Towards High-Performance Intrusion Detection with Robustness Guarantees on Programmable Switches at ISP Scale

Han Zhang[1], Xuefeng Liu[1], Linqiang Qian[1], Guyue Liu[3], Tianyu Zhang [1], Kaiyang Zhao[2], Yantu Tong[1], Zeji Xiao[1], Dongbiao He[5], Yahui Li[1], Ke Ruan[4], Jilong Wang[1], Yongqing Zhu[4], Xia Yin[2]
[1]Institute for Network Sciences and Cyberspace, Tsinghua University
[2]Department of Computer Science and Technology, Tsinghua University
[3]School of Computer Science, Peking University
[4]China telecom
[5] School of Computer Science and Engineering, Southeast University

ABSTRACT

In order to provide security connections to the enterprise campus sites, internet service providers are offering comprehensive intrusion detection services at the network layer. However, existing network intrusion detection systems (NIDS) are either ineffective or inefficient for high-speed network protection, especially for encrypted traffic analysis. In this paper, we design and implement SiteGuard, an inline network intrusion detection system with programmable switches specifically developed to protect enterprise campus sites connecting to ISP. SiteGuard proposes a dual-plane feature extraction model to extract extensive traffic features at near line-speed. SiteGuard also proposes a lightweight one-class classification model that trains the best parameters exclusively on benign traffic to identify malicious traffic. In addition, SiteGuard introduces an online update mechanism that aims to dynamically adjust the detection model in response to environmental changes. SiteGuard has been in production for more than three years. Our production and testbed evaluations demonstrate SiteGuard can detect malicious traffic with approximately 90% accuracy in minutes.

CCS CONCEPTS

• **Networks** → **Network monitoring**; **Network measurement**; **Network security**;

KEYWORDS

Security; Network Intrusion Detection; Programmable Switches

ACM Reference Format:
Han Zhang[1], Xuefeng Liu[1], Linqiang Qian[1], Guyue Liu[3], Tianyu Zhang [1], Kaiyang Zhao[2], Yantu Tong[1], Zeji Xiao[1], Dongbiao He[5], Yahui Li[1], Ke Ruan[4], Jilong Wang[1], Yongqing Zhu[4], Xia Yin[2]. 2026. Towards High-Performance Intrusion Detection with Robustness Guarantees on Programmable Switches at ISP Scale. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3789240.3829107

1 INTRODUCTION

The rapid development of big data and AI [25] has significantly increased the demand for campus sites while also driving a steady rise in economic losses from network attacks [12, 13, 16, 58]. The connection of these sites to ISP networks has further amplified the scale and frequency of attacks (Figure 1). Our analysis, based on a survey of 132 enterprise campus sites from two ISPs (§2), identifies three key challenges complicating defense: (i) a tenfold surge in campus site ingress bandwidth over the past decade (Figure 2); (ii) the prevalence of encrypted traffic (now >90%) (Figure 3), which obscures threat detection; and (iii) rapid change, with >20% year-over-year shifts in traffic patterns (Figure 4). Collectively, these trends present profound challenges for intrusion detection solutions at the ISP scale.

Network Intrusion Detection Systems (NIDS) are the frontline defense for monitoring traffic and identifying anomalies, but current frameworks fall short in addressing the unique demands of enterprise campus protection at the ISP scale. Existing NIDS are broadly categorized into out-of-band and inline deployments, each with critical **limitations**:

Out-of-band NIDS, which passively monitor traffic via mirrored data streams, face a fundamental tradeoff between speed and capability. Rule-based variants [8, 29, 33] excel at high-speed processing but fail to handle encrypted traffic, while machine learning (ML) [17, 18, 39, 62] and deep learning (DL)-based approaches [53, 60, 67] show promise for encrypted traffic analysis but require duplicating massive volumes of high-speed traffic to offloaded servers—creating prohibitive latency and resource overhead for ISP networks.

Inline NIDS, which operate directly on the critical traffic path for real-time detection, suffer from cost or accuracy constraints. FPGA-based implementations [49, 55] deliver high performance but are expensive, inflexible, and difficult to upgrade. Programmable switch solutions [3, 14, 37, 40, 61, 65] offer cost-effectiveness and line-speed processing but rely on simplistic detection rules that cannot handle complex or encrypted attacks and lack adaptability to network changes.

SiteGuard Framework. To address these gaps, we present SiteGuard, an inline NIDS specifically designed for enterprise campus site protection, leveraging the capabilities of programmable switches (§3). SiteGuard achieves both *efficient* and *robust* malicious traffic detection through the following key designs:

Dual-plane feature extraction model (§3.1). To extract key network traffic features, the state-of-the-art programmable switch-based NIDS frameworks [7, 34, 37, 61, 65] primarily rely on threshold-driven detection in the control plane or offload simple machine learning algorithms to programmable data planes for efficient anomaly detection. While these approaches enable line-speed traffic processing, their accuracy remains low due to the limited feature diversity and the simplicity of the detection methods.

Instead, we propose a new dual-plane detection model: basic operations (e.g., packet counting, lookups) are offloaded to programmable data plane engines for high performance, while more

This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGCOMM '26, Denver, CO, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2467-1/26/08
https://doi.org/10.1145/3789240.3829107

complex computations (e.g., division, exponential operations) are delegated to CPU engines to fully utilize their general computing capabilities.

This dual-plane feature extraction model enables SiteGuard to successfully balance the high speed and accuracy of traffic profiling. SiteGuard extracts 80 statistical features, 8 TLS semantic features, and 5 sequence features to provide a comprehensive traffic description, where the statistical features are capable of identifying plaintext attacks (e.g., DDoS), the TLS semantic features are capable of identifying malicious client fingerprints (e.g., unauthorized access), and the sequence features are capable of identifying encrypted malicious traffic (e.g., malicious programs).

Lightweight one-class classification model (§3.2). The advanced NIDS [53, 60, 67] commonly employ deep learning-based methodologies for traffic analysis to guarantee the accuracy of detection. Despite high accuracy, they necessitate plenty of labeled traffic and a large amount of computational resources. Inspired by the fact that machine learning algorithms incorporating dedicated feature extractions can achieve a high degree of accuracy [17, 39, 62], SiteGuard advocates performing lightweight anomaly detection.

The extraction of valuable information concealed within encrypted traffic in a resource-constrained environment is challenging. Therefore, we propose a 2D Discrete Fourier Transform (2DDFT) to transform the traffic representation to model the spatiotemporal relationship within and between the feature sequences. In practice, system developers also face the challenge of obtaining the security event traffic of the production network for privacy reasons. SiteGuard introduces a one-class classification model that trains the best parameters exclusively on benign traffic, which enables it to detect network anomalies in a lightweight manner with high performance.

An online update mechanism (§3.3). Traditionally, machine learning algorithms would retrain with all the traces to update the parameters, which would result in significant overhead as well as performance fluctuations. To determine optimal parameters, SiteGuard employs an online update mechanism for parameter updates. When a high false alarm rate is detected, SiteGuard enters the update phase, during which the detection model is refined using samples. Our results demonstrate that SiteGuard is capable of adjusting the parameters to environmental changes. We also prove that as the feedback number approaches infinity, the performance disparity between the optimal static strategy and our online update mechanism converges to zero.

Contributions. SiteGuard has been in production for three years, handling thousands of TB of traffic daily. In summary, our main contributions include:

- We undertake an in-depth study of the evolving requirements for campus site protection and present a new perspective on network intrusion detection at ISPs (§2).
- We design SiteGuard, an inline intrusion detection system leveraging programmable switches to provide effective and precise malicious traffic detection service for the campus sites (§3).
- We perform comprehensive evaluations both in production (§4) and real-world traffic experiments (§5). Our production results indicate that SiteGuard performs about 30% better than the commercial NIDS. Our testbed evaluations show that SiteGuard has stable performance, and it can achieve a precision improvement of up to 25% compared to state-of-the-art programmable switch-based frameworks (Pegasus [63], Netbeacon [65], Jaqen [37]).

2 BACKGROUND AND MOTIVATION

We first introduce the enterprise campus site security challenges that bring three new requirements (§2.1). Then we present the intrusion detection service in production at the ISP scale (§2.2). Finally, we illustrate the limitations of existing work, which provides the motivation for SiteGuard (§2.3).

2.1 Enterprise Campus Site Security Problems

The enterprise campus sites are encountering escalating security concerns due to their storage and processing of extensive quantities of sensitive information, such as personal data, company secrets, and critical financial records. The number of network attacks faced by sites has increased rapidly in recent years. As illustrated in Figure 1, our survey conducted across 132 enterprise campus sites from two Internet Service Providers (ISPs) indicates that the number of reported attack events per site is rising by about 10% per year.

Attack Analysis. We further conducted a more thorough examination of the survey's attack events, and Figure 1 illustrates that the following four attack types account for over 90% of network attacks: (1) Network probing (62%). The attackers always use some scan tools (e.g., nmap [43]) to find the weaknesses of the device systems; therefore, they can exploit the vulnerabilities to compromise the system; (2) Denial of Service attack (20%). Attackers can send packets to targeted devices to make the resource exhausted (e.g., SYN flooding). Denial of service (DoS) attacks have become a major threat to current computer networks [9]; (3) Malicious Program Abuse (10.1%). External intruders can exploit the malware (e.g., Parite [4], Zeus [59]) to launch attacks from remote locations through the internet. The Malicious Program Abuse (MPA) attacks have already adopted the encrypted protocols to secure the communications [4]; (4) Unauthorized Access to Users (5%). The Unauthorized Access to Users (U2U) attacks attempt to abuse vulnerabilities in the system in order to gain superuser privileges or perform unauthorized access.

The exponential growth in data generation, combined with rapidly evolving internet connectivity, has created new imperatives for campus network protection.

Requirement 1 - Low-delay processing for high-speed traffic. In recent years, campus networks [19, 31] have undergone significant upgrades to accommodate soaring bandwidth demands. Figure 2 demonstrates that the campus sites in our study have experienced a significant growth of over ten times in the percentage of high-speed Internet Service Providers (ISPs) access (such as speeds of 10 Gbps or greater) from 2017 to 2025. With the increasing speed of internet connections, it is crucial for ISPs to make sure that their security systems can effectively analyze the high-speed network traffic without causing too many delays.

Requirement 2 - Encrypted traffic analysis. A growing number of internet sites are adopting traffic encryption to secure their traffic. Figure 3 shows approximately 92% of ISP to the campus site traffic was encrypted in 2024, which is 3× larger than in 2014 in our survey. However, traffic encryption is a double-edged sword, which complicates the discernment of malicious behaviors. The

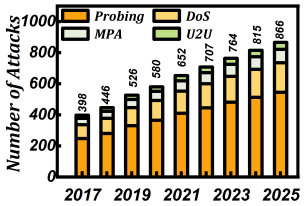


Figure 1: Attack events.

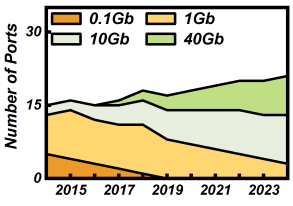


Figure 2: Ingress capacity.

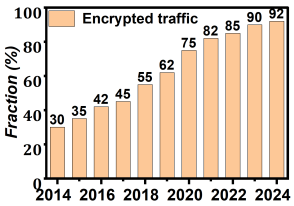


Figure 3: Traffic fraction.

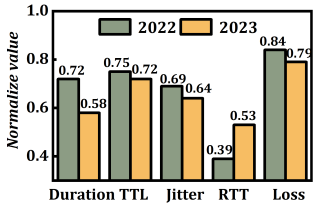


Figure 4: Network feature.

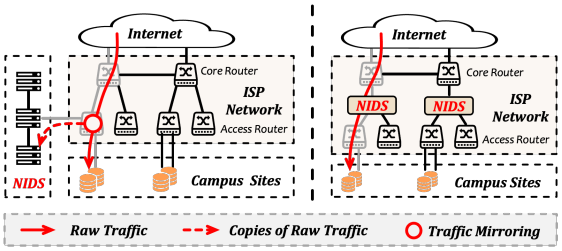


Figure 5: Out-of-band (left) and inline (right) mode.

	Existing frameworks	R1	R2	R3
Out-of-band	**Rule**(e.g., Chimera [8], Kargus [29], vNIDS [33])	●	○	○
	Machine learning(e.g.,Kitsune [39], Whisper [17])	◐	◐	◐
	Deep learning (e.g.,Tiki-taka [60], BurNet [53])	○	◐	○
Inline	**FPGA** (e.g., NetFPGA [55], BIDS [49])	◐	○	○
	Programmable switch(e.g., Jaqen [37],Pegasus [63], Netbeacon [65])	●	◐	○
	SiteGuard	●	●	●

Table 1: Existing frameworks are incapable of meeting the new demands, where ○, ◐, and ● correspond to unsupported, partially supported, and fully supported.

encryption diminishes the plaintext data essential for identifying malicious traffic. Traditional traffic analysis methods (e.g., deep packet inspection [1]) are invalid under encryption (e.g., SSL/TLS). Due to the rising prevalence of encrypted internet applications, ISPs are facing greater difficulties in detecting attacks.

Requirement 3 - Concept drift adaption. In the data mining area, the concept drift is a change in the relationship between the input data and the target model. It reflects the evolution of the underlying problem statement or process over time. For the malicious traffic detection, concept drift frequently occurs due to the ongoing evolution of traffic attributes, such as varying protocol percentages and connection duration [21, 22]. This drift is primarily attributable to the inherently dynamic characteristics of network environments. These characteristics encompass changes in user behavior, the deployment of novel applications, and the emergence of increasingly sophisticated cyberattacks. Figure 4 presents a comparative analysis of the average flow length, average packet interval, and average packet size across three campus sites in 2022 and 2023, as derived from our survey. These feature drifts can potentially undermine the effectiveness of protection methods over time, rendering models designed for analyzing such data rapidly obsolete.

2.2 Intrusion Detection Service at ISP

ISP security services [5, 26, 45] denote a suite of managed cybersecurity protections delivered directly at the network layer by an ISP as a value-added layer atop core connectivity offerings. Up until now, many ISPs (e.g., AT&T [5], China Telecom [11]) have already provided such services.

Compared with traditional campus-centric solutions, security models at the ISP present several distinct **advantages**: First, *proactive defense at the network edge*. By deploying detection and mitigation capabilities within their backbone or peering points, ISPs can identify and filter threats (e.g., DDoS attacks, malware) before they reach the customer's network perimeter. This upstream, network-wide visibility enables earlier response than campus-centric NIDS alone. Second, *effortless deployment and simplified management*. Security measures are implemented and maintained at the ISP level, eliminating the need for customers to select, deploy, and operate third-party security appliances. This turnkey approach reduces operational complexity and accelerates time-to-protection. Third, *scalability and cost-effectiveness*. In a campus-centric model, each organization must purchase, deploy, and maintain its own security infrastructure, which is expensive. By centralizing security resources, ISPs achieve economies of scale, delivering protection as a subscription service that lowers both the financial outlay and the technical barrier for end users.

2.3 Limitations of Existing Work

Network intrusions often manifest as anomalous traffic patterns. As shown in Figure 5, ISPs typically deploy Network Intrusion Detection Systems (NIDS) at the access layer of campus networks to deliver security services to their customers. As demonstrated in Table 1, the existing NIDS can be categorized into two primary types based on their deployment strategies: *out-of-band* and *inline*.

Out-of-band NIDS. As illustrated in Figure 5, the out-of-band NIDS operates in a passive manner, monitoring network traffic by passively sniffing the internal interface of the switch in a read-only mode. It subsequently transmits alerts to an analysis server via a separate network interface that supports both read and write operations. The analysis server is responsible for executing traffic detection algorithms and generating alarms when malicious activities are detected. Based on the detection methodologies, the out-of-band NIDS can be further classified into three categories: rule-based methods, machine learning-based methods, and deep learning-based methods.

The rule-based methods (e.g., Chimera [8], Kargus [29], vNIDS [33]) detect the malicious behaviors with the predefined rules. In contrast, the machine learning-based methods (e.g., HyperVision [18], Kitsune [39], Whisper [17]) can effectively identify the zero-day malicious traffic, but they always achieve low detection accuracy for the encrypted traffic and are unable to adapt to environment change. The deep learning methods (e.g., Vuldeepecker [67], Tiki-taka [60], BurNet [53]) are capable of achieving high detection accuracy. Nevertheless, these methods necessitate a significant amount of computational resources for operation and behave in a black box fashion with an interpretive problem.

Inline NIDS. As shown in Figure 5, the inline NIDS operates on the critical path and directly inspects raw network traffic. This approach

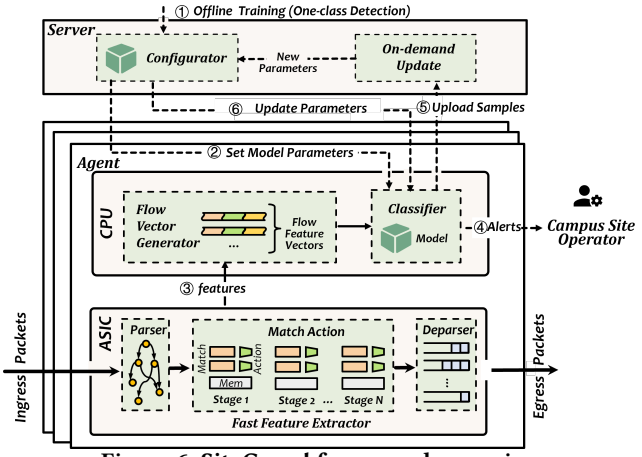


Figure 6: SiteGuard framework overview.

ensures that potential threats are identified and addressed in real-time, enhancing the overall efficiency and effectiveness of the intrusion detection system. By analyzing the traffic as it flows through the network, the inline mode can provide immediate responses to detect anomalies, thereby maintaining network security without significant latency. As shown in Table 1, according to hardware types, existing inline frameworks can be categorized into FPGA-based methods (e.g., NetFPGA [55]) and programmable switch-based methods (e.g., HorusEye [14], Jaqen [37], Peregrine [3], iGuard [40], In-Forest [34], Netbeacon [65], FlowLens [7],Pegasus [63], Leo [28]). The FPGA foundational methods exhibit high performance; however, these frameworks are expensive, possess limited functionalities, and are challenging to upgrade for evolving threats.

Existing programmable switch-based approaches can be further divided into two categories: control plane detection methods (e.g., Jaqen [37], Poseidon [61]) and intelligent data plane methods (e.g., Netbeacon [65], In-Forest [34], Pegasus [63], Leo [28]). The control plane detection methods focus on extracting useful flow information from the data plane and performing anomaly detection on the control plane. Due to the frequency interaction latency between the control plane and the data plane, most existing approaches [37, 61] adopt simple detection rules to guarantee the processing speed. Further, their malicious detection algorithms are too simple and incapable of conducting encrypted traffic analysis. The intelligent data plane methods recommend all operations be offloaded to data plane in order to facilitate line-speed detection. Despite these methods being capable of detecting malicious traffic at line speed, data plane is incapable of performing complex computations (e.g., exponentiation, division). Consequently, their precision might be low.

3 SITEGUARD DESIGN

This part presents the design and implementation of SiteGuard , an inline intrusion detection system designed with programmable switches to provide precise and effective malicious traffic detection.

As shown in Figure 6, SiteGuard consists of two high-level components: **Agent** and **Server**. The SiteGuard agent works at the programmable switch. It contains three parts, i.e., Fast Traffic Processor (FTP), Feature Vector Generator (FVG), and Classifier. The FTP component extracts flow statistics features (e.g., average flow length) and packet sequence features (e.g., packet size sequence) at

Type	Encrypted	Robust	Session-aware	Coordination
control plane detection	●	○	○	●
intelligent data plane	●	○	○	●
SiteGuard	●	●	●	●

Table 2: Comparisons between SiteGuard and state-of-the-art programmable switch inline NIDS solutions, where ○, ◐, and ● correspond to unsupported, partially supported, and fully supported.

the data plane. The FVG component extracts the semantic features (e.g., TLS extension patterns) and forms the feature vector at the control plane. The Classifier component runs the malicious traffic detection algorithm. The SiteGuard server works at the ISP network operator center (NOC), and it is responsible for configuring and updating the model. SiteGuard involves the operational phase and the update phase.

Operational phase: Since it is hard for the system developers to attain the production-labeled attack events,SiteGuard performs offline training of the one-class malicious traffic detection model just by the benign traffic patterns (①,§3.2). Then SiteGuard server configures the agent (②). By generating flow vectors through an adjustable detection window ϵ (e.g., two to three minutes), the FVG component can effectively balance detection latency with cost efficiency (③, §3.1). Upon identifying malicious behavior, the classifier promptly reports alarms to the campus site operators (④).

Update phase: In practice, the fundamental characteristics of the network, such as bandwidth, packet interval, and connection duration, can exhibit significant variability. This variability may cause the training environment to diverge from the conditions observed in the real deployment environment. When the false positive rate of traffic analysis exceeds a predefined threshold after deployment, it indicates that the current parameters are no longer adequately aligned with the network conditions, prompting the system to enter the update phase. During this phase, the classifier component of SiteGuard agent reports the extracted flow features and the corresponding detection results to SiteGuard server (⑤). SiteGuard also provides an interface to execute a model update algorithm designed to adapt to these environmental changes (§ 3.3). The updated parameters are subsequently configured to the corresponding agents. Once the observed false positive rate falls below the threshold, SiteGuard transitions back to the operational phase (⑥).

Novelty. As shown in Table 2, compared with existing programmable switch-based methods, SiteGuard differs in **Four** key aspects: First, *mixed traffic analysis.* While prior work focuses on volumetric attacks (e.g., DoS) and is ineffective against encrypted threats, SiteGuard extracts statistical, semantic, and sequential features to detect malware hidden in encrypted flows. *Second, robust detection.* Unlike static models in current frameworks, SiteGuard employs an online parameter-update scheme based on operator feedback, ensuring continuous adaptation and long-term robustness. *Third, session-aware data plane.* Unlike prior switch-based telemetry frameworks (e.g., *Flow [54], Sonata [20]) that adopt a *session-agnostic per-flow cache* exporting raw packet tuples for software-side aggregation, SiteGuard maintains per-session state and extracts semantic, sequential, and statistical features in one pass, detecting malware hidden in encrypted flows rather than merely volumetric attacks (e.g., DoS). *Fourth, coordinated resource utilization.* Whereas existing approaches isolate the control and data planes, SiteGuard synergizes both: high-frequency features are extracted in the data plane

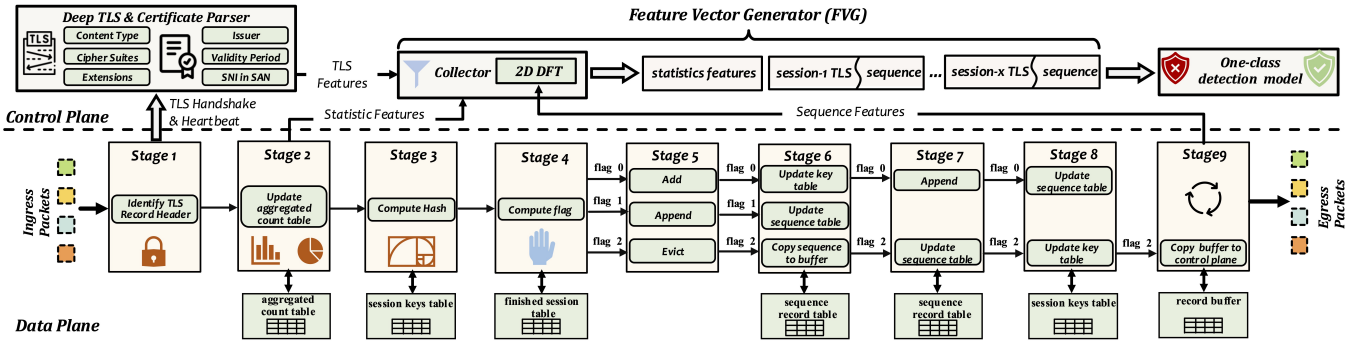


Figure 7: The dual-plane feature extraction model is implemented on the programmable switch.

while a lightweight 2D-DFT one-class classifier runs on the control plane, performing on-switch inference—exploiting both hardware resources for efficient and accurate detection.

3.1 Fast Feature Extraction

Accurately modeling high-speed, encrypted network traffic is challenging. SiteGuard overcomes this by extracting statistical, semantic, and sequential features. Statistical features (e.g., IPs per time window) identify pattern-based attacks like port scans and DoS, regardless of encryption. Semantic features (e.g., TLS extensions, certificate anomalies) detect malware and unauthorized access. Sequential features (e.g., packet size sequences) capture temporal patterns, revealing complex threats hidden in encrypted flows.

★ **Design 1: A dual-plane feature extraction model for fast and comprehensive traffic description.** Programmable switches feature both general-purpose CPU engines and specialized programmable data plane engines, each with distinct limitations. CPU engines support complex logic like semantic analysis but cannot sustain rapid packet processing. Data plane engines deliver line-rate performance for basic operations but have limited memory for active flows and lack support for advanced arithmetic (e.g., division, exponentiation). To bridge these asymmetries, SiteGuard adopts a cooperative model, where the data plane handles high-speed preprocessing and the CPU executes complex logic.

As shown in Figure 7, the dual-plane feature extraction model tracks session-level states in the data plane using a session key table, an aggregated count table, a sequence record table, a finished session table, and a record buffer. Due to the limited processing capacity of programmable data planes for complex semantic analysis, TLS handshake and heartbeat packets are mirrored to the control plane for deep parsing. Sequence features for each session are stored in the sequence record table. For every session, SiteGuard extracts the size, arrival time, jitter, protocol type, and payload size of the first K packets in the data transmission phase to form its sequence features. If the session contains fewer than K packets, zeros are padded for the missing entries. Once the first K packet's sequence features are extracted, the session hash is added to the finished session table and excluded from further processing within the current time window. The aggregated count table updates statistics across five granularities: destination IP, destination port, source–destination IP pair, source socket, and destination socket. The metrics tracked include packet count, total packet size, unique port count, and unique source IP count. During processing, SiteGuard queries this table for session-specific statistics. In total, 20 aggregated count features and 5 sequence features are extracted per session in the data plane (detailed in Appendix B). The record buffer, structured as a ring buffer, temporarily stores extracted features and is transferred to the switch CPU's main memory via DMA once it is full or at the end of the adjustable detection window ϵ. This avoids frequent high-overhead data copies to data plane memory.

On ingress, the parser detects TLS headers and mirrors handshake and heartbeat packets to the control plane for semantic analysis. Support for other protocols, such as QUIC, is discussed in Appendix A. In later stages, SiteGuard aggregates packet statistics into bidirectional sessions based on 5-tuples, with termination determined by TCP flags or timeouts. A session hash is then computed to query the state tables. For new ($flag = 0$) or active ($flag = 1$) sessions, sequential features are extracted and stored in the sequence record table. Once the first K packets are captured or the session terminates ($flag = 2$), its record is moved to the buffer and marked as finished to prevent further processing, maintaining line-rate efficiency and avoiding redundant computation.

Since the data plane cannot perform complex computations, the control plane handles the remaining feature extraction tasks. First, the control plane analyzes the mirrored packets to derive 8 TLS semantic features, including protocol behavior, client fingerprinting, and certificate identity (Appendix B), which constitute TLS feature vector v_{tls}. Second, SiteGuard computes the statistics features using the aggregated count features. Let $g(m)$ denote the m-th aggregated count feature, which is obtained by aggregating the statistical features of individual flows from the data plane for the current time window. For the m-th aggregated count feature, SiteGuard first derives the average rate as $l(m) = g(m)/\epsilon$. Let $l'(m)$ denote the average rate for the m-th count feature over previous time windows. Let $d(m)$ denote the gap between $l(m)$ and $l'(m)$: $d(m) = |l(m) - l'(m)|/l'(m)$. By concatenating $g(m)$, $l(m)$, $l'(m)$ and $d(m)$, we will derive statistics feature vector v_s, whose size is 80 (20 base features × 4 dimensions) at each detection window end.

3.2 Lightweight One-class Classification

At the Classifier component, SiteGuard performs anomaly detection to achieve robust, high-performance malicious traffic detection. Prior studies have demonstrated that machine learning models with dedicated feature extraction can attain high detection accuracy. Therefore, SiteGuard proposes implementing detection on the programmable control plane rather than offloading it to external hardware servers.

To realize this, SiteGuard must address two key **challenges**: First, the detection model must be lightweight yet accurate, as programmable switches have limited general-purpose CPU resources. This requires efficient processing of session vectors within tight resource constraints to ensure detection speed. Second, the model should achieve high accuracy without relying on expert-labeled datasets. While network anomaly detection typically demands large volumes of labeled attack data, manual traffic labeling is resource-intensive and expertise-dependent. Moreover, attack data is often sensitive and inaccessible to system developers.

★ **Design 2: A lightweight one-class classification model for identifying malicious traffic.** Recent studies demonstrate that encryption implementations leave identifiable footprints in encrypted data, enabling effective traffic analysis. Deep learning algorithms [35, 51, 64] excel at extracting temporal patterns from sequence features, yet their complexity limits deployment in high-speed scenarios such as programmable switches. To address this, SiteGuard introduces a 2D Discrete Fourier Transform (2DDFT) method, converting sequence features into compact statistical representations suitable for real-time analysis.

For each time window ϵ, SiteGuard randomly selects F flows to perform the 2DDFT. Let $h(x, y)$ denote the sequence feature matrix extracted at the data plane for each flow f. The element at row x ($1 \le x \le M$) and column y ($1 \le y \le N$) is the y-th value of the x-th sequence feature. The corresponding frequency domain feature can be derived as $\sum_{x=1}^{M}\sum_{y=1}^{N} h(x,y) \cdot e^{-j2\pi\left(\frac{p*(x-1)}{M}+\frac{q*(y-1)}{N}\right)}$. Let $r(p,q)$ and $s(p,q)$ represent the real and imaginary parts after transformation, where $1 \le p \le M$, $1 \le q \le N$. By applying the Euler's formula to change $r(p,q)$ and $s(p,q)$ to trigonometric function form, we can compute the amplitude of each frequency domain component: $z(p,q) = \sqrt{r^2(p,q) + s^2(p,q)}$. To make the frequency domain features stable, we perform a transformation on $z(p,q)$ and use the integer constant C_{min} and C_{max} to adjust the range of the frequency domain features. Let Z^p_{min} and Z^p_{max} denote the maximal and minimal value of the p-th sequence feature, we have $z'(p,q) = \lfloor \frac{C_{max}-C_{min}}{Z^p_{max}-Z^p_{min}} \times (z(p,q) - Z^p_{min}) \rfloor + C_{min}$.

SiteGuard differs from prior frequency-based methods [17, 50] in two main ways. First, rather than applying Fourier transforms directly to payload bits—a computationally costly approach that fails against encrypted traffic—SiteGuard uses packet metadata (e.g., size, arrival time) to maintain both efficiency and privacy. Second, unlike techniques such as Whisper [17], which flatten multi-dimensional features into a 1D sequence and lose feature independence, SiteGuard preserves structural relationships through a 2D feature matrix. This design captures the macroscopic spatiotemporal distribution, going beyond mere temporal periodicity.

Finally, each flow's sequence features will end up with a new vector, whose length is $L = M * N$. Let $\overrightarrow{R}_f$ denote the sequence vector after 2DDFT for flow $f \in F$, then SiteGuard proposes an auto-encoder compression model to compress the sequence vector to avoid sparsity. The 2DDFT sequence features will be passed to an auto-encoder whose encoder and decoder part are constructed by several neuron convolution layers. We also add skip connections to the original structure to accelerate the training speed, whose connecting strategy follows the pre-activation version of the deep residual network [24]. The decoder net outputs a vector $\overrightarrow{R}'_f$ and SiteGuard aims to minimize the mean square error, i.e., $\min \sum_{i=1}^{L}(\overrightarrow{R}_{fi} - \overrightarrow{R}'_{fi})^2$. The output of the encoder part v_f of the flow f whose size is $|v_f|$. Together with the statistics flow vector v_s, SiteGuard will construct the *anomaly detection vector*, whose size is $|v_f| * |F| + |v_s|$ Let $\overrightarrow{w}$ denote the weight vector and ρ denote the threshold, with the detection vector v_t for the t-th time window ϵ, SiteGuard detects anomalies: $g_t = sign(\overrightarrow{w} \cdot \overrightarrow{v}_t - \rho)$, where $sign(.) \in \{-1, 1\}$ is the sign function, i.e., $g_t = 1$ means the t-th time window contains malicious behaviors and $g_t = -1$ means it is normal. Let T denote the total training anomaly detection vector set with benign traffic, whose size is $|T|$, SiteGuard trains $\overrightarrow{w}$ as:

$$\min_{\overrightarrow{w},\xi,\rho} \quad \frac{1}{2}\|\overrightarrow{w}\|^2 + \frac{1}{\nu|T|}\sum_{t\in T}\overrightarrow{v}_t - \rho \tag{1}$$

$$\text{subject to} \quad (\overrightarrow{w} \cdot \phi(\overrightarrow{v}_t)) \ge \rho - \xi_t, \quad \xi_t \ge 0, \quad \forall t \in T \tag{2}$$

In reality, attacks are themselves increasingly *distributed*: botnet-driven reflection DDoS, multi-source brute-forcing, and coordinated scanning campaigns deliberately spread their traffic across many sources and paths so that each fragment stays individually inconspicuous. This distribution is further amplified at the network level by ECMP load balancing, multi-homed peering, and anycast routing, so a single coordinated attack against one victim enters through *multiple* ingress switches at once—and no individual agent ever observes the complete attack. To solve this, SiteGuard adopts a *two-tier* architecture in which local agents cooperate through the centralized NOC server to reassemble and detect such distributed, multi-ingress attacks.

At the end of each detection window ϵ, every agent uploads to the NOC server: (i) its 80-dimensional statistics feature vector, which is already computed by the local pipeline (20 aggregated count features × 4 dimensions), incurring no additional extraction overhead; and (ii) a victim-keyed digest, which summarizes per-destination intensity and is derived directly from the data-plane count table. The server aggregates these inputs across all agents by applying SUM over count-based dimensions (e.g., total packets and bytes) and MAX over rate and deviation dimensions to preserve the worst-case signal. The merged statistics and digest are then concatenated into a global view vector. The server subsequently runs a same second-tier one-class detector but with its own parameters, trained on historical benign global-view vectors. When the detector triggers, the server alerts all relevant agents and campus operators; confirmed global events are fed back through the online update rule, adapting both the global detector and, where attributions are feasible, the corresponding local agents.

This design enables detection of coordinated multi-ingress attacks that no single agent can perceive, while preserving each agent's autonomy and line-rate operation: the local detection pipeline remains unchanged, and the server merely fuses summaries that are byproducts of existing computation.

3.3 Online Update Scheme

As network conditions evolve, reliance on static model parameters can lead to a rise in false alarm rates. In practice, campus site clients may report security events to their operators, who can verify the actual occurrence of attacks. SiteGuard provides interfaces to support

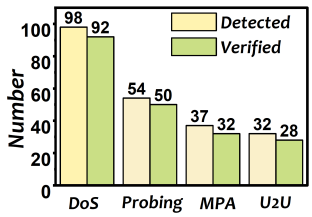


Figure 8: Total events.

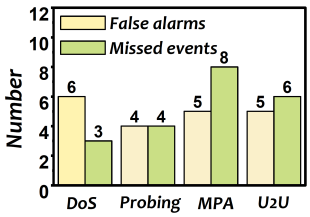


Figure 9: Failure detection.

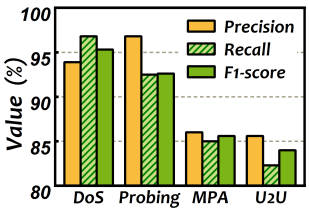


Figure 10: Metrics.

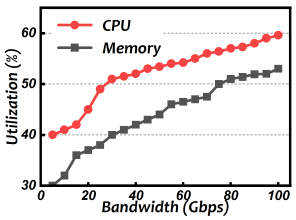


Figure 11: Resources.

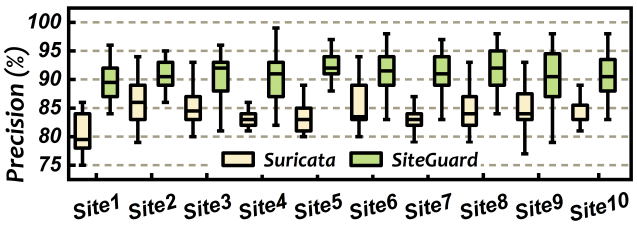


Figure 12: Precision comparison across campus sites.

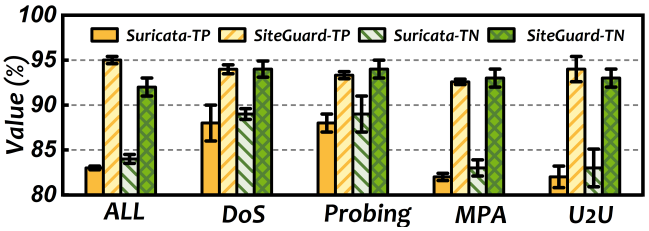


Figure 13: Precision comparison for different attacks.

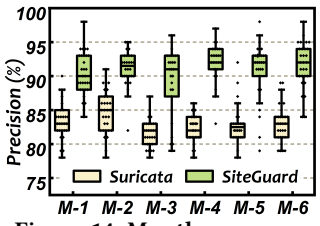


Figure 14: Month accuracy.

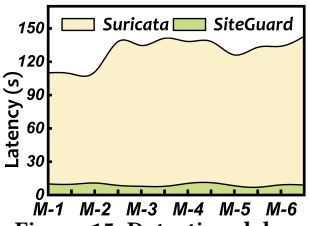


Figure 15: Detection delay.

model updates based on this feedback. As illustrated in Figure 6, when the deviation between the empirically observed false alarm rate (α) and the predefined threshold (β) exceeds a critical value (δ), SiteGuard enters an update phase. During this phase, campus operators can submit verified results to the SiteGuard server, which then applies an online parameter adjustment method to adapt the model to environmental changes. Once the difference between α and β falls below δ, the SiteGuard agent notifies the server and resumes normal operation.

★ Design 3: An online update scheme to make the parameters adapt to the environment changes. After entering the update phase, SiteGuard server begins to update the model parameters with the corrected detection results from SiteGuard agent. In reality, the ISP NOC can also help to verify the corrected anomaly detection vectors to help improve the precision. Let g_i present the corrected detection result (-1 is benign and 1 is malicious) and $\overrightarrow{v_i}$ is the anomaly detection vector of the i-th time window. We now define the following loss function for the i-th time window: $\theta(g_i \neq sign(\overrightarrow{w} \cdot \overrightarrow{v}_i - \rho))$, where $\theta(.)$ is the indicator function that takes the value 1 if the parameter is satisfied and 0 otherwise. However, it is hard to use in practice since it is 0/1 loss and needs to access the whole data set. We will replace it with the function: $L(\overrightarrow{w}, i) = \max\{0, 1 - g_i \times (\overrightarrow{w} \cdot \overrightarrow{v}_i - \rho)\}$. Let F denote the anomaly detection vectors with corrected detection results. we can derive the best parameter $\overrightarrow{w}$:

$$\overrightarrow{w} = \arg\min_{\overrightarrow{c}} \{\sum_{i \in F} L(\overrightarrow{c}, i) + R(\overrightarrow{c})\} \tag{3}$$

$R(.)$ is a regularized function (i.e., the regularizer) to stabilize the prediction. From (3), we can derive the best parameters. However, it is often time-consuming to solve, which hurts the spirits of online prediction. To derive the best parameter when receiving the i-th time window feedback, we aim to use any vector which has a minimal loss on all past rounds, i.e.,

$$\overrightarrow{w}_{i+1} = \arg\min_{\overrightarrow{c}} \{\sum_{j=1} L(\overrightarrow{c}, j) + R(\overrightarrow{c})\} \tag{4}$$

In this case, we adopt the online mirror descent algorithm [38] and change (3) to its gradient function : $L(\overrightarrow{c}, i) \simeq \frac{\partial L(\overrightarrow{c}, i)}{\partial \overrightarrow{c}} * \overrightarrow{c}$. We use L2 Regularization [38] and set $R(\overrightarrow{c})$ as $R(\overrightarrow{c}) = \frac{1}{2\eta_i}(\overrightarrow{c} - \overrightarrow{w}_0)^2$, where η_i is the learning rate at the i-th iteration.$\overrightarrow{w}_0$ is the initial weight vector before entering the update phase. After the transformation, we can get the optimal by solving (4). According to Karush–Kuhn–Tucker condition, we take the derivative of (4) equals to 0. The equation is easy to be solved and we can get the optimal update value of vector $\overrightarrow{w}$ as $\overrightarrow{w}_{i+1} = \overrightarrow{w}_i + \eta_i \times g_i \times \overrightarrow{v_i}$.

Let $\boldsymbol{w}^*$ denote the best static decision parameters. The regret metric [23] can measure the difference between the online algorithm and the best parameters. A practical online algorithm should have a sub-linear regret upper bound. This implies that on average the algorithm is performing as the best static strategy we can have in hindsight,. i.e.,

$$Regret_F = \max_{\overrightarrow{w^*}} \left\{ \sum_{i \in F} L(\overrightarrow{w}, i) - \sum_{i \in F} L(\overrightarrow{w^*}, i) \right\} = o(F) \tag{5}$$

Lemma 3.1. *Let G be the maximal value of the vector $\overrightarrow{w}$ whose length is H, Let $\eta_i = G/\sqrt{\eta_i}$, the regret metric is smaller than $2GH^2\sqrt{|F|}$.*

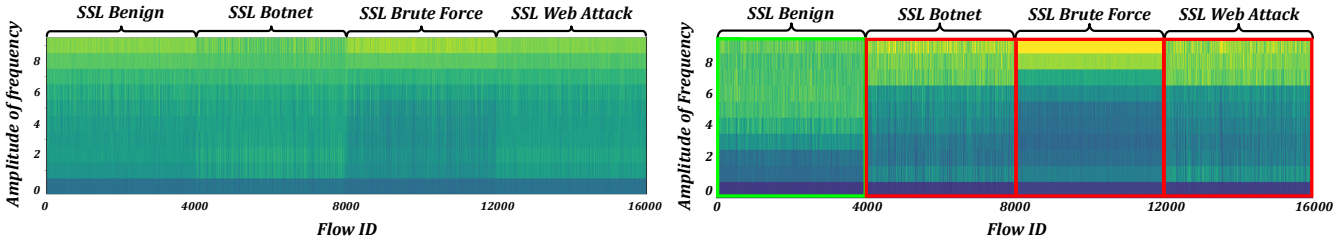
We show the proof in appendix C. This implies that the gap between the best offline static strategy and the model goes to zero as the feedback number goes to infinity.

4 SITEGUARD IN PRODUCTION

SiteGuard has been in production for three years, handling thousands of TB traffic daily. Over the past three years, it has identified over 246 major attack events.

4.1 Overall performance

In this section, we will report the overall performance of SiteGuardacross the two ISPs. In our production results, campus site operators served as the ground-truth verifiers. When malicious



(a) 1D-DFT (Whisper): Indistinguishable spectral patterns.

(b) 2D-DFT (SiteGuard): Distinct spectral patterns.

Figure 16: Performance comparison with 1:1 mixed benign traffic alongside mixed malicious traffic scenarios.

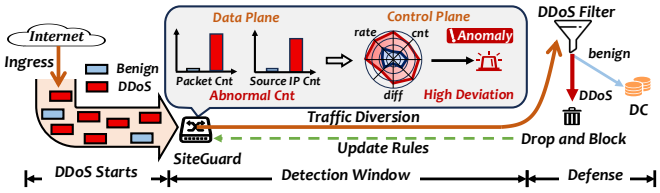


Figure 17: Rapid DDoS Defense.

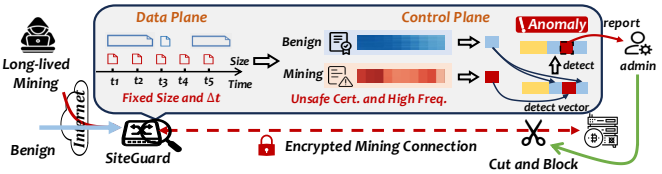


Figure 18: Crypto Mining Detection.

events were identified, SiteGuard notified the site operators, who then validated the detections and relayed the results to the ISP operators. Site operators also reported security incidents disclosed by their customers to the ISP operators, thereby incorporating threats that SiteGuard had not initially detected. Figure 8 summarizes the overall detection results, showing that more than 90% of all detected events were correct. As illustrated in Figure 9, nearly all DoS and port-scan attacks were detected by SiteGuard, while fewer than 14% of MPA and U2U events were missed. Figure 10 indicates that the overall accuracy of SiteGuard exceeds 90% across all attack types and performance metrics, demonstrating stable performance in production. Furthermore, Figure 11 shows that the control-plane resource utilization of SiteGuard remains below 60% even in 100Gbps.

4.2 A/B test

In comparison, we also deployed the commercial Suricata-based out-of-band rule based NIDS before 10 campus sites to analyze the traffic. The product manual indicates that the Suricata-based NIDS is implemented utilizing 20 general-purpose servers proficient in rapidly sampling and detecting malicious traffic according to pre-established rules. The A/B test lasted 6 months ($M1$-$M6$).

Figure 12 demonstrates that the overall precision accuracy of SiteGuard is over 90% over the 10 campus sites. SiteGuard has stable performance, and it is up to 30% better than the Suricata-based NIDS. The performance of SiteGuard is superior to Suricata-based NIDS due to its ability to manage all traffic and adapt to changes in the environment, whereas the Suricata-based NIDS are limited to managing partial traffic with strict rules. Figure 13 illustrates that SiteGuard exhibits steady detection accuracy across all attack types, since it can effectively capture diverse features to describe both the plain-text traffic and the encrypted traffic. In comparison, the Suricata-based NIDS has relatively high performance for the volumetric attacks (e.g., DDoS and port scan) since it lacks efficient rules for the encrypted traffic. Figure 14 shows the detection accuracy comparison of the 6 months, we can see SiteGuard performs stable, and it is about 20% better than the Suricata-based NIDS. Note that SiteGuard is an inline NIDS, when compared with the out-of-band NIDS, it has much lower detection latency since it does not need to copy the traffic to extra servers for analysis. Figure 15 shows that the detection delay of SiteGuard is about 10× smaller than the Suricata-based NIDS over the 6 months.

4.3 2DDFT in production

To explain the high detection accuracy observed in production, particularly against obfuscated threats, we visualized the traffic representations. Benign ISP traffic was mixed with malicious traffic at a 1:1 ratio. For each flow, the first 100 packets ($N = 100$) were used, with $C_{min} = 0$ and $C_{max} = 10$. Figure 16 presents the results, comparing four traffic scenarios in a single view: benign traffic and mixed botnet, brute-force, and web-attack traffic. Darker colors indicate a higher packet density at a given frequency.

As shown in Figure 16(a), the 1DDFT method (e.g., Whisper [17]) performs poorly in this mixed environment. Its projection onto a single dimension statistically smooths out localized attack features—such as correlated changes in packet size and timing—making malicious traffic less distinguishable. In contrast, SiteGuard's 2DDFT approach (Figure 16(b)) successfully reveals distinct attack patterns. By preserving the matrix structure, anomalous couplings between packet size and timing are directly mapped to identifiable high-frequency components, enabling clear identification of malicious traffic. This distinct feature separation is the fundamental reason SiteGuard maintains robust performance in complex production.

4.4 Representative Real-World Examples

We further validate the performance of SiteGuard in production through two representative deployment cases.

Rapid DDoS Defense. SiteGuard mitigated a high-volume DDoS attack on a 40 Gbps link (Figure 17). The attack exhibited a sudden surge in packet rate and unique source IPs. SiteGuard detected this anomaly through line-speed statistical aggregation in the data plane and real-time deviation analysis from historical baselines in the control plane. Upon confirmation, it immediately redirected traffic to a DDoS filter and deployed blocking rules. This integrated mechanism responds approximately 15× faster than commercial

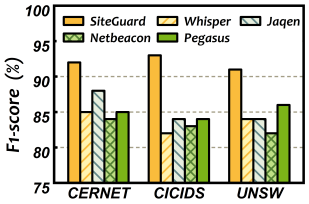


Figure 19: Different traces.

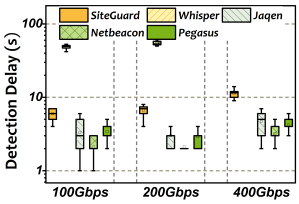


Figure 20: Delay vs. load.

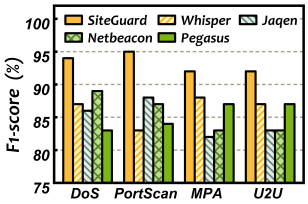


Figure 21: Different attacks.

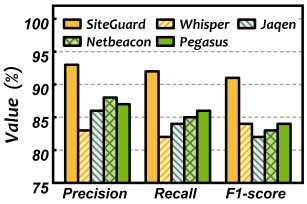


Figure 22: Metrics.

Suricata-based NIDS, blocking attack traffic within minutes to minimize service disruption and economic loss.

Crypto Mining Detection. SiteGuard successfully detected a TLS-encrypted cryptojacking attack [48] during deployment. Figure 18 shows that the attack was characterized by a long-lived connection to a remote mining pool with remarkably regular packet intervals and fixed payload sizes. This regularity translated into a concentrated frequency-domain signature, starkly contrasting the diffuse spectrum of normal traffic. SiteGuard fused this signal with semantic TLS fingerprints associated with risk to confirm the anomaly. Following attribution to the specific malicious flow, the system generated an alert and enforced a blocking rule in real time to neutralize the threat.

5 TESTBED EVALUATION

In this section, we evaluate the performance of SiteGuard by replaying real-world network traffic traces. Our main results are as follows:

- Compared to existing methods, SiteGuard demonstrates superior performance across datasets and achieves high detection accuracy for all attack types (§5.2).
- There is no noticeable processing latency change for the SiteGuard enabled programmable switches (§5.3). SiteGuard successfully adapts to environmental shifts, whereas baseline methods struggle to adjust and suffer from significant performance degradation (§5.4).

5.1 Setup

SiteGuard agent is implemented over the 3.2 Tbps Tofino switch [6]. In each experiment, 10 servers are connected to the switch with 40 Gbps to replay the traffic. We enabled the Intel DPDK [15] library on each server to achieve high-performance traffic replay. We used 9 servers to replay benign traffic and the other server to replay abnormal traffic in our evaluations. Furthermore, we used an additional server with a 400 Gbps capacity that is connected to the switch as the target and launched an additional server to execute the model update and configuration. Every five seconds (ϵ), SiteGuard agent would report the detection results. The ground truth of each flow is regarded as the feedback of the user. The default predefined false alarm rate threshold (e.g., β) is 10% and the default update trigger threshold δ is 5% in our evaluations. We choose the first 5 packets (K=5) to extract the sequence features and then choose 100 flows ($|F|$=100) to form the detection vector at each detection window. By default, each experiment is conducted 10 times, and we report the average value as the result.

Baselines. We compared SiteGuard with the following three type of methods: (1) *Out-of-band NIDS.* We employed Whisper [17] as the representative out-of-band NIDS framework. Since the out-of-band NIDS requires traffic to be copied to the server for offline analysis, we launched a new server to run the method; (2) *Control plane detection inline NIDS.* Jaqen [37] is a switch-native approach for volumetric DDoS defense with programmable switch. The source code of Jaqen [37] is not publicly available, and we independently implemented it on the programmable switch using the same parameters outlined in [37]; (3) *Intelligent data plane inline NIDS.* NetBeacon [65] and Pegasus [63] consider both packet level and flow level features for their detection. We implemented NetBeacon and Pegasus based on their open source code.

Traffic trace and ground truth. We evaluated the performance with the following three real world traffic traces: (1) *CERNET dataset.* The China Education and Research Network (CERNET) is the largest national education and research computer network in China. The dataset comprises approximately 54GB of data, containing around 400 million flow records with raw packet information collected between December 1 and December 2, 2020, with a sampling ratio of 1:1. The trace has already been categorized as either normal or as specific attack types by experts. (2) *CICIDS 2017 Dataset* [52]. The CICIDS 2017 dataset was developed by the University of New Brunswick and is generated within a simulated real-world network environment. This dataset encompasses a variety of cyberattacks, including DoS, Brute Force, Infiltration, Heartbleed, DDoS, Botnet, and Web-based attacks; All the traffic have already labeled benign and the anomaly type. (3)*UNSW-NB15 2015 Dataset* [41]. The UNSW-NB15 2015 dataset was developed by the University of New South Wales and contains over two million labeled records. Each record is characterized by 49 features, which are represented as nominal, integer, or floating-point values. The dataset encompasses seven distinct types of anomalies: Analysis, Backdoor, DoS, Exploits, Fuzzers, etc.

5.2 End-to-end Detection

F1-score vs. traces. We begin by showing the performance comparison under different traffic traces. Figure 19 shows the F1-score comparison between SiteGuard and the other frameworks. We can see that SiteGuard has consistently high detection accuracy under all datasets, and it is able to perform up to 10% better than the other frameworks. The reason for this is that SiteGuard adopts various heterogeneous features for security incident detection. In contrast, while certain frameworks may extract a greater number of features (e.g., Pegasus [63]), they only consider sequence features or statistics features, resulting in a relatively lower level of accuracy.

Latency vs. load. SiteGuard designs the dual-plane feature extraction model for fast and comprehensive traffic description (§3.1). Figure 20 shows that the detection delay of all the frameworks. We can see that the detection delay of SiteGuard is 10 × shorter than

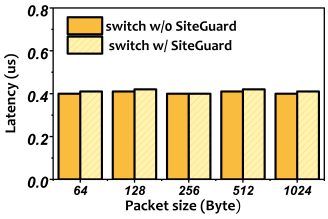


Figure 23: Processing.

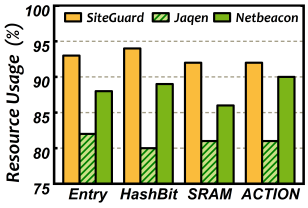


Figure 24: Resource usage.

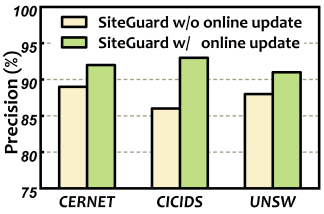


Figure 25: Benefits.

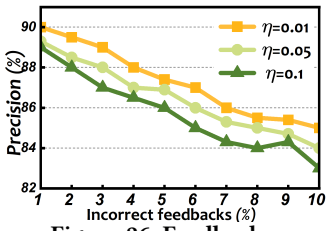


Figure 26: Feedbacks.

that of the out-of-band methods, but it is approximately 10% longer than that of the programmable switch-based frameworks due to the fact that it processes complex computations. In practice these additional delays are insignificant and acceptable to users.

F1-score of different attacks. As introduced in section 3.1, through extracting the statistics features and sequence features, SiteGuard is able to detect the plain-text anomaly and encrypted anomaly . Figure 21 shows that SiteGuard maintains high detection accuracy for all attack events. In comparison, the programmable switch based methods (e.g., Jaqen [37]) have high performance for the volumetric attacks. Whisper considers the sequence features, therefore, it has higher accuracy for the encrypted attacks (e.g., MPA).

Different metrics. The precision metric is the number of true positive (TP) items divided by the total number of elements labeled as the positives. The recall metric is the value of a class is the number of true positive items divided by the sum of true positives and false negatives. F1-score combines the precision and recall. Figure 22 shows that SiteGuard has the steady performance under all the metrics. For a more granular evaluation, including more performance details and an ablation study validating the efficacy of our 2DDFT design, please refer to Appendix D.

5.3 Overhead Comparison

Process latency. One advantage of using hardware switches is that the processing latency is extremely small for legitimate traffic. To confirm this, we evaluated the processing latency of SiteGuard. As depicted in Figure 23, there is no noticeable processing latency change for the SiteGuard enabled programmable switch.

Hardware resource usage. The programmable switches allow developers to define their own packet formats and program the packet actions by a series of match action tables, which are mapped into different stages in a sequential order, along with dedicated resources (e.g., match entries, hash bits, SRAMs, and action slots). Figure 24 shows the resource usage comparison of different programmable switch based inline NIDS. We can see that compared with other schemes, SiteGuard is able to fully utilize the resource when compared with other frameworks.

5.4 Online Update Scheme Evaluation

Benefits of online update. With the online update scheme, SiteGuard is able to adjust its parameters according to network changes. Figure 25 shows that SiteGuard with online update scheme performs about 8% better than SiteGuard without online updates, which proves its benefits.

React to incorrect feedbacks. In practice, the false feedback from data center users or operators could hurt the detection accuracy. We injected a small fraction of false feedback. Figure 26 illustrates that SiteGuard maintains high detection precision in the presence of a limited number of false feedbacks, demonstrating the robustness of SiteGuard.

6 EXPERIENCE & LESSONS

In this section, we present our experience and key lessons learned from deploying SiteGuard .

6.1 On Detection

Lesson 1: In an encrypted world, flow context beats packet content. Our most consequential early decision was to abandon deep packet inspection (DPI) entirely. Two forces made this non-negotiable. *Technically*, TLS/QUIC decryption in the switch pipeline is infeasible—it would overwhelm the limited stages and break line rate. *Operationally*, decrypting user traffic is a privacy and legal non-starter for ISPs. We therefore committed to metadata-only analysis: packet timing, size distributions, handshake patterns, and communication-graph structure. This was not without cost—metadata reduces detection granularity—but in production it proved decisive: over the A/B test, SiteGuard sustained >90% precision on encrypted threats (botnet, brute-force, and web attacks) where the Suricata baseline, blind to encrypted flows, fell 20–30 pp behind. The lesson for the community: as the encrypted fraction of traffic approaches 100%, the question is no longer *how to see inside packets* but *how richly we can model the flow around them.*

Lesson 2: Beyond NetFlow—header aggregation is a dead end for encrypted threats. Currently, many vendors' core routers (e.g., Cisco, Huawei) offer NetFlow-based network intrusion detection system (NIDS) capabilities. While NetFlow-based solutions are suitable for high-speed wide-area networks, they only aggregate packet header information and therefore cannot detect anomalies in encrypted traffic. Given that encrypted traffic now exceeds 90% of overall network traffic, NetFlow-based inline NIDS is seldom deployed in our production environments. In fact, based on our operational experience, NetFlow is more effective for network management—particularly for billing purposes—than for security monitoring. We opted for programmable switches for two main reasons. First, as shown in Table 3, compared to FPGA-based frameworks, programmable switch-based approaches are more cost-effective and consume less power. Second, their programmability makes it easier for us to modify or update feature extraction logic as needed.

Methods	Money	Power	Delay	Encrypted	Robust
NSFOCUS [44]	$2,412/Gbps	10Wtts/Gbps	30 sec.	False	False
SiteGuard	$3/Gbps	0.1Wtts/Gbps	10 sec.	True	True

Table 3: SiteGuard and FPGA-based methods.

6.2 On Adaptation

Lesson 3: A static model degrades predictably, so adaptation must be continuous. The dominant source of operational overhead was concept drift in benign traffic, induced by the launch of new applications, peering and routing changes, and campus bandwidth expansion. Over the three-year deployment we recorded 19 drift events of sufficient magnitude to affect detection, averaging roughly one every eight weeks. Representative triggers included the rollout of a campus-wide video-conferencing platform and a major client-OS update that altered prevailing TLS fingerprints; in each case the false-positive rate exceeded our operating threshold β within 12–48 hours. SiteGuard's online update mechanis restored precision to its pre-drift level within 3–5 update cycles (typically under six hours) without operator-initiated retraining.

Lesson 4: Operator feedback is noisy and delayed, and the update loop must be designed accordingly. The online update of SiteGuard depends on the operators confirming detections through tickets and on the end users reporting missed attacks. In practice, this feedback diverged substantially from the idealized oracle assumed in much of the literature. The security expertise of the operator varied considerably between the sites, and we estimated that 8–12% of the returned labels were incorrect. Two design choices allowed the loop to remain effective under these conditions. First, alerts are aggregated into per-incident tickets rather than per-flow labels, bounding daily triage to a few minutes per ticket and keeping the workload tractable for a single operator. Second, the online update rule is made robust to label noise: in a controlled experiment, injecting up to 10% false feedback reduced precision by less than 2%. The lesson is that human-in-the-loop detection at ISP scale must be engineered around imperfect feedback—using aggregation to bound operator workload and noise-tolerant updates to bound the impact of erroneous labels.

Lesson 5: Site heterogeneity requires per-site, not fleet-wide, adaptation. We initially assumed a single trained model could be replicated unchanged across all sites. Production traffic profiles, however, differed sufficiently between campus types—residential, research-intensive, and administrative—that a uniform model yielded inconsistent precision, varying by as much as 5 across sites. Specializing the detection threshold and online weights per site—bootstrapped from a shared base model but refined using each site's own benign traffic and feedback—was necessary to achieve the stable, above-90% precision observed across all sites. The lesson is that at ISP scale the appropriate unit of adaptation is the individual site rather than the fleet as a whole.

6.3 On Reliability

Lesson 6: Carrier-grade reliability is a precondition, not a feature. Deploying programmable switch-based intrusion detection in ISP networks requires solutions that guarantee operational stability. Our implementation focused on three reliability-first approaches. First, we designed a fail-safe bypass mechanism that automatically disables detection logic during system anomalies, ensuring continuous packet forwarding. Second, we established a staged deployment framework, beginning with sampled traffic validation before full-scale implementation. Third, we created real-time consistency verification that continuously compares detection outcomes between programmable and legacy paths. Crucially, we maintained dual control paths—preserving traditional management interfaces alongside programmable functions for seamless fallback.

6.4 Looking Ahead

Two directions emerged from production. First, *from detection to investigation*: we have begun correlating SiteGuard alerts with external knowledge bases (IP geolocation, IP-attribute and reputation databases) to turn raw alerts into contextualized incidents, and we see correlation analysis as the path from detection toward automated threat investigation and response. Second, *inline as the backbone default*: as Tb/s-class backbones proliferate, the storage and compute cost of out-of-band traffic replication becomes prohibitive; our experience suggests inline, switch-resident detection is the practical reference architecture for ultra-high-speed infrastructure.

7 RELATED WORKS

Malicious traffic detection. Online traffic anomaly detection methods are widely investigated [2, 39, 46, 47, 57]. [46, 47] combine the online and offline machine learning methods to make the methods easier to train and able to identify new traffic based on the prediction result of online and offline algorithms. [2] proposed a fast activation function that helps the deep neural network converge faster, which is then used to detect DDoS attacks. [39] provide an online schema for training the traffic detection model, which achieves a low training overhead. [57] leverage deep dictionary learning to achieve online anomaly detection for encrypted traffic. However, unlike SiteGuard, these methods, they do not guarantee that the model will converge.

Machine learning-based NIDS. Machine learning-based Network Intrusion Detection Systems (NIDS) can detect zero-day attacks and achieve a higher detection accuracy compared with traditional rule-based methods. [10, 32] adopted statistical machine learning methods, i.e., wavelet analysis and PCA, to detect malware traffic in an unsupervised manner. Although, these methods have a small detection overhead. They often have a low detection accuracy and are sensitive to outliers. [30, 36] detected malicious traffic, i.e., DDoS and Black Holes, by using flow-level statistical features and Naive Bayes Classifiers. However, these methods could not achieve real-time detection. [39] proposed Kitsune, which aims to reduce the detection overhead by using lightweight deep learning models, i.e., autoencoders. [14, 17, 42, 61, 65, 66] advocates using programmable switches to detect.

8 CONCLUSION

In this paper, we developed SiteGuard, an inline NIDS leveraging the programmable switch to identify the malicious traffic entering the ISP sites. *This work does not raise any ethicalsissues.*

ACKNOWLEDGMENTS

The research is supported by the National Natural Science Foundation of China under Grant No. 62572269, No. 62394322 and No.92582109, and by the National Key R&D Program of China (Grant No. 2024YFB2906800). The correspoding authors are Guyue Liu and Xia Yin.

REFERENCES

[1] Tamer AbuHmed, Abedelaziz Mohaisen, and DaeHun Nyang. 2008. A Survey on Deep Packet Inspection for Intrusion Detection Systems. (2008). arXiv:cs.CR/0803.0037 https://arxiv.org/abs/0803.0037

[2] Khaled Alrawashdeh and Carla Purdy. 2018. Fast Activation Function Approach for Deep Learning Based Online Anomaly Intrusion Detection. In *4th IEEE International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, and IEEE International Conference on Intelligent Data and Security, BigDataSecurity/HPSC/IDS 2018, Omaha, NE, USA, May 3-5, 2018*. IEEE, 5–13. https://doi.org/10.1109/BDS/HPSC/IDS18.2018.00016

[3] João Romeiras Amado, Francisco Pereira, David Pissarra, Salvatore Signorello, Miguel Correia, and Fernando M. V. Ramos. 2024. Peregrine: ML-based Malicious Traffic Detection for Terabit Networks. (2024). arXiv:cs.NI/2403.18788 https://arxiv.org/abs/2403.18788

[4] Blake Anderson, Subharthi Paul, and David McGrew. 2018. Deciphering malware's use of TLS (without decryption). *Journal of Computer Virology and Hacking Techniques* 14, 3 (2018), 195–211.

[5] AT&T. 2026. AT&T Business. https://www.business.att.com/portfolios/cybersecurity.html.. (2026).

[6] Barefoot Networks. 2022. Tofino Switch. https://www.barefootnetworks.com/products/brief-tofino/. (2022).

[7] Diogo Barradas, Nuno Santos, Luís Rodrigues, Salvatore Signorello, Fernando M. V. Ramos, and André Madeira. 2021. FlowLens: Enabling Efficient Flow Classification for ML-based Network Security Applications. In *Proceedings of the 28th Network and Distributed System Security Symposium*. San Diego, CA, USA.

[8] Kevin Borders, Jonathan Springer, and Matthew Burnside. 2012. Chimera: a declarative language for streaming network traffic analysis. In *Proceedings of the 21st USENIX Conference on Security Symposium (Security'12)*. USENIX Association, USA, 19.

[9] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. 2007. Denial of Service or Denial of Security? *(CCS '07)*. Association for Computing Machinery, New York, NY, USA, 92–102. https://doi.org/10.1145/1315245.1315258

[10] Christian Callegari, Stefano Giordano, Michele Pagano, and Teresa Pepe. 2011. Combining sketches and wavelet analysis for multi time-scale network anomaly detection. *Computers & Security* 30, 8 (2011), 692–704.

[11] China Telecom. 2026. China Telecom. https://www.ctamericas.com/services/other-services/ddos-services/. (2026).

[12] Colonial Pipeline ransomware attack. 2012. Colonial Pipeline ransomware attack. https://www.cisa.gov/news-events/news/attack-colonial-pipeline-what-weve-learned-what-weve-done-over-past-two-years. (2012).

[13] cyberark. 2024. Identity Security Threat Landscape Report. https://www.cyberark.com/threat-landscape/. (2024).

[14] Yutao Dong, Qing Li, Kaidong Wu, Ruoyu Li, Dan Zhao, Gareth Tyson, Junkun Peng, Yong Jiang, Shutao Xia, and Mingwei Xu. 2023. HorusEye: A Realtime IoT Malicious Traffic Detection Framework using Programmable Switches. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 571–588. https://www.usenix.org/conference/usenixsecurity23/presentation/dong-yutao

[15] dpdk. 2024. Data plane developer kit (dpdk). https://software.intel.com/en-us/networking/dpdk. (2024).

[16] dyn. 2023. Dyn DDOS attack. https://en.wikipedia.org/wiki/DDoS_attacks_on_Dyn. (2023).

[17] Chuanpu Fu, Qi Li, Meng Shen, and Ke Xu. 2021. Realtime Robust Malicious Traffic Detection via Frequency Domain Analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. Association for Computing Machinery, New York, NY, USA, 3431–3446. https://doi.org/10.1145/3460120.3484585

[18] Chuanpu Fu, Qi Li, and Ke Xu. 2024. Flow Interaction Graph Analysis: Unknown Encrypted Malicious Traffic Detection. *IEEE/ACM Transactions on Networking* 32, 4 (2024), 2972–2987. https://doi.org/10.1109/TNET.2024.3370851

[19] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2016. Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 58–72. https://doi.org/10.1145/2934872.2934891

[20] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. 2018. Sonata: Query-driven streaming network telemetry. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*. 357–371.

[21] Dongqi Han, Zhiliang Wang, Wenqi Chen, Kai Wang, Rui Yu, Su Wang, Han Zhang, Zhihua Wang, Minghui Jin, Jiahai Yang, Xingang Shi, and Xia Yin. 2023. Anomaly Detection in the Open World: Normality Shift Detection, Explanation, and Adaptation. In *NDSS*. https://doi.org/10.14722/ndss.2023.24830

[22] Dongqi Han, Zhiliang Wang, Wenqi Chen, Ying Zhong, Su Wang, Han Zhang, Jiahai Yang, Xingang Shi, and Xia Yin. 2021. DeepAID: Interpreting and Improving Deep Learning-based Anomaly Detection in Security Applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. Association for Computing Machinery, New York, NY, USA, 3197–3217. https://doi.org/10.1145/3460120.3484589

[23] Shangbin Han, Qianhong Wu, Han Zhang, Bo Qin, Jiankun Hu, Xingang Shi, Linfeng Liu, and Xia Yin. 2021. Log-Based Anomaly detection with robust feature extraction and online learning. *IEEE Transactions on Information Forensics and Security* 16 (2021), 2300–2311.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. (2015). arXiv:cs.CV/1512.03385

[25] Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. 2023. Edge computing with artificial intelligence: A machine learning perspective. *Comput. Surveys* 55, 9 (2023), 1–35.

[26] imperva. 2025. Protection for Telecoms and ISPs. https://www.imperva.com/solutions/protection-for-telecoms-and-isps/. (2025).

[27] Jana Iyengar and Martin Thomson. 2021. RFC 9000: QUIC: A UDP-based multiplexed and secure transport. (2021).

[28] Syed Usman Jafri, Sanjay Rao, Vishal Shrivastav, and Mohit Tawarmalani. 2024. Leo: Online ML-based Traffic Classification at Multi-Terabit Line Rate. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 1573–1591. https://www.usenix.org/conference/nsdi24/presentation/jafri

[29] Muhammad Asim Jamshed, Jihyung Lee, Sangwoo Moon, Insu Yun, Deokjin Kim, Sungryoul Lee, Yung Yi, and KyoungSoo Park. 2012. Kargus: a highly-scalable software-based intrusion detection system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*. Association for Computing Machinery, New York, NY, USA, 317–328. https://doi.org/10.1145/2382196.2382232

[30] Myungsook Klassen and Ning Yang. 2012. Anomaly based intrusion detection in wireless networks using Bayesian classifier. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*. 257–264. https://doi.org/10.1109/ICACI.2012.6463163

[31] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. 2015. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15)*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/2785956.2787478

[32] Anukool Lakhina, Mark Crovella, and Christophe Diot. 2004. Diagnosing Network-Wide Traffic Anomalies. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '04)*. Association for Computing Machinery, New York, NY, USA, 219–230. https://doi.org/10.1145/1015467.1015492

[33] Hongda Li, Hongxin Hu, Guofei Gu, Gail-Joon Ahn, and Fuqiang Zhang. 2018. vNIDS: Towards Elastic Security with Safe and Efficient Virtualization of Network Intrusion Detection Systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 17–34. https://doi.org/10.1145/3243734.3243862

[34] Jiaye Lin, Qing Li, Guorui Xie, Yong Jiang, Zhenhui Yuan, Changlin Jiang, and Yuan Yang. 2023. In-Forest: Distributed In-Network Classification with Ensemble Models. In *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. 1–12. https://doi.org/10.1109/ICNP59255.2023.10355602

[35] Chang Liu, Longtao He, Gang Xiong, Zigang Cao, and Zhen Li. 2019. FS-Net: A Flow Sequence Network For Encrypted Traffic Classification. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 1171–1179. https://doi.org/10.1109/INFOCOM.2019.8737507

[36] Tao Liu, Ailing Qi, Yuanbin Hou, and Xintan Chang. 2008. Method for Network Anomaly Detection Based on Bayesian Statistical Model with Time Slicing. In *2008 7th World Congress on Intelligent Control and Automation*. 3359–3362. https://doi.org/10.1109/WCICA.2008.4593458

[37] Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. 2021. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3829–3846. https://www.usenix.org/conference/usenixsecurity21/presentation/liu-zaoxing

[38] H. Brendan Mcmahan. 2011. A Unified View of Regularized Dual Averaging and Mirror Descent with Implicit Updates. *Computer Science* abs/1009.3240 (2011), 2011.

[39] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *CoRR* abs/1802.09089 (2018). arXiv:1802.09089 http://arxiv.org/abs/1802.09089

[40] Sankalp Mittal, Harikrishnan V., Patel Heetkumar, and Praveen Tammana. 2024. iGuard: Efficient Isolation Forest Design for Malicious Traffic Detection in Programmable Switches. In *Proceedings of the 20th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '24)*. Association for Computing Machinery, New York, NY, USA, 55–64. https://doi.org/10.1145/3680121.3697807

[41] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (MilCIS)*. IEEE, 1–6.

[42] Francesco Musumeci, Ali Can Fidanci, Francesco Paolucci, Filippo Cugini, and Massimo Tornatore. 2022. Machine-learning-enabled ddos attacks detection in p4 programmable networks. *Journal of Network and Systems Management* 30 (2022), 1–27.

[43] nmap.org. 2022. NMAP. https://nmap.org/. (2022).

[44] NSFOCUS. 2024. NSFOCUS NIDS. https://nsfocusglobal.com/products/intrusion-prevention-system/.. (2024).

[45] NSFOCUS. 2025. Value-Added Service Solution for Service Providers. https://nsfocusglobal.com/solutions-overview-3/value-added-service-solution/. (2025).

[46] Murugaraj Odiathevar, Winston KG Seah, and Marcus Frean. 2019. A hybrid online offline system for network anomaly detection. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–9.

[47] Murugaraj Odiathevar, Winston KG Seah, Marcus Frean, and Alvin Valera. 2021. An Online Offline Framework for Anomaly Scoring and Detecting New Traffic in Network Streams. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[48] Sergio Pastrana and Guillermo Suarez-Tangil. 2019. A First Look at the Crypto-Mining Malware Ecosystem: A Decade of Unrestricted Wealth. In *Proceedings of the Internet Measurement Conference (IMC '19)*. Association for Computing Machinery, New York, NY, USA, 73–86. https://doi.org/10.1145/3355369.3355576

[49] Auangkun Rangsikunpum, Sam Amiri, and Luciano Ost. 2024. BIDS: An efficient Intrusion Detection System for in-vehicle networks using a two-stage Binarised Neural Network on low-cost FPGA. *Journal of Systems Architecture* 156 (2024), 103285.

[50] Satadal Sengupta, Niloy Ganguly, Pradipta De, and Sandip Chakraborty. 2019. Exploiting diversity in android tls implementations for mobile app traffic classification. In *The World Wide Web Conference*. 1657–1668.

[51] Tal Shapira and Yuval Shavitt. 2021. FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification. *IEEE Transactions on Network and Service Management* 18, 2 (2021), 1218–1232. https://doi.org/10.1109/TNSM.2021.3071441

[52] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1 (2018), 108–116.

[53] Meng Shen, Zhenbo Gao, Liehuang Zhu, and Ke Xu. 2021. Efficient Fine-Grained Website Fingerprinting via Encrypted Traffic Analysis with Deep Learning. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. 1–10. https://doi.org/10.1109/IWQOS52092.2021.9521272

[54] John Sonchack, Oliver Michel, Adam J. Aviv, Eric Keller, and Jonathan M. Smith. 2018. Scaling Hardware Accelerated Network Monitoring to Concurrent and Dynamic Queries With *Flow. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 823–835. https://www.usenix.org/conference/atc18/presentation/sonchack

[55] Tran Ngoc Thinh, Tran Trung Hieu, Van Quoc Dung, and Surin Kittitornkun. 2012. A FPGA-based deep packet inspection engine for Network Intrusion Detection System. In *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. 1–4. https://doi.org/10.1109/ECTICon.2012.6254301

[56] M Thomson and S Turner. 2021. Rfc 9001: Using tls to secure quic. (2021).

[57] Junchi Xing and Chunming Wu. 2020. Detecting Anomalies in Encrypted Traffic via Deep Dictionary Learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 734–739. https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162940

[58] yahoo-hacks. 2012. A History of Yahoo Hacks. https://www.technologyreview.com/2016/12/15/106901/a-history-of-yahoo-hacks/. (2012).

[59] Zeus. 2022. Zeus. https://github.com/Visgean/Zeus/. (2022).

[60] Chaoyun Zhang, Xavier Costa-Pérez, and Paul Patras. 2020. Tiki-taka: Attacking and defending deep learning-based intrusion detection systems. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*. 27–39.

[61] Menghao Zhang, Guanyu Li, Shicheng Wang, Chang Liu, Ang Chen, Hongxin Hu, Guofei Gu, Qianqian Li, Mingwei Xu, and Jianping Wu. 2020. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *the 27th Network and Distributed System Security Symposium (NDSS 2020)*.

[62] Pei Zhang, Fangzhou He, Han Zhang, Jiankun Hu, Xiaohong Huang, Jilong Wang, Xia Yin, Huahong Zhu, and Yahui Li. 2023. Real-time malicious traffic detection with online isolation forest over sd-wan. *IEEE Transactions on Information Forensics and Security* 18 (2023), 2076–2090.

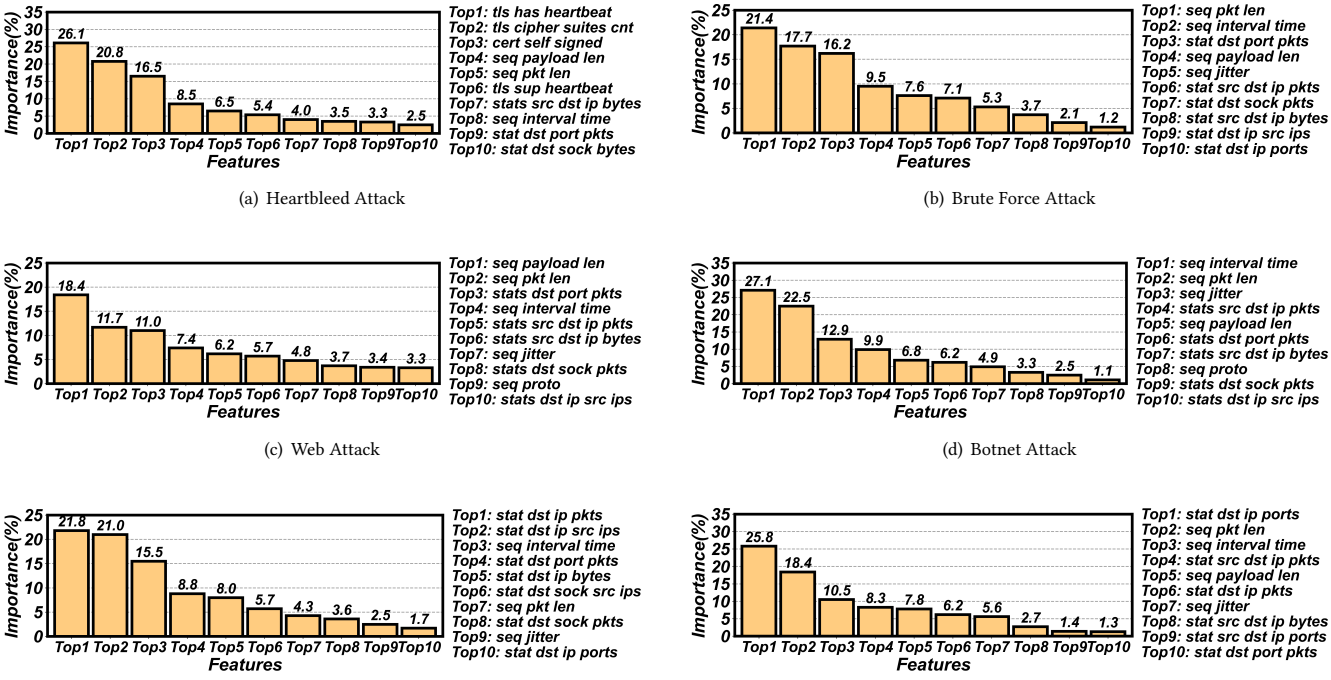
[63] Yinchao Zhang, Su Yao, Yong Feng, Kang Chen, Tong Li, Zhuotao Liu, Yi Zhao, Lexuan Zhang, Xiangyu Gao, Feng Xiong, Qi Li, and Ke Xu. 2025. Pegasus: A Universal Framework for Scalable Deep Learning Inference on the Dataplane. In *Proceedings of the ACM SIGCOMM 2025 Conference (SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 692–706. https://doi.org/10.1145/3718958.3750529

[64] Ruijie Zhao, Xianwen Deng, Zhicong Yan, Jun Ma, Zhi Xue, and Yijun Wang. 2022. Mt-flowformer: A semi-supervised flow transformer for encrypted traffic classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2576–2584.

[65] Guangmeng Zhou, Zhuotao Liu, Chuanpu Fu, Qi Li, and Ke Xu. 2023. An efficient design of intelligent network data plane. In *32nd USENIX Security Symposium (USENIX Security 23)*. 6203–6220.

[66] Huancheng Zhou, Sungmin Hong, Yangyang Liu, Xiapu Luo, Weichao Li, and Guofei Gu. 2023. Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3178–3192.

[67] Deqing Zou, Sujuan Wang, Shouhuai Xu, Zhen Li, and Hai Jin. 2021. VulDeePecker: A Deep Learning-Based System for Multiclass Vulnerability Detection. *IEEE Transactions on Dependable and Secure Computing* 18, 5 (2021), 2224–2236. https://doi.org/10.1109/TDSC.2019.2942930



(a) Heartbleed Attack

(b) Brute Force Attack

(c) Web Attack

(d) Botnet Attack

(e) DDoS Attack

(f) Port Scan Attack

Figure 27: Feature Importance Analysis. The Gini importance scores indicate that SiteGuard leverages distinct feature groups (TLS, Sequence, Statistical) depending on the attack vector.

A QUIC EXTENSION

SiteGuard treats QUIC [27] as a protocol-template extension to the same dual-plane pipeline, requiring no payload decryption and no changes to the classification algorithm or online update scheme. In the data plane, since QUIC connections may migrate across source IP/port pairs, the parser identifies QUIC packets via the UDP port and header form bit and keys sessions by the `Destination Connection ID` (DCID) rather than the 5-tuple, preserving session continuity; session termination uses an idle timeout τ_q (default 30 s) in place of TCP FIN/RST flags, after which the extracted sequence and aggregated count features are flushed to the ring buffer as usual. In the control plane, because the TLS handshake is encrypted within QUIC crypto frames, the 8 TLS semantic features are partially or wholly unavailable; SiteGuard addresses this by appending a protocol indicator bit b_p to the feature vector (b_p=1 for QUIC) and masking the TLS dimensions to zero, so the one-class classifier implicitly learns QUIC-specific decision boundaries from the remaining statistical and sequential features—optionally, the QUIC Initial packet (whose protection keys are derivable from the DCID [56]) is parsed on the control plane to recover ClientHello fields (SNI, ALPN, cipher suites) on a best-effort basis.

B FEATURE LIST AND ANALYSIS

Table 4 presents a comprehensive description of all features extracted and utilized by SiteGuard for the traffic classification task. To capture diverse aspects of network behavior, these features are organized into three distinct categories: TLS semantic features, statistic features, and sequence features.

Feature Importance Analysis. To validate the rationale and necessity of our multi-dimensional feature selection, we conducted a feature importance analysis using the Gini importance metric derived from a Random Forest classifier on the CIC-IDS-2017 dataset. This analysis aims to quantify the specific contribution of different feature groups in characterizing distinct threat vectors. As illustrated in Figure 27, the results confirm that no single feature category is sufficient; rather, a heterogeneous feature set is required to cover the diverse spectrum of attacks.

First, TLS semantic features are indispensable for protocol-level exploits. The **Heartbleed attack**, for instance, specifically exploits a vulnerability in the OpenSSL Heartbeat extension to exfiltrate memory data. Reflecting this mechanism, the feature *tls has heartbeat* exhibits the highest discriminative power in Figure 27(a) (26.1%), as the presence of Heartbeat packets is the prerequisite for this exploit. Additionally, the high ranking of *cert self signed* (16.5%) aligns with the common observation that attackers often utilize illegitimate, self-signed certificates to deploy malicious servers. This validates that deep TLS parsing on the control plane is essential, as traditional flow statistics cannot peer into these specific protocol fields.

Second, sequence features are essential for exposing the mechanical nature of automated behaviors. **Brute Force attacks** (Figure 27(b)) involve automated scripts cyclically attempting credentials, generating a rigid rhythm. Consequently, *seq pkt len* and *seq interval time* emerge as top predictors (21.4% and 17.7%), effectively exposing the inherent periodicity of these scripts. **Web Attacks** (Figure 27(c)) such as SQL Injection or XSS often inject malformed

Table 4: Description of All Extracted Features

#	Feature Name	Description
TLS Semantic Features		
1	*tls has heartbeat*	A boolean flag indicating if any TLS Heartbeat record (Type 24) appeared during the session lifetime.
2	*tls cipher suites cnt*	The count of cipher suites in the TLS Client Hello message.
3	*tls extensions cnt*	The count of extensions in the TLS Client Hello message.
4	*tls has SNI*	A boolean flag indicating the inclusion of the Server Name Indication (SNI) extension in the Client Hello message.
5	*tls sup heartbeat*	A boolean indicator of whether the Heartbeat extension is advertised in the Client Hello extension list.
6	*cert self signed*	A boolean flag indicating whether the server certificate's Issuer matches its Subject (and is not a trusted Root CA).
7	*cert validity duration*	The time difference between the certificate's *Not Before* and *Not After* fields.
8	*cert SNI in SAN*	A validation flag checking if the SNI sent by the client exists within the server certificate's Subject Alternative Names (SAN).
Statistic Features*		
9	*stats dst ip pkts*	Total number of packets received by the destination IP.
10	*stat dst ip bytes*	Total bytes of data received by the destination IP.
11	*stat dst ip ports*	Number of unique destination ports accessed on the destination IP.
12	*stat dst ip src ips*	Number of unique source IPs communicating with the destination IP.
13	*stat dst port pkts*	Total number of packets received by the destination port.
14	*stat dst port bytes*	Total bytes of data received by the destination port.
15	*stat dst port ports*	Number of unique source ports accessed on the destination port.
16	*stat dst port src ips*	Number of unique source IPs communicating with the destination port.
17	*stat src dst ip pkts*	Total number of packets transmitted by the source and the destination IP.
18	*stat src dst ip bytes*	Total bytes of data transmitted by the source and the destination IP.
19	*stat src dst ip ports*	Number of unique ports transmitted by the source and the destination IP.
20	*stat src dst ip src ips*	Alignment feature (usually 1).
21	*stat src sock pkts*	Total number of packets sent by the source socket.
22	*stat src sock bytes*	Total bytes of data sent by the source socket.
23	*stat src sock ports*	Number of unique destination ports connected to by the source socket.
24	*stat src sock src ips*	Alignment feature (usually 1).
25	*stat dst sock pkts*	Total number of packets received by the destination socket.
26	*stat dst sock bytes*	Total bytes of data received by the destination socket.
27	*stat dst sock ports*	Number of unique source ports connected to by the destination socket.
28	*stat dst sock src ips*	Number of unique source IPs communicating with the destination socket.
Sequence Features		
29	*seq pkt len*	A list of lengths for the first K packets of the session.
30	*seq interval time*	The time gaps between consecutive packets within the first K packets.
31	*seq jitter*	The change in inter-arrival times calculated from the first K packets.
32	*seq proto*	The ordered list of protocols observed in the first K packets.
33	*seq payload len*	The size of the actual data segment for the first K packets.

* These 20 base features represent statistics based on flows. In the Control Plane, they are aggregated into a vector with 80 dimensions for the detection window by calculating the value, rate, baseline, and deviation for each base feature.

strings that alter the typical payload size distribution. Thus, *seq payload len* becomes the decisive discriminator (18.4%), as it captures the distinct shape of data packets containing injection payloads without needing decryption. **Botnet attacks** (Figure 27(d)) rely on Command and Control (C&C) communications which maintain persistent connections. The high rankings of *seq interval time* (27.1%) and *seq jitter* (12.9%) in our analysis confirm that capturing the subtle timing irregularities and heartbeat intervals is the key to identifying C&C channels.

Third, statistical features remain the foundational metric for detecting volumetric and scanning attacks. **DDoS attacks** (Figure 27(e)) aim to exhaust target resources through massive volumetric flooding from distributed sources. This behavior is directly captured by the statistical features : *stat dst ip pkts* (21.8%) reflects the flooding volume, while *stat dst ip src ips* (21.0%) reveals the distributed nature of the source botnet. Similarly, **Port Scan attacks** (Figure 27(f)) represent a reconnaissance phase where an attacker probes a target to discover open services. This behavior creates a distinct one-to-many mapping where one IP accesses multiple ports, making *stat dst ip ports* (27.5%) the decisive discriminator.

The distinct importance distribution across attack types proves that that statistical, sequence, and TLS semantic features each play an irreplaceable role. While statistical features handle volumetric anomalies, the inclusion of TLS and sequence features provides the necessary granularity to detect encrypted and application level attacks.

C PROOF OF THE LEMMA

We first give the following lemma:

Lemma C.1. *Compared with the best static decision $\overrightarrow{w}^*$, the regret of the online learning algorithm is bounded by:*

$$Regret_F \leq R_{|F|}(\overrightarrow{w}^*) + \sum_{i=1}^{F}[L(\overrightarrow{w}, i) - L(\overrightarrow{w}, i+1)], \tag{6}$$

where $R(\cdot)$ is the regularization term in the loss equation, and $\overrightarrow{w}^*$ is the optimal parameters in hindsight.

Proof. First, we consider the regret against the regularized loss function, i.e.,

$$l_i(\overrightarrow{w}_j) = L(\overrightarrow{w}_j, i) + R_i(\overrightarrow{w}^*) \tag{7}$$

The regularized loss can be computed as:

$$J_F = \sum_{i=1}^{|F|}\{\sum_{j=1}^{i} l_i(\overrightarrow{w}_j) - \sum_{j=1}^{i-1} l_i(\overrightarrow{w}_j)\} - \sum_{i=1}^{|F|} l_i(\overrightarrow{w}^*) \tag{8}$$

Since, the objective of online learning is to find the $\overrightarrow{w}_{i+1}$ which has the minimum accumulated loss over previous i steps. Therefore, $\overrightarrow{w}_{i+1}$ has the smallest accumulated loss over previous i steps.

$$\begin{aligned} J_F &\le \sum_{i=1}^{|F|}\{\sum_{j=1}^{i} l_i(\overrightarrow{w}_j) - \sum_{j=1}^{i-1} l_i(\overrightarrow{w}_j)\} - \sum_{i=1}^{|F|} l_i(\overrightarrow{w}_{|F|+1}) \\ &= \sum_{i=1}^{|F|}\{\sum_{j=1}^{i} l_i(\overrightarrow{w}_j) - \sum_{j=1}^{i} l_{i+1}(\overrightarrow{w}_j)\} \end{aligned} \tag{9}$$

According to the definition of regret shown in (5), then taking (7) and (8) into (9), we could derive the following equation:

$$\begin{aligned} Regret_F \le \sum_{i=1}^{|F|} R_i(\overrightarrow{w}^*) + \sum_{i=1}^{|F|}\{\sum_{j=1}^{i}[L(\overrightarrow{w}_i, i) + R_j(\overrightarrow{w}_i)] \\ - \sum_{i=1}^{t}[L(\overrightarrow{w}_j, i+1) + R_j(\overrightarrow{w}_{i+1})]\} - \sum_{i=1}^{|F|} R_i(\overrightarrow{w}_i) \end{aligned} \tag{10}$$

$\overrightarrow{w}_i$ has the minimal accumulated loss of previous $i-1$ steps. And our regularization term $R(\cdot)$ is non-negative. Therefore, we have:

$$\sum_{j=1}^{i-1}[L(\overrightarrow{w}_j, i+1) + R_j(\overrightarrow{w}_{i+1})] \ge \sum_{j=1}^{i-1}[L(\overrightarrow{w}_i, j) + R_i(\overrightarrow{w}_i)] \tag{11}$$

With (11), we have:

$$\begin{aligned} L(\overrightarrow{w}_i, i) - L(\overrightarrow{w}, i+1) + R_i(\overrightarrow{w}_i) - R_i(\overrightarrow{w}_{i+1}) \ge \\ \sum_{i=1}^{t}[L(\overrightarrow{w}_i, j) + R_j(\overrightarrow{w}_i)] - \sum_{j=1}^{i}[L(\overrightarrow{w}_j, i+1) - R_j(\overrightarrow{w}_{i+1})] \end{aligned} \tag{12}$$

Combine(10) and (12) we have:

$$Regret_F \le R_{|F|}(\overrightarrow{w}^*) + \sum_{i=1}^{|F|}[L(\overrightarrow{w}, i) - L(\overrightarrow{w}, i+1)] \tag{13}$$

□

With Lemma C.1, we now prove Lemma 3.1.

Proof. For each weight vector $\overrightarrow{w_i}$ and $\overrightarrow{w_j}$, we have: $||\overrightarrow{w_i} - \overrightarrow{w_j}|| \le ||\overrightarrow{w_i}|| + ||\overrightarrow{w_j}||$ Since G is the maximal value of vector $\overrightarrow{w}$, we have $\overrightarrow{w_i} \le GH$, therefore,

$$||\overrightarrow{w_i} - \overrightarrow{w_j}|| \le 2GH \tag{14}$$

According to Lemma C.1, we have:

$$\begin{aligned} Regret_F &\le \frac{1}{2\eta}(\overrightarrow{c} - \overrightarrow{w}_0)^2 + \sum_{i=1}^{F}[L(\overrightarrow{w}, i) - L(\overrightarrow{w}, i+1)] \\ &\le \frac{2G^2H^2}{\eta} = 2GH^2\sqrt{|F|} \end{aligned} \tag{15}$$

□

D ADDITIONAL ANALYSIS

D.1 TESTBED RESULTS

In this part, we first show more details of the performance comparison between SiteGuard and other schemes. As shown in Figure 28, SiteGuard has consistently high detection accuracy under all datasets. Overall, SiteGuard performs up to 10% higher than the other methods.

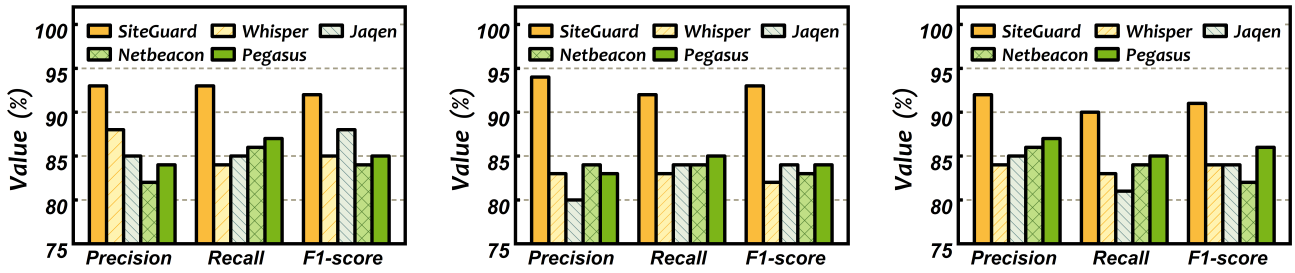
The Area Under ROC Curve (AUC) score stands for the area under the ROC curve. The ROC curve is created by plotting the true positive rate against the false positive rate at different thresholds. The average precision score (AP), the average precision score stands for the area under the precision-recall curve (PR curve). The PR curve is created by plotting the precision score against the recall score at different thresholds. Next, we select the samples that contain the specific attacks to evaluate the performance, e.g., UNSW-Fuzzers denotes that traffic only contains Fuzzer attack traffic and benign traffic. Table 5 shows the results measured with AUC and AP score. We find that SiteGuard can detect all 15 attacks with AUC-ROC ranging from 0.903 to 0.995 and AUC-PR ranging from 0.691 to 0.978, which performs up to 2× better than other schemes. These results demonstrate the model's superior discriminative power and theoretical robustness independent of detection thresholds.

To evaluate operational effectiveness at a specific deployment point, Figure 29 details the Precision and Recall breakdown for specific attack vectors. As shown in Figure 29(a), SiteGuard maintains a Precision of over 90% across all categories, ensuring the low false alarm rate necessitated by inline production environments. Critically, Figure 29(b) highlights the Recall performance. While baseline methods like Jaqen show competitive recall for volumetric attacks (e.g., DoS), their performance degrades significantly on encrypted vectors such as MPA and U2U. In contrast, SiteGuard sustains high Recall (>90%) on these threats. This empirically validates the design premise in Section 3.

D.2 The impact of 2DDFT

To strictly validate the necessity of the proposed 2D Discrete Fourier Transform (2DDFT) module, we conducted an ablation study on the CICIDS2017 and UNSW datasets by comparing the full SiteGuardmodel against two variants: one replacing 2DDFT with a 1DDFT (similar to Whisper [17]) and another removing frequency analysis entirely.

As shown in Figure 30, the proposed 2DDFT approach consistently yields superior performance in all metrics. On the CICIDS2017 dataset, SiteGuardachieves an F1-score of approximately 94%, significantly outperforming the 1D-DFT method by nearly 9% and surpassing the raw feature baseline by over 15%. Similar



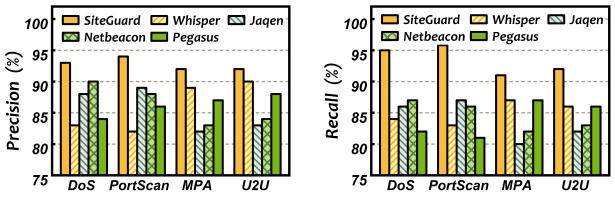
(a) CERNET data set (b) CICIDS data set (c) UNSW data set

Figure 28: Performance over different datasets.

Table 5: The detection accuracy for different attacks.

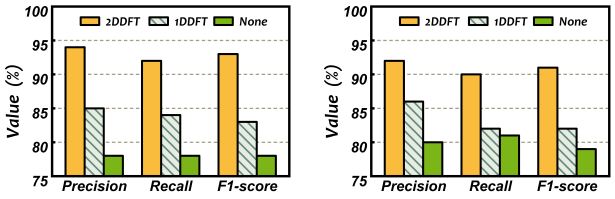
Methods	Whisper		Jaqen		Netbeacon		Pegasus		SiteGuard	
Attack types	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
CERNET-DoS	0.812	0.721	0.812	0.863	0.84	0.87	0.783	0.741	0.941	0.973
CERNET-DDoS	0.613	0.451	0.891	0.813	0.83	0.85	0.749	0.695	0.962	0.892
CERNET-PortScan	0.722	0.213	0.942	0.832	0.82	0.81	0.763	0.727	0.903	0.968
CICIDS-DDos	0.532	0.477	0.834	0.687	0.82	0.83	0.796	0.774	0.989	0.978
CICIDS-WebAttacks	0.724	0.153	0.931	0.763	0.83	0.84	0.902	0.871	0.985	0.883
CICIDS-Botnet	0.861	0.123	0.792	0.653	0.78	0.66	0.804	0.729	0.912	0.691
CICIDS-BruteForce	0.724	0.105	0.913	0.264	0.83	0.78	0.846	0.837	0.981	0.990
UNSW-Shellcode	0.981	0.558	0.961	0.97	0.82	0.79	0.843	0.794	0.941	0.901
UNSW-Reconnaissance	0.921	0.481	0.934	0.834	0.84	0.78	0.812	0.787	0.952	0.926
UNSW-Fuzzers	0.958	0.391	0.996	0.696	0.80	0.77	0.813	0.779	0.981	0.787
UNSW-Backdoor	0.966	0.089	0.939	0.914	0.78	0.77	0.734	0.709	0.995	0.924

[1] We highlight the best in • and the worst in • and AUC <0.5 means no better than a random guess.



(a) Precision (b) Recall

Figure 29: Precision and recall over different attacks.



(a) CICIDS dataset (b) UNSW dataset

Figure 30: Ablation study validating the contribution of 2D-DFT.

performance margins are evident in the UNSW dataset, confirming the module's robustness.

The significant performance gap highlights specific limitations in the baseline approaches. The inferior results of the 1DDFT method suggest that flattening the feature matrix destroys the critical joint correlations between packet metadata (e.g., size) and timing, causing complex encrypted patterns to lose their distinctiveness. Furthermore, the poor performance of the normal baseline indicates that raw sequence features are inherently too sparse and noisy for effective classification. These results empirically verify that the 2DDFT transformation is essential for concentrating distinct periodic behaviors, such as botnet heartbeats, into identifiable high-frequency components that are otherwise submerged in time-domain noise.