.

We partition the Z-score domain into four discrete severity levels via a threshold function:

$$q_j = \mathcal{Q}(Z_j) = \begin{cases} 0 & \text{if } |Z_j| < 2 \quad \text{(nominal)} \\ 1 & \text{if } 2 \le |Z_j| < 3 \quad \text{(anomalous)} \\ 2 & \text{if } |Z_j| \ge 3 \quad \text{(critical)} \\ -1 & \text{if } Z_j \le -2 \quad \text{(significant decrease)} \end{cases} \tag{7}$$

The quantized vector $\mathbf{q} = (q_1, q_2, \ldots, q_k)^T \in \{-1, 0, 1, 2\}^k$ is a compact, interpretable summary of the current network state that may be compared against theoretical attack profiles.

D. Similarity Matrix and Attack Classification

Let $\mathbf{A} \in \mathbb{R}^{k \times c}$ be the theoretical interval matrix, k the number of monitored metrics and c the one of attack classes plus the baseline (normal) class. Each column $\mathbf{a}_l \in \mathbb{R}^k$ encodes the expected quantized severity signature of attack class l. The classification is based on the vector of Euclidean distances between $\mathbf{q}$ and each column of $\mathbf{A}$:

$$p_l = \|\mathbf{q} - \mathbf{a}_l\|_2 = \sqrt{\sum_{j=1}^{k} (q_j - a_{lj})^2}, \quad l = 1, \ldots, c \tag{8}$$

The predicted attack class is then:

$$\hat{l} = \arg \min_{l \in \{1, \ldots, c\}} p_l \tag{9}$$

A prediction is escalated as an alert only when $p_{\hat{l}} < \theta_{\text{detect}}$ for a sensitivity threshold $\theta_{\text{detect}} > 0$ and simultaneously $p_{\text{baseline}} > \theta_{\text{normal}}$, ensuring that the engine does not misclassify legitimate traffic bursts as attacks.

Fig. 3 shows the evolution of the Z-scores for three representative metrics across the experimental timeline. It includes the baseline phase and the three injected attack scenarios, illustrating the clear separability of the attack signatures.

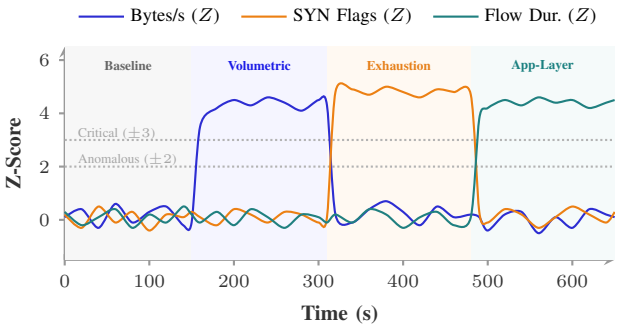

Fig. 3. Z-score time series of leading indicators across experimental phases. The engine dynamically evaluates independent metrics against severity thresholds ($|Z| = 2$ anomalous, $|Z| = 3$ critical) to expose highly specific threat signatures.

IV. Passive Virtual Machine Identification

An accurate ST faithfully describes virtualized environments. Identifying VMs exclusively through passive network monitoring is inherently challenging because a VM's protocol stack is functionally indistinguishable that of a physical host at the flow level. Techniques that rely solely on OS fingerprinting are prone to systematic errors in cloud and hypervisor environments where hardware signatures are homogenized across virtual instances. Our methodology addresses the two most prevalent VM networking configurations: network address translation and bridged adapters.

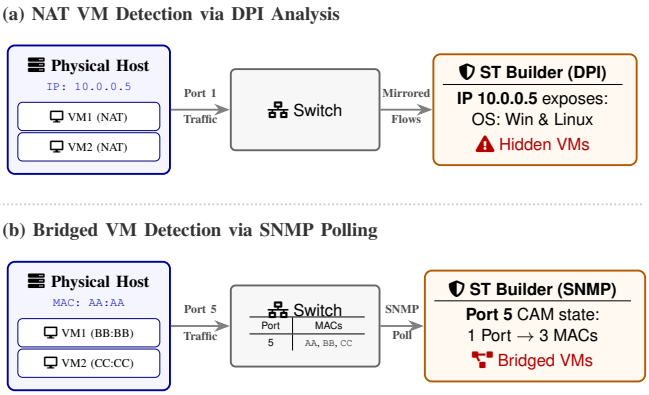

Fig. 4. Passive identification of virtualized environments. (a) Detecting NAT VMs by capturing deep packet inspection (DPI) OS fingerprinting contradictions from a single origin IP. (b) Mapping Bridged VMs by polling switch CAM tables via SNMP to expose multi-MAC port violations.

A. Identifying NAT-Configured Virtual Machines

When a VM operates behind a hypervisor NAT, it shares the public IP address of the physical host and relies on a virtual router to translate internal addresses. From the perspective of a passive observer on the main network segment, all outbound traffic appears to originate from a single endpoint.

a) Detection Principle: The ST builder exploits the observable inconsistencies in OS fingerprinting to infer the presence of hidden VMs. Advanced DPI libraries extract OS fingerprints by analyzing a combination of signals from the initial packets of connection-oriented sessions and DHCP lease requests. Each OS kernel exhibits a characteristic set of TCP/IP parameter selections (TTL, initial window size, window scaling options, MSS, DHCP option ordering) that constitute its passive fingerprint.

If a single IP address α exhibits simultaneously two or more distinct OS fingerprints $\mathcal{F} = \{f_1, f_2, \ldots, f_r\}$ with $r \ge 2$ over a time window Δt, the ST builder infers the existence of at least $r - 1$ hidden VMs behind α. The physical host is identified as the node whose fingerprint corresponds to the longest continuous uptime estimate, derivable from TCP timestamp clock skew analysis.

b) Limitations: Method fails when all VMs run identical OS with same kernel configuration. As mitigation, the ST builder additionally tracks TCP timestamp drift the rates and TLS session identifier cycling, which differ measurably between native and guest instances. False positives from OS reinstallation are suppressed by requiring the fingerprint change to coincide with a DHCP lease renewal on an unchanged MAC address in the CAM table.

B. Identifying Bridged Virtual Machines

In a bridged adapter configuration, the hypervisor creates a virtual NIC for the VM that receives its own distinct IP address via DHCP and a unique MAC address, causing standard flow monitors to perceive it as an independent physical device.

a) Detection Principle: The ST builder exploits the structural property that a bridged VM appears as an independent host that shares the same physical switch port as its hypervisor. This property is observable through the switch's MAC address forwarding table (CAM table), which maps MAC addresses to physical ports.

Under normal operating conditions, a single host-facing port p in the CAM table is associated with exactly one MAC address. After excluding cascade uplink ports and wireless access point ports, the presence of $r \geq 2$ MAC addresses on port p, as identified through OUI prefix analysis and neighbor discovery protocols, is a reliable indicator of $r - 1$ bridged VMs attached to the physical host at that port.

The ST builder retrieves CAM table entries via SNMP polling of the MIB-II `dot1dTpFdbTable` object. The polling interval τ_{snmp} has to be shorter than the CAM table ageing timeout of the switch (typically 300 s) to ensure no transient MAC entry is missed.

b) Formal VM Topology Mapping: Let $\mathcal{M}(p)$ denote the set of MAC addresses observed on port p across the SNMP polling window. The bridged VM identification procedure is:

1) For each port p: if $|\mathcal{M}(p)| \geq 2$ and $p \notin \mathcal{P}_{\text{uplink}} \cup \mathcal{P}_{\text{wap}}$, flag p as a VM host port.
2) Let $\text{MAC}_{\text{host}}(p)$ be the MAC address in $\mathcal{M}(p)$ with the earliest DHCP lease timestamp observed by the flow monitor.
3) All the other $\text{MAC}_i \in \mathcal{M}(p) \setminus \{\text{MAC}_{\text{host}}(p)\}$ are virtual nodes with parent host $\text{MAC}_{\text{host}}(p)$.
4) Update the edges of ST to reflect the inclusion relationship.

V. Experimental Evaluation

Our experiments are structured into two complementary components. The first evaluates the statistical DoS classification engine using a controlled simulation environment. The second assesses the passive VM discovery methodology using a physical testbed. Together, these experiments validate the principal claims of this paper: zero misclassification across all three attack categories, and an accurate reconstruction of the inventory for both NAT and bridged VM configurations [27].

A. Simulation Environment and Experimental Setup

The DoS classification evaluation required an environment capable of generating precise, massively scalable, and fully reproducible traffic floods while preserving a realistic background traffic baseline. We selected the ns-3 discrete-event network simulator (v3.40), which provides an accurate representation of the full network stack and produces PCAP traces that are structurally identical to those from physical NICs.

Our simulated system is centred on a primary edge router that acts as the passive observation point, connecting a target application server to a wide-area network populated by 500 legitimate client nodes. Background traffic was generated using an ON/OFF statistical model with Pareto-distributed on/off intervals, accurately replicating the heavy-tailed burstiness characteristic of modern web protocols (HTTP/1.1 and HTTP/2 session patterns).

a) Baseline Learning Phase: The detection engine was initialized by processing 600 seconds of benign background traffic to allow the Welford algorithm to reach statistical convergence. Table I shows the extracted baseline parameters.

TABLE I
Baseline Statistical Parameters During Normal Operation

Monitored Metric	Mean ($\hat{\mu}$)	Std. Dev. ($\hat{\sigma}$)
Incoming Flow count	312	41
Bytes received per second	330	64
Packets received per second	52	13
UDP flow count	175	35
Amplitude class variation	54	10
TCP SYN flag count	149	8
Out-of-order / Retransmissions	14	2
Average flow duration (s)	5.0	1.5
Target throughput (Kbps)	280	55

B. Attack Injection and Z-Score Analysis

Following baseline convergence, three distinct attack vectors were injected sequentially by compromised nodes within the simulated WAN: (i) a UDP flood representing a volumetric attack; (ii) a SYN flood representing resource exhaustion; and (iii) a Slowloris attack representing an application-layer exploit. Table II shows the Z-scores at the peak of each attack.

TABLE II
Z-Scores at Peak Attack Intensity for Each Experimental Scenario

Monitored Metric	Volumetric	Exhaustion	Application
Incoming Flows	$+2.8$	$+4.6$	$+2.4$
Bytes per second	$+4.5$	$+0.8$	$+0.2$
Packets per second	$+4.2$	$+2.9$	$+0.4$
UDP Flows	$+4.8$	-0.2	-0.1
Amplitude Variation	$+3.9$	$+4.1$	$+0.5$
TCP SYN Flags	-0.5	$+4.9$	$+2.1$
Out-of-Order Packets	$+2.8$	$+3.2$	$+2.2$
Average Flow Duration	-1.2	-2.5	$+4.5$
Target Throughput	-0.5	-0.7	-4.8

C. Distance Vector Analysis and Classification Results

To compare the Z-score vectors against the theoretical similarity matrix y, we computed the Euclidean distance. Let p_B, p_V, p_R, and p_A denote the distance from, respectively, the baseline (normal), volumetric, resource-exhaustion, and application-layer profiles. The distance vectors are:

$$\mathbf{p}^{(\text{Vol})} = [4.24,\ 1.73,\ 3.87,\ 4.90] \tag{10}$$

$$\mathbf{p}^{(\text{Res})} = [4.12,\ 3.46,\ 0.00,\ 4.36] \tag{11}$$

$$\mathbf{p}^{(\text{App})} = [3.32,\ 4.47,\ 4.00,\ 1.00] \tag{12}$$

In all three scenarios, the minimum distance index correctly identifies the active attack type. Crucially, the baseline column distance p_B remains significantly elevated across all attack scenarios (ranging from 3.32 to 4.24), confirming that the engine reliably avoids false positives due to legal traffic bursts.

Fig. 5 shows the distance vectors, making the classification margin unambiguous.

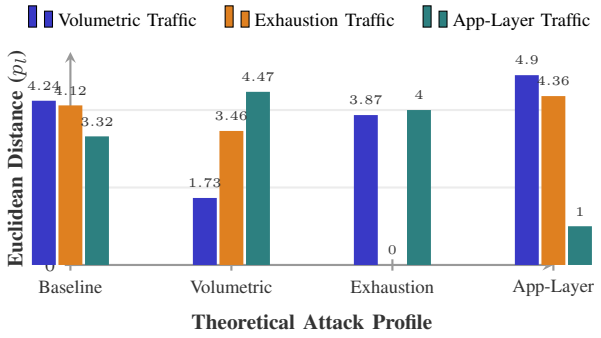

Fig. 5. Euclidean distances p_l between observed traffic streams and theoretical profiles. The minimum distance correctly isolates the attack type, while robust margins from the baseline profile securely prevent false positives.

D. Sensitivity Analysis: Threshold Variation

To characterize the classification robustness with respect to the severity quantization thresholds, we conducted a parametric sensitivity analysis by varying the critical Z-score threshold θ_{crit} from 2.0 to 4.0 in increments of 0.25. For each threshold, we computed the number of correctly classified time windows across the three attack scenarios. Figure 6 reports the classification accuracy as a function of θ_{crit}.

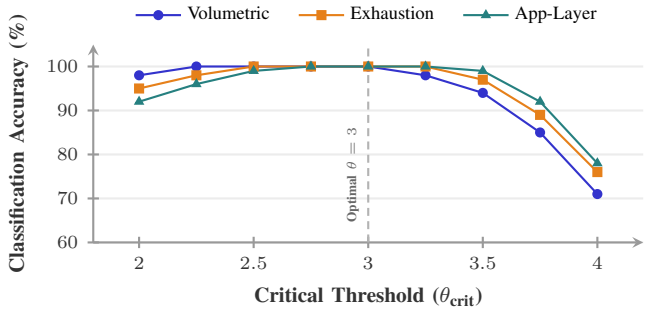

Fig. 6. Classification accuracy across the three attack profiles as a function of the severity quantization threshold. Performance degrades gracefully beyond the optimal $\theta_{\text{crit}} = 3.0$.

The results confirm that $\theta_{\text{crit}} = 3.0$ represents the optimal operating point, yielding 100% accuracy across all three attack categories. We attribute the degradation for $\theta_{\text{crit}} > 3.5$ to metrics whose Z-scores fall below the elevated threshold during low-intensity attack ramp-up phases. This can be mitigated by lowering the detection threshold at the cost of a marginal increase in false positive rate during heavy legitimate congestion periods.

E. Physical Testbed: Virtual Machine Discovery

The VM discovery evaluation required a physical testbed to faithfully replicate the hardware-level interactions of commercial hypervisors and managed network switches. The laboratory environment consisted of: a physical target host running a Linux kernel with a Type-2 hypervisor; a managed Layer-2 switch with SNMP v2c capabilities; a local network router; and a dedicated observation node executing the extended NotLine ST builder.

a) NAT VM Detection: The target physical host instantiated a Windows VM behind hypervisor NAT. The passive DPI engine captured an initial DHCP request fingerprinted as Linux (matching the physical host), followed by an outbound TLS session fingerprinted as Windows (originating from the VM). The contradiction in OS fingerprints tied to a single IP address triggered the VM detection heuristic. By correlating TCP timestamp clock skew values across the two fingerprinting events, the ST builder correctly attributed the longer uptime to the physical host, thereby registering one hidden virtual node. No manual intervention was required.

b) Bridged VM Detection: Table III report the switch CAM table state before and after bridged VM initialization.

TABLE III
SWITCH CAM TABLE BEFORE AND AFTER BRIDGED VM INITIALIZATION

Before		After	
Port	**MACs**	**Port**	**MACs**
1	MAC_{host}	1	MAC_{host}, $\text{MAC}_{\text{virtual}}$
7	$\text{MAC}_{\text{monitor}}$	7	$\text{MAC}_{\text{monitor}}$
8	$\text{MAC}_{\text{router}}$, MAC_{ext}	8	$\text{MAC}_{\text{router}}$, MAC_{ext}

Port 8 was excluded from VM detection by the uplink exclusion filter (two MAC addresses co-resident on a port connected to the network router are consistent with a downstream switch, not a VM host). Port 1 exhibited a second MAC address upon VM activation: the platform correctly identified MAC_{host} as the physical entity (earliest DHCP timestamp) and $\text{MAC}_{\text{virtual}}$ as its bridged virtual child. The complete topology was reconstructed in a single SNMP polling cycle following VM activation.

c) Computational Overhead: Table IV reports the measured per-packet and per-flow processing latencies for each NotLine module on the observation node (Intel Core i5-12400, 16 GB RAM), confirming that the extended platform introduces negligible overhead compatible with real-time operation.

TABLE IV
PER-OBSERVATION PROCESSING LATENCY FOR EACH NOTLINE MODULE

Module	**Mean Latency**	**Max Latency**
Flow aggregation	1.2 μs	3.8 μs
Welford update (per metric)	0.08 μs	0.21 μs
Z-score computation	0.11 μs	0.28 μs
Similarity matrix product	3.4 μs	7.2 μs
DPI OS fingerprinting	18.7 μs	42.1 μs
SNMP CAM poll (per switch)	12.3 ms	28.9 ms

The Welford update and Z-score computation together require under 200 ns per metric per observation, confirming the $\mathcal{O}(1)$ complexity claim in practice. The SNMP polling latency, executed asynchronously at configurable intervals, has no impact on the per-packet processing pipeline.

VI. Conclusion

The effectiveness of a ST is bounded by the accuracy and structural completeness of its underlying inventory model. This paper has shown that continuous passive flow analysis enriches an ST along three orthogonal dimensions without requiring active probing or labeled attack data.

The constant-time statistical engine, grounded in the numerically stable Welford algorithm [26], achieves zero misclassification across all three canonical DoS categories—volumetric, resource-exhaustion, and application-layer—via a formal similarity matrix framework. The dual-mode passive VM identification methodology, correlating DPI OS fingerprinting inconsistencies for NAT environments with SNMP CAM table analysis for bridged configurations, resolves hardware topology obfuscation with complete accuracy in a single polling cycle. Together, these mechanisms produce a continuously updated, high-fidelity ST that unifies behavioral anomaly detection and structural topology reconstruction within a single non-intrusive pipeline. Measured processing latencies confirm real-time compatibility on commodity hardware.

The unified framework produces a high-fidelity, continuously updated ST that generalizes traditional anomaly detection by incorporating structural, behavioral, and performance dimensions within a single passive observation pipeline. Measured processing latencies confirm that the extended NotLine platform meets the requirements for real-time deployment on commodity hardware.

Future work will: (i) dynamically adjust the SNMP polling interval to observed CAM table change rates, minimizing overhead while preserving reconstruction accuracy; (ii) extend the statistical engine to online multivariate covariance estimation, enabling detection of coordinated multi-vector attacks whose individual metric signatures remain sub-threshold.

References

[1] F. Baiardi, V. Sammartino, and S. Ruggieri, "Notline: A non-intrusive automated platform to build a digital twin," in *2025 29th IEEE DS-RT*, 2025, pp. 1–8.

[2] F. Baiardi, V. Sammartino *et al.*, "From digital twins to ai agents: A synthetic data paradigm for next-generation cybersecurity," *Artificial Intelligence in Cybersecurity: Unlocking the Power of Large Language Models*, 2026.

[3] F. Baiardi and V. Sammartino, "Quantifying resilience of cyber-physical systems to zero-day threats: A digital twin-based what-if analysis." ESREL 2026, 2026.

[4] F. Baiardi, V. Sammartino, and S. Ruggieri, "Evaluating adversary strategies through a security twin," in *Proc.s of the IEEE PerCom*, 2026.

[5] F. Baiardi, S. Ruggieri, and V. Sammartino, "Ai-enabled cybersecurity using synthetic data," in *2025 IEEE PerCom*, 2025, pp. 140–145.

[6] F. Baiardi and V. Sammartino, "Quantifying the impact of cvss score ordering on attack paths." EAI GOODTECHS 2026, 2026.

[7] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation," in *Proceedings of the ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*. ACM, 2004, pp. 109–118.

[8] V. Sammartino, "A framework for proactive cyber-resilience: Non-intrusive modeling for autonomous defense," in *2025 29th IEEE DS-RT*. IEEE, 2025, pp. 1–4.

[9] V. Sammartino, F. Baiardi, and S. Ruggieri, "A Security Twin to Defeat Intrusions in Cyber Physical Systems," in *ESREL SRA-E 2025*, 2025.

[10] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[11] Y. Mirsky *et al.*, "Kitsune: An ensemble of autoencoders for online network intrusion detection," *arXiv*, 2018.

[12] R. Hansen, "Slowloris HTTP DoS," http://ha.ckers.org/slowloris/, 2009, accessed: 2024-10-01.

[13] T. Zseby, M. Molina, N. Duffield, K. Claffy, and F. Raspall, "Sampling and filtering techniques for IP packet selection," *IETF RFC 5475*, 2009, available at https://datatracker.ietf.org/doc/rfc5475/.

[14] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information," Internet Engineering Task Force, RFC 7011, 2013.

[15] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proc. of the ACM Workshop on Artificial Intelligence and Security (AISec)*. ACM, 2019, pp. 35–46.

[16] S. Bano, P. Richter, M. Javed, S. Sundaresan, Z. Durumeric, S. J. Murdoch, R. Mortier, and V. Paxson, "Scanning the internet for liveness," in *Proc. of the ACM SIGCOMM Conference*. ACM, 2018, pp. 586–599.

[17] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," *White Paper*, 2014, florida Institute of Technology.

[18] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167 653–167 671, 2019.

[19] M. Eckhart and A. Ekelhart, "Towards security-aware virtual environments for digital twins," in *Proc. of the ACM Workshop on Cyber-Physical Systems Security and Privacy*. ACM, 2018, pp. 61–72.

[20] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proc. of the ACM Workshop on Cyber-Physical Systems Security and Privacy (CPS-SPC)*. ACM, 2018, pp. 72–83.

[21] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symposium on Security and Privacy (S&P)*, 2010, pp. 305–316.

[22] M. Dietz and G. Pernul, "Integrating digital twin security simulations in the security operations center," in *Proc. ACM/SIGAPP Symposium on Applied Computing (SAC)*, 2022, pp. 1632–1641.

[23] V.-T. Pham, M.-T. Le, and N. K. Tran, "Passive virtual machine fingerprinting and identification in data center networks," in *Proc. of the IEEE International Conference on Advanced Information Networking and Applications (AINA)*. Springer, 2020, pp. 487–498.

[24] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Proc. Network and Distributed System Security Symposium (NDSS)*, 2003.

[25] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Algorithms for computing the sample variance: Analysis and recommendations," *The American Statistician*, vol. 37, no. 3, pp. 242–247, 1983.

[26] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[27] F. Baiardi and V. Sammartino, "A quantitative framework for the validation of twin-based cyber defense," in *I3M*, 2025.