KnowML: Improving Generalization of Anomaly-based ML-NIDS using Symbolic Reasoning on Attack Knowledge

Abstract

Anomaly-based ML-NIDS (A-NIDS) are widely studied for their potential to detect unseen and evolving attacks. Yet, despite these advances, their suitability for deployment remains uncertain. Contemporary schemes often suffer from high false positive rates, which can overwhelm analysts with excessive alerts. Moreover, evaluations are frequently conducted on homogeneous datasets with fixed configurations, potentially overestimating real-world performance and overlooking the diversity of adversarial strategies. In our study, we observe that state-of-the-art models perform poorly on recent and previously unstudied attack variants, including those tied to newly disclosed vulnerabilities, with F1-scores dropping dramatically (on some occasions to 0%), even for well-studied families. We attribute this limitation to a knowledge gap: current systems primarily encode heuristic assumptions of designers rather than a structured representation of adversarial behavior. To tackle it, we propose KnowML, a knowledge-guided framework that integrates symbolic reasoning over knowledge graphs to systematically enumerate attack strategies and derive knowledge-augmented features. This design improves generalization across attack variants, and incorporates an efficient extractor suitable for online deployment. Experiments on both benchmark and newly collected datasets show that KnowML significantly outperforms three A-NIDS baselines and two contrastive/incremental learning methods designed to improve generalization. KnowML achieves up to 99% F1-score while keeping the False Positive Rate $\leq 0.0137\%$ whereas baseline models perform as poorly as 0% F1-score. These results underscore KnowML 's practicality for real-world deployment.

1 Introduction

Among Machine Learning-based Network Intrusion Detection Systems (ML-NIDS), Anomaly-based ones (A-NIDS) have been widely studied for their potential to detect both evolving and previously unseen variants without requiring extensive labeling to boost generalization ability [16, 99, 100]. Recent studies have shown strong performance on benchmark datasets, demonstrating progress in detecting known attacks and achieving improved generalization [79, 83, 99, 100]. Despite these successes, important challenges remain for practical deployment. Existing approaches face two key limitations. First, they often achieve high detection accuracy at the expense of elevated False Positive Rates (FPRs) [60], leading to analyst burden and alert fatigue (§2.1). Real-world networks handle hundreds of thousands to millions of packets per second, and even a 1% FPR can generate an overwhelming number of false alarms [22]. Second, evaluations are typically conducted on homogeneous datasets [39], which do not fully capture the diversity of attack strategies encountered in practice [72].

In this paper, we evaluate the suitability of A-NIDS for practical use. We extract feature spaces from four methods and two datasets that represent distinct network environments: CIC-IDS2017 [81] and CICIoT2023 [71]. We extend these datasets with recent attack variants, including exploits of newly reported vulnerabilities that are absent from prior benchmarks. Our study shows that existing A-NIDS approaches fail to generalize. This failure includes systems explicitly designed to detect variants or zero-day attacks [99, 100]. It occurs even in well-studied families such as TCP DoS, HTTP DoS, and Brute Force, where certain variants reduce F1-scores to nearly 0%. We argue that this failure arises from a fundamental *knowledge gap*. Current systems rely on ad-hoc heuristics and incomplete assumptions about attacker strategies. As a result, they use non-discriminative features that inflate FPRs (§2.1) and hinder generalization. We categorize this knowledge gap into three categories (§2.2): 1) Out-of-Dimension Blindness, 2) Throughput Bias in Feature Selection, and 3) M–N Endpoint Aggregation Failure.

To address these limitations, we propose KnowML (§3), a framework that uses Large Language Models (LLMs) to construct a Knowledge Graph (KG) of attack strategies from attack implementations; it applies symbolic reasoning over the KG to analyze existing attack strategies, their combinations,

and attackers' techniques to avoid detection. From this reasoning, we derive Knowledge-Augmented Features that capture attack semantics (§4) to enhance generalization. KNOWML's feature space improves generalization and separation of benign and malicious traffic, yielding up to a 99% increase in F1-score while maintaining a practical FPR of $\leq 0.0137\%$ (§5.1). Since we propose a new feature space, we first compare against the following baselines: CICFlowmeter [38], the standardized ML-NIDS feature set [79], and Kitsune [63]. To test whether gains could instead stem from model choice, we evaluate all feature spaces across three representative anomaly detection models: Gaussian Mixture Models (GMM) [11,13,96], Denoising Autoencoders (DA) [88], and Kitsune [63] (§5.2). Finally, we assess whether frameworks designed for generalization can replace Knowledge-Augmented Features, including incremental learning (Trident [100]) and contrastive learning (AOC-IDS [99]) (§5.4). This layered evaluation ensures fair comparison across features, models, and frameworks, showing that KNOWML improves generalization with overhead comparable to existing systems (§5.3) and that each symbolic rule contributes to learning (§5.5).

In this paper, we make the following contributions:

- We systematize the weaknesses of A-NIDS approaches, identifying three fundamental categories of knowledge gaps, arising from reliance on non-discriminative features and from an incomplete understanding of attacker strategies (§2).
- We introduce KNOWML (§4.1), a framework for systematic threat analysis that constructs a Knowledge Graph (KG) of attack strategies using Large Language Models (LLM) (§4.2). By applying symbolic reasoning over this KG (§4.3), KNOWML systematically explores the threat landscape, enumerates attack strategies, and derives the Knowledge-Augmented Features that improve generalization to unseen variants (§4.4).
- We design KNOWML for practical deployment, and demonstrate through extensive evaluation that it generalizes across diverse attack variants , increasing baseline F1-scores for variant detection up to 99% in Internet-of-Things (IoT) and in enterprise-like networks, outperforming existing approaches (§5.2). It achieves low FPRs ($\leq 0.0137\%$) (§5.1), which reduces analyst burden. KNOWML also incorporates an efficient feature extraction process with overhead comparable to state-of-the-art online systems (§5.3).
- We release KNOWML's KG, and four extracted feature spaces, two at the flow level and two at the packet level, for CICIoT2023 and CIC-IDS2017 (§5). We also collect, annotate, and release 9 recent attack variants absent from existing datasets and include them in this study.

2 Issues with Current Anomaly-based NIDS

While most supervised ML-NIDS that learn on both benign and malicious traffic have shown great promise [8, 16], they rely on diverse and high-quality labeled data for training [87]. Anomaly-based ML-NIDS (A-NIDS) offer a more practical alternative: they learn patterns from benign traffic and flag deviations as potential threats, which reduces the need for manual labeling. In this section, we show that current solutions often achieve high detection performance, only at the cost of elevated FPR.

2.1 FPR Inflation in Anomaly-based NIDS

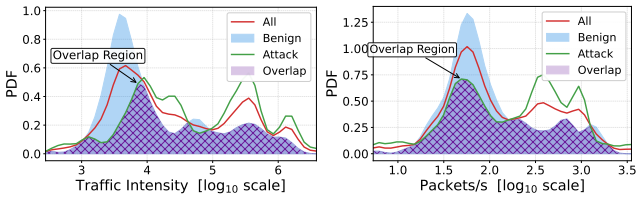
A-NIDS methods learn on benign traffic and set an anomaly threshold during training to balance Recall and FPR. Many studies report strong benchmark results because they allow relatively high FPR, which raises Recall and F1-scores [26, 60]. In network security, even a small FPR can generate too many alerts to handle, due to the base-rate fallacy as benign traffic dominates real environments [20, 22]. In deployment, these alarms overwhelm analysts, increase operational costs, and cause alert fatigue [22].

To demonstrate the impracticality of existing work, we employ two widely adopted A-NIDS models [10, 11, 101]: a Gaussian Mixture Model (GMM) [13, 96] and Kitsune [63], an ensemble of denoising autoencoders. We evaluate them on two HTTP DoS attacks, Hulk and GoldenEye, which are standard benchmark cases with consistently high results in prior studies [12, 59, 86]. Experiments are conducted on CIC-IDS2017, using only benign traffic for training and a separate holdout set for evaluation. Results are reported in Table 1, where "FPR_{tr}" is the false positive tolerance applied during training, and "Benign FPR_{te}" is the rate measured on benign traffic only in the holdout set. When FPR_{tr} is loose (up to 10%), both GMM and Kitsune achieve high Recall and F1-scores, but FPR_{te} reaches 10–11%, which is impractical for deployment. Tightening FPR_{tr} to 0.1% or less reduces FPR_{te} but raises the anomaly threshold. Many true attack instances that resemble benign traffic no longer cross this threshold and are misclassified. As a result, GMM Recall on Hulk drops from 93.72% to 10.15%, and Kitsune Recall falls from 72.16% to 56.04%. These results show that both models rely on non-discriminative features that are insufficient to separate benign and malicious traffic with high accuracy.

Cause: Over-Reliance on Non-discriminative Features Due to Knowledge Gaps in the Feature Space.. The observed degradation stems from a knowledge gap in the feature space. The available features lack the attack-relevant semantics needed to separate benign and malicious traffic at practical FPR. For example, GoldenEye floods servers with HTTP GET and POST requests while altering the User Agent string, Headers, query values, and the Referer field to masquerade as benign traffic [80]. Effective detection requires features

Table 1: **Detection Performance (%) of Two Baseline Anomaly Detectors on CIC-IDS2017 Dataset (CIDS-17).** FPR_{tr} denotes the target FPR used for threshold tuning, and FPR_{te} the rate measured on the test set.

FPR_{tr}	Anomaly Model	Metric	DoS GoldenEye	DoS Hulk	Benign FPR_{te}
≤ 10.0	GMM	Recall	93.38	93.72	11.02
		F1-score	92.71	96.26	
	Kitsune	Recall	72.62	57.15	9.98
		F1-score	78.99	72.16	
≤ 0.1	GMM	Recall	17.59	10.15	0.10
		F1-score	29.90	18.43	
	Kitsune	Recall	63.87	38.94	0.10
		F1-score	77.92	56.04	



(a) Throughput distribution (b) Packet/s distribution

Figure 1: **Overlap between Benign and Attack Traffic (DoS GoldenEye) in CIC-IDS2017 for (a) Throughput and (b) Packet Rate**. Shaded regions indicate feature similarity that makes anomaly-based detection challenging, underscoring the need for more discriminative features.

that capture variations in HTTP traffic, yet the CIC-IDS2017 feature set contains none. This forces a model to rely on provided generic features such as `bytes/sec` and `packets/sec`. These features also peak during benign high-throughput activities, e.g., video streaming or file transfers, creating significant overlap between benign and attack distributions. Figure 1 shows regions where the two classes are statistically indistinguishable. Reducing false positives then requires stricter thresholds, which also suppresses true positives in the overlap and lowers Recall. In short, A-NIDS constrained by such feature spaces cannot achieve both low FPR and high detection performance. This limitation motivates the need for features that encode attack-specific knowledge and motivates our taxonomy of knowledge gaps, as we describe next.

2.2 Three Categories of Knowledge Gap

To explain why A-NIDSs often lack discriminative power, we systematize such *knowledge gap* in feature space into three fundamental categories. Each source captures a distinct way in which missing or insufficient knowledge prevents models from separating malicious from benign activity.

Category I: Out-of-Dimension Blindness. The first source of the knowledge gap arises from the inherent limitations of human-designed feature spaces. In practice, feature selection is typically guided by the researcher's knowledge of plausible attack strategies and guided by specific detection task to be solved. Attacks exploiting these missing dimensions cannot be distinguished from benign traffic. For example, the Nkiller2 DoS attack [4] manipulates the TCP window-size field without affecting packet length or inter-arrival times. A system like Kitsune [63], which monitors jitter and packet size, sees no difference between benign and malicious flows, causing their distributions to overlap. This overlap makes High Recall unattainable at low FPR. When discriminative features are absent, forcing separation compels the model to misclassify benign variations as malicious, driving up false positives. Data-driven methods cannot solve this. For instance, feature engineering can only reweigh or recombine existing inputs, and kernel methods [70] can only remap the observed space. Neither can introduce dimensions that were never captured.

Category II: Throughput Bias in Feature Selection.. Many A-NIDS models only rely on throughput-based features such as connection counts packet rates, or data volumes [11,40,63]. While effective for detecting high-volume attacks like DDoS, these features create a significant knowledge gap for low-throughput threats. In addition, this focus overlooks attackers who deliberately match benign throughput patterns while exploiting other traffic dimensions. Attackers can easily evade such detection by making their traffic look normal in terms of throughput (e.g., mimicry attacks [92]). For example, low-rate TCP attacks [55] send brief bursts that, over time, blend in with regular traffic. Payload-padding attacks [52] adjust packet sizes to match benign patterns, and SSH brute-force attacks now often use a single persistent connection [56], avoiding detection by connection count.

Category III: M-N Endpoint Aggregation Failure.. Another source of the knowledge gap arises from feature granularity. Most A-NIDS often employ tools like CICFlowMeter [37] to extract features at the per-flow or per-endpoint level. Such a narrow view obscures broader traffic patterns that emerge only when aggregating across multiple connections or over longer timescales. This missing context is particularly problematic for distributed or coordinated attacks: while each flow may appear benign, the combined activity clearly reveals malicious intent. For instance, multi-source DDoS attacks are difficult to detect unless traffic is aggregated at the victim or service level. In this case only by aggregating traffic from multiple sources (M attackers) to a single destination ($N = 1$ victim) does the attack pattern become apparent. This *M-N Endpoint Aggregation Failure* shows how per-connection analysis conceals coordinated malicious behavior within benign-looking flows. Although the distributed nature of such attacks is well studied [9, 76, 82], it is rarely

reflected in system design (demonstrated in §5.2).

3 Motivation

The knowledge gaps in §2.2 reflect real-world challenges rather than isolated cases. Adversaries reuse mechanisms, recycle evasive behaviors, and combine older techniques into new variants. NIST vulnerability data confirms these recurring and compositional patterns [67]. Nkiller2, for example, exploits the TCP window mechanism by forcing the window size to zero. The same tactic has appeared repeatedly, with reports in 2008, 2009, and 2024 [30, 64, 66]. Evasive behaviors also return. Slowloris throttles connections to keep them open, a behavior first reported in 2007 and resurfacing in 2025 [1, 7]. Composition has now become common. A 2025 HTTP denial-of-service attack fused a 2024 header exploitation with request smuggling first observed in 2020 [4, 6, 14].

These patterns show why current feature spaces might be insufficient. Features that do not encode recurring mechanisms, returning evasive behaviors, or emerging compositions cannot separate benign from malicious traffic at low false positive rates. To this end, we propose three design principles to address this gap. The first enumerates atomic strategies to capture the range of mechanisms adversaries use for attacks (Atomic Rule §4.3.1). The second principle aims at amplifying strategies that are shared across families that would otherwise be disregarded as weak signals. This helps distinguish between the underlying behaviors such strategies underpin and the potentially overlapping benign noise (Cross-Family Rule §4.3.2). The third models interdependencies among strategies to represent compositions and adaptations that attackers exploit (Transitive Rule §4.3.3).

We present KNOWML (§4) as a framework built on these principles. KNOWML extracts attack strategies from implementation-level details and organizes them into a knowledge graph. It applies the atomic, cross-family, and transitive rules to perform symbolic reasoning. The result is a model-agnostic feature space that encodes attack-relevant semantics and bridges the identified gaps. This enables anomaly detection models to sustain high performance across diverse attacks while maintaining strict false positive constraints.

4 KNOWML: Knowledge-Guided ML

Design Goals.. Informed by our analysis in §3, the design of KNOWML pursues two core goals: 1) *Model-Agnostic Generalization*: provide a feature space adaptable to diverse anomaly detection models and deployment settings; 2) *Practicality*: achieve high detection performance under strict false positive constraints, while maintaining the computational efficiency required for online deployment.

Threat Model.. Our work focuses on active attacks that generate observable traffic, with an emphasis on classes recently documented by NIST [67] or supported by open-source implementations. This ensures our evaluation remains realistic and reproducible, covering well-known, yest still prevalent threats such as HTTP DoS, TCP DoS, and Brute Force attacks [23]. We do not consider passive threats (e.g., eavesdropping [77]) that lack distinctive traffic dynamics, or payload-dependent attacks requiring Deep Packet Inspection (DPI), which is impractical to analyze in modern encrypted traffic [44].

Background: Symbolic Reasoning and Knowledge Graphs.. In contrast to statistical and data-driven methods that learn from raw data, symbolic reasoning operates on structured, human-interpretable knowledge to deduce general rules [97]. It models relationships between abstract concepts and integrates structural and semantic knowledge, enabling models to generalise across concepts rather than relying only on surface patterns that are prone to overfitting.

A primary tool for implementing symbolic reasoning is the Knowledge Graph (KG). A KG is a directed graph composed of subject-predicate-object (s, p, o) triples, where each triple defines a relationship between entities. Subject nodes describe entities or concepts, object nodes represent linked values, and predicates define the type of relationship between the subject and the object [53]. In this paper, we formally define our generated KG as $G = (V, E)$. The nodes V encompass four entity types: strategy nodes $S = \{s_1, s_2, ..., s_n\}$, cluster nodes $C = \{c_1, c_2, ..., c_p\}$, family nodes $F = \{f_1, f_2, ..., f_k\}$, and repository nodes $R = \{r_1, r_2, ..., r_m\}$. These nodes are connected through edges E. Edges can be bidirectional, denoted as $E_{v_1 v_2} \subseteq v_1 \times v_2$, or unidirectional, denoted as $v_1 E v_2$, connecting nodes of any of the four types.

4.1 Overview of KNOWML

KNOWML is a framework for constructing Knowledge-Augmented Features that directly address the generalization challenge of A-NIDS. The overall pipeline is shown in Figure 2 and consists of the following key stages:

- **KG Construction ❶ – ❸** (§4.2): Open-source attack repositories are automatically parsed to build a KG that structurally unifies strategies, families, and implementation-level relationships.
- **Symbolic Reasoning ❹** (§4.3): Inference rules are applied over the KG to capture higher-order attacker behaviors, including atomic strategies, composite attack paths, and cross-family invariants.
- **Knowledge-Augmented Features ❺** (§4.4): The inferred knowledge is translated into features that encode atomic, composite, and family-invariant rules[1], forming a model-agnostic feature space.

[1]Throughout the paper we use the terms rule and strategy interchangeably.

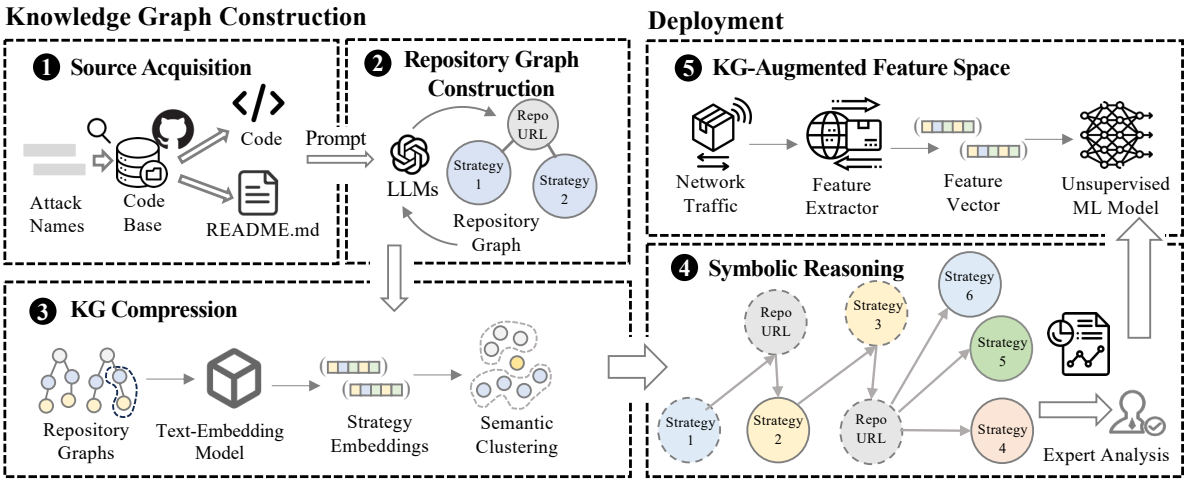


Figure 2: **Overview of the KnowML Pipeline from Attack Implementation to Feature Generation.** Given an attack name, the system retrieves its implementation and constructs repository graphs, which are then unified into a KG (Steps 1–3). Symbolic reasoning is applied over the KG to infer atomic, composite, and cross-family strategies (Step 4). These results are translated into Knowledge-Augmented Features that serve as inputs to A-NIDS for training and evaluation (Step 5).

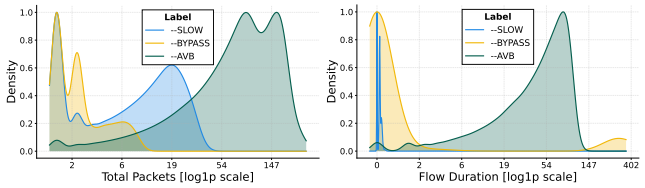
Unlike approaches limited by human heuristics or assumptions about possible attack variants, KnowML automates large-scale analysis of diverse implementations and unifies them into a structured KG. This removes designer bias and addresses the cognitive bottleneck, which is the human limitation in assimilating all attack knowledge and performing large-scale relational analysis.

4.2 KG Construction

We construct a KG by systematically extracting attacker strategies from open-source implementations. Our premise is that the configurable parameters in publicly-available attack repositories, documented in `README.md` files are explicit manifestations of attacker choices. Each "parameter" activates a concrete attack mechanism, and combinations yield composite strategies. For example:

```
$ python script.py --keep-alive --syn_flag
```

In this case, `--keep-alive` and `--syn_flag` correspond to distinct strategies, and jointly they form a composite strategy. Note that we intentionally avoid examining the specific values assigned to these parameters (e.g., `--keep-alive=0` vs. `--keep-alive=1`). This abstraction avoids overfitting to narrow configurations, and enabling generalization to unseen variants. An example of how attackers alter attack traces by adjusting setting parameters is shown in Figure 3. Building such a KG requires systematically linking repositories, parameters, and strategies. We realize this through three steps: (1) acquiring relevant repositories, (2) constructing Repository Graphs, and (3) compressing them into a unified KG.



(a) Total Packet Count (b) Flow Duration

Figure 3: **Impact of Parameter Selection on Network-level Attack Characteristics.** Different parameter settings in the HTTP DoS attack [52] yield distinct behaviors, as seen in the distributions of (a) Total Packet Count and (b) Flow Duration, highlighting how parameters shape attack strategies.

4.2.1 Source Acquisition ❶

We design a pipeline to mine open-source repositories for diverse attack implementations (see ❶ in Figure 2 and details of the acquisition process are provided in Appendix H). Given an attack name, the pipeline retrieves candidate repositories, collects source code and documentation (e.g., `README.md`), and automatically links parameter mentions to their corresponding functions or modules. The resulting data supports constructing Repository Graphs as a knowledge source for further processing.

4.2.2 Repository Graph Construction ❷

In this step, we transform individual attack implementations retrieved in ❶ into Repository Graphs. To achieve this, we define an ontology (KG schema) following best practices in ontology engineering [53]. The schema specifies four key

classes and properties: (i) *Attack* (e.g., TCP DoS), (ii) *Strategy* (a specific technique employed by attackers), (iii) *Description* (a textual explanation of how the strategy configures the attack), and (iv) *Repository Identifier* (a URI linking each strategy to its source repository). The URI is a functional property, ensuring a one-to-one mapping between strategies and repositories. We cast the extraction problem as Named Entity Recognition (NER) and employ LLMs to identify entities and relations from code and documentation at scale. Prior work has shown the effectiveness of LLMs in security-related NER tasks [36, 43, 61, 78]. We adopt a GraphRAG-style approach [36], enabling LLMs to extract the most relevant entities. Using the GPT-4o-mini model, our implementation achieves 90.91% Recall on our NER benchmark (see Appendix F for details) comparable to state-of-the-art systems especially designed for NER tasks [78]. This benchmark is based on a human-annotated dataset we created for evaluation, and the high Recall demonstrate strong agreement with human annotations. This design enables large-scale, automated construction of Repository Graphs, previously infeasible with manual analysis. The subsequent paragraphs discuss how KNOWML mitigates issues related to scalability, Recall degradation (due to increased context-window sizes), and LLM hallucinations.

Logical Consistency and Tractable Reasoning. To ensure that the constructed KG remains scalable and suitable for symbolic reasoning, we defined the ontology in accordance with OWL Lite standards [31], which guarantee polynomial-time reasoning under Description Logic (DL). We further validated the ontology using the HermiT reasoner [33], confirming that (i) no contradictions exist, (ii) all concepts are satisfiable, and (iii) all axioms are mutually compatible.

Scalability and Recall Preservation. A challenge in extracting attack strategies arises from the unpredictable length of `README.md` files and source code, which can degrade LLM Recall as context size increases [54, 58]. To address this, we implemented an iterative "gleaning" method. After the initial extraction, the LLM performs binary (`YES/NO`) checks for missed entities, guided by logit bias to enforce confirmation. If the model returns `YES`, an additional extraction round is triggered while keeping the context manageable [36].

Managing Hallucination and Noise. LLMs are known to hallucinate, i.e., generate inaccurate or fabricated entities [49]. To mitigate this, we employed a few-shot prompting strategy with explicit examples that cover the full spectrum of expected inputs: (i) *Structured inputs*, where extraction is straightforward (e.g., option lists in code or documentation: `python script.py [-p1] [-p2]` with `-p1` and `-p2` explained): (ii) *Unstructured inputs*, where strategies are described in free-form text; (iii) *Empty inputs*, where no parameters are explicitly mentioned (e.g., simply `$ python script.py`); and (iv) *Irrelevant inputs*, where search results contain unrelated repositories (e.g., an HTTP Redis pooler returned in response to an HTTP DoS query [47]). In addition, we enforced structured output templates aligned with our ontology [73], placing each entity into predefined fields. By constraining the LLM to produce structured outputs and applying few-shot prompting covering diverse scenarios, we achieve over 90% Recall in the NER task, matching human-annotated data and reducing hallucination (see Appendix F).

4.2.3 KG Compression ❸

To eliminate redundant strategies, we compress the extracted Repository Graphs by clustering similar strategies, e.g., "Randomize Ports" and "Random Port Targeting". Each strategy is embedded using its Name and Description via OpenAI's text-embedding model [74], and clustering is performed with Hierarchical Agglomerative Clustering (HAC) [65]. We adopt HAC with complete linkage, as it maximizes inter-cluster separation and yields more fine-grained clusters compared to alternatives [34], ensuring that semantically distinct strategies are not merged. Here, we prioritize inter-cluster distance over intra-cluster distance, ensuring that the worst case produces a few repetitive clusters rather than losing unique attack strategies. For each cluster C_j, we select a representative strategy S_j defined as the node with minimum cumulative embedding distance to all others in the same cluster:

$$S_j = \arg\min\{\sum_{i:s_i \in C_j} \|e_i - e_\ell\| : s_\ell \in C_j\}, \quad (1)$$

where e_i is the embedding of strategy s_i. This representative preserves the core semantics of the cluster while eliminating redundant variants.

4.3 Symbolic Reasoning ❹

We define three symbolic rules that transform the KG into Knowledge-Augmented Features: (i) atomic tactics (Atomic Rule §4.3.1), (ii) family-invariant signals (Cross-Family Rule §4.3.2), and (iii) composite strategies (Transitive Rule §4.3.3). Together, these rules form the Knowledge-Augmented Features in Figure 4.

4.3.1 Atomic Strategy Rule (R1)

The atomic rule enumerates representative instances from each clustered strategy to ensure comprehensive coverage of the analyzed strategies. This includes the examination of rare or legacy strategies and mechanisms that attackers may employ, which can potentially resurface in future attacks. Given clusters $\Pi = \{C_1, \ldots, C_p\}$ from KG compression (§4.2.3), we select one representative S_j for each cluster:

$$\mathcal{A} = \{S_j : C_j \in \Pi\}. \quad (2)$$

To preserve provenance, each S_j is linked to every repository that contained any strategy in C_j through E_{SR} (link to the

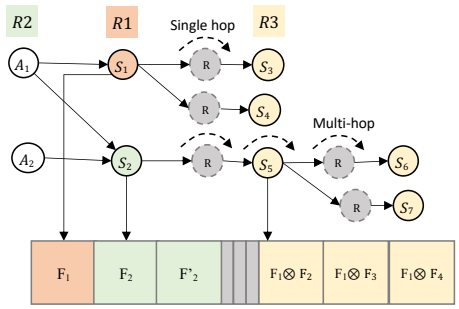


Figure 4: **Example of Symbolic Reasoning Rules (top) to derive Knowledge-Augmented Features (bottom)**. Atomic Strategy (R1), Cross-Family (R2), and Transitive Rule (R3).

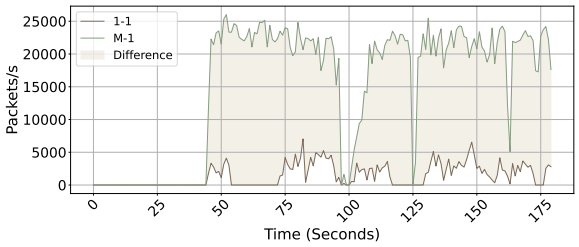


Figure 5: **WireShark Monitoring of a Family-Invariant Attack Strategy across Aggregation Levels.** The shaded region shows how weak anomalies become evident only under multi-destination monitoring, illustrating R2's principle.

source where strategy is implemented):

$$E^{\star}_{SR} = \{(S_j, r) : \exists s \in C_j \text{ such that } (s, r) \in E_{SR}\}. \quad (3)$$

4.3.2 Cross-Family Rule (R2)

The Cross-Family Rule extracts family-invariant strategies. These are behaviors that do not define a single attack family but recur across many families, suggesting they are not unique core mechanisms of an attack family. Instead, these are recurring strategies that attackers use as evasive tactics, shaping their traffic to mimic benign behavior and avoid detection. For example, delays, packet counts, and connection duration are often manipulated in this way [55]. Cross family strategies have important implications. On their own they overlap with other families and potentially with benign traffic, which reduces their discriminative power. At the same time, their recurrence across families makes them valuable for generalization because they capture common attacker behaviors. We therefore treat them not as noise but as weak yet globally relevant signals. Their diagnostic value emerges when amplified through relational and contextual reasoning in our framework (see §4.4). Formally, let S be the set of strategy nodes, F the family nodes, and E_{SF} the strategy–family edges. The set of cross-family strategies is:

$$\mathcal{S}_{\text{inv}} = \{s \in S : |\{f \in F : (s, f) \in E_{SF}\}| > 1\}. \quad (4)$$

4.3.3 Transitive Rule (R3)

The Transitive Rule captures composite strategies by identifying tactics that co-occur either within a single repository or across multiple repositories. This reflects how attackers may chain strategies, configuring multiple parameters jointly or combining tactics learned from different implementations. For instance, many TCP DoS tools allow enabling both `--ACK` and `--fragment`, yielding a *Fragmented ACK* attack that merges flooding with fragmentation [91]. To generalize such behavior, we define E_{trans} as the transitive closure of strategy–strategy relations. For a given strategy s_x, the set of transitively connected strategies is (illustrated in Figure 4):

$$TC(s_x) = \{s_y \in S : (s_x, s_y) \in E_{\text{trans}}\}. \quad (5)$$

Edges in E_{trans} arise from: (i) *Single-hop co-occurrence:* two strategies s_1 and s_2 appear in the same repository r ($\exists r \in R : (s_1, r) \in E_{SR} \wedge (r, s_2) \in E_{RS}$); (ii) *Multi-hop chaining:* strategies are connected through intermediate strategies or clusters, each linked by single-hop co-occurrence, e.g., $s_x \to s_i \to \cdots \to s_y$. This construction exposes both direct (single-hop) and indirect (multi-hop) composites, expanding the feature space to cover variants that recombine known tactics in novel ways.

4.4 Knowledge-Augmented Features ❺

The Knowledge-Augmented Features derive from the symbolic reasoning rules (R1–R3 in §4.3), each capturing a structural property of attacker behavior and yielding a corresponding feature set for A-NIDS. Formally, let $\mathcal{F}_{\phi_1}$, $\mathcal{F}_{\phi_2}$, and $\mathcal{F}_{\phi_3}$ denote the feature sets derived from R1, R2, and R3 respectively.

R1 → Atomic Strategy Features.. R1 enumerates the set of atomic strategies $\mathcal{A} = \{S_1, \ldots, S_n\}$ obtained after KG compression. Each atomic strategy naturally corresponds to a measurable network-level feature. For instance, the strategy `--packet-size` maps to the feature *packet size*, while `--flow-duration` maps to *flow duration*. These features are numerical and extracted directly from raw traffic at the flow/session level:

$$\phi_1 : \mathcal{A} \to \mathcal{F}_{\phi_1}, \quad \phi_1(S_i) = f_i. \quad (6)$$

R2 → Invariant Cross-family Features.. R2 extracts the set of family-invariant strategies $\mathcal{S}_{\text{inv}}$, which are less discriminative in isolation but valuable as weak, globally relevant signals. For each $S_i \in \mathcal{S}_{\text{inv}}$, we aggregate its per-endpoint feature values $f_{i,e}$ across all flows to the same destination, yielding an aggregated feature $f'_i \in \mathcal{F}_{\phi_2}$. This captures subtle global shifts

in common tactics (e.g., as in Figure 5 for the generic strategy `--send_packets`).

$$\phi_2 : \mathcal{F}_{\phi_1} \times \mathcal{E} \to \mathcal{F}_{\phi_2}, \quad f_i' = \mathrm{Agg}_{e \in \mathcal{E}}(f_{i,e}), \tag{7}$$

where $\mathcal{E}$ is the set of endpoints and Agg is a destination-based aggregation function.

R3 → Composite Interaction Features.. R3 identifies pairs (or chains) of strategies that are connected via transitive links in the KG, indicating their potential co-occurrence in composite attacks. For instance, in some TCP DoS tools, enabling both `--ACK` and `--fragment` produces a *Fragmented ACK* variant; the corresponding composite feature is `tcp_ack_fragment_count`. Formally, let $TC(s)$ denote the set of transitive connections of strategy s in the KG. For each $(S_i, S_j) \in TC(s)$, we construct a composite feature $f_i \otimes f_j \in \mathcal{F}_{\phi_3}$, where $\otimes$ denotes a combination operator such as multiplication or concatenation:

$$\phi_3 : TC(s) \to \mathcal{F}_{\phi_3}, \quad \phi_2((S_i, S_j)) = f_i \otimes f_j. \tag{8}$$

Feature Value Extraction.. For each feature $f_i \in \mathcal{F}_{\phi_1} \cup \mathcal{F}_{\phi_2}$, we compute four descriptive statistics, mean (μ), standard deviation (σ), cumulative sum (CS), and sum of squared residuals (SSR). To do this we adopt Welford's online algorithm [94], as well as the approach introduced in [63], which enable constant-time updates with minimal memory (details of the algorithms used for feature extraction are provided in Appendix A). For features in $\mathcal{F}_{\phi_2}$, the same statistics are first computed per conversation (H_1, H_2) ($\mathcal{F}_{\phi_1}$), and then aggregated across all conversations sharing the same destination. This enables detection of subtle global shifts in family-invariant strategies.

Final Feature Vector.. These per-feature statistics collectively form the final Knowledge-Augmented Features. At time t, the complete feature vector is:

$$\boldsymbol{v}(t) = \mathcal{F}_{\phi_1} \cup \mathcal{F}_{\phi_2} \cup \mathcal{F}_{\phi_3}, \tag{9}$$

where $\mathcal{F}_{\phi_1}$ denotes features derived from atomic strategies, $\mathcal{F}_{\phi_2}$ from aggregated family-invariant strategies, and $\mathcal{F}_{\phi_3}$ from composite interactions.

5 Evaluation

To evaluate KnowML under realistic settings, we follow best practices for A-NIDS assessment [17, 20] with a time-aware split (80% benign traffic for training, 10% for validation, 10% for testing). Unlike traditional A-NIDS methods that model benign traffic, KnowML models attack behaviors using Knowledge-Augmented Features. In this Section, we show how these features, designed to capture high-level attack concepts, improve detection. We first introduce datasets and baselines, then address the 5 research questions below:

RQ1: Does KnowML achieve effective attack detection while meeting the practical FPR requirements of realistic operational environments? (§5.1)
RQ2: Does KnowML improve generalization across different attack variants? (§5.2)
RQ3: Does KnowML deliver computational efficiency comparable to state-of-the-art online systems? (§5.3)
RQ4: Does KnowML provide comparable or superior performance to purely data-driven approaches for feature enhancement? (§5.4)
RQ5: Does each symbolic reasoning rule in KnowML contribute significantly to detection performance? (§5.5)

Prototype.. We implemented a prototype of KnowML, covering three key components: KG construction, Symbolic Reasoning, and the extraction of Knowledge-Augmented Features from network traffic. Packet capture and preprocessing rely on `tshark`. The extraction module is designed as a plug-and-play component, supporting both offline analysis (`pcap` files) and online analysis, ensuring compatibility with a variety of A-NIDSs. The KG is created from 7,853 repositories (details of the collection process are provided in Appendix H), and stored in `.graphml` format to support symbolic inference (for a summary of the extracted KnowML features, see Appendix B).

Datasets.. We evaluate KnowML on four datasets aligned with our design objectives. To capture deployment diversity, we use two standard benchmarks where performance is known to vary by environment [72]: CICIoT2023 Dataset (CIoT-23) [71], with traffic from 105 IoT devices, and CIC-IDS2017 Dataset (CIDS-17) [81], which emulates an enterprise network with heterogeneous protocols. We extract both datasets at two levels of granularity to match the extracted feature spaces: flow-level (for SFS and CICFlowMeter) and packet-level (for KIT and KnowML). For CIDS-17, we adopted the corrected labels proposed in [38]. For CIoT-23, we generated the labels following the CIoT-23 authors' guidelines, which are based on how the attacks were conducted [81], and applied the same labeling procedure as in [38]. Beyond existing benchmarks, we evaluate on two additional datasets to assess generalization to novel attack variants. SIM (Simulated Attack Variants) is generated by most recent vulnerabilities and attacks reported in NIST [67], e.g., Nkiller2 [66] (2024), HTTP Mal [7] (2025). Variants are synthesized using the `scapy` library, following established practices in network security research [15, 45, 51], while preserving the statistical and structural properties of the base environments. In contrast, CAP (Captured Attack Variants) consists of live traffic recorded from diverse real-world implementations, and includes two traces collected by external collaborators. Unlike SIM, which is controlled and property-preserving, CAP introduces variability and noise inherent to in-the-wild traffic (see Appendix C for simulation and data collection details). Together, SIM and CAP enable rigorous evaluation of KnowML under both controlled and realistic conditions, including cross-environment testing where models

Table 2: **Attack Detection at Low FPR.** F1-scores of KNOWML and baselines. **Bold** marks the best feature space per detection model, ↑ indicates KNOWML improvements, and FPR values are presented in percentage (%). Benign FPR_{te} denotes the false positive rate on the hold-out benign-only test set.

Attack	GMM				DA				KIT-ML			
	SFS	CIC	KIT	KNOWML	SFS	CIC	KIT	KNOWML	SFS	CIC	KIT	KNOWML
DoS-SYN (CIoT-23)	0.0439	0.0146	0.9173	↑ **0.9834**	0.0897	0.0593	0.9156	↑ **0.9835**	0.0930	0.0000	**0.9990**	0.9835
DoS-TCP (CIoT-23)	0.1734	0.7509	0.8800	↑ **1.0000**	0.3273	0.3329	0.9786	↑ **0.9999**	0.3357	0.0000	0.9988	↑ **0.9999**
DoS-HTTP (CIoT-23)	0.6676	0.1453	0.0010	↑ **0.9003**	0.7277	0.0742	**0.9991**	0.9000	0.6905	0.0256	**0.9748**	0.9001
Brute Force (CIoT-23)	0.0000	0.0000	0.0000	↑ **0.9727**	0.0000	0.0000	0.0000	↑ **0.9750**	0.0000	0.0000	0.0000	↑ **0.9750**
DoS Hulk (CIDS-17)	0.0228	0.1843	0.7637	↑ **0.9178**	0.0406	0.2616	0.6317	↑ **0.9677**	0.0270	0.5604	0.8641	↑ **0.9679**
DoS GoldenEye (CIDS-17)	0.3609	0.2990	0.0150	↑ **0.9934**	0.3122	0.4376	0.0005	↑ **0.9981**	0.3052	0.7792	0.0000	↑ **0.9953**
Brute Force (CIDS-17)	0.0000	0.0000	0.0000	↑ **0.9607**	0.0002	0.0000	0.0028	↑ **0.4476**	0.0000	0.0000	0.0000	↑ **0.2243**
Benign FPR_{te} (CIoT-23)	0.0980%	0.0662%	0.0296%	**0.0017%**	0.0914%	0.1200%	0.0074%	**0.0000%**	0.0147%	0.1323%	0.0030%	**0.0000%**
Benign FPR_{te} (CIDS-17)	0.1203%	0.1021%	0.1186%	**0.0137%**	0.0971%	0.0994%	0.1465%	**0.0000%**	0.0957%	0.1047%	0.1967%	**0.0076%**

are trained on benign traffic from one environment (CIoT-23 or CIDS-17) and tested on attacks in the other, following established methodologies [18, 19, 28].

Feature-Space Baselines.. To assess the contribution of KNOWML's Knowledge-Augmented Features, we compare against three widely-used alternatives that reflect distinct feature-construction paradigms: (i) Kitsune Feature Set (KIT) [63]: a manually-defined set of 100 features designed from partial attack knowledge (e.g., using jitter in IP camera streams to detect man-in-the-middle attacks), (ii) Standardized Feature Set (SFS) [79]: features derived from CISCO NetFlow standards [27]; (iii) CICFlowMeter Features (CIC) [81] (fixed version [38]): 87 generic flow statistics and protocol counters without explicit attack relevance.

A-NIDS Baselines.. We evaluate each feature space, including that of KNOWML, with three representative A-NIDS of varying complexity: (i) Gaussian Mixture Models (GMM) [11, 13, 96]—a simple probabilistic baseline that models benign traffic as a mixture of Gaussians; (ii) Denoising Autoencoder (DA) [88]—a neural model that reconstructs benign traffic patterns, with anomalies identified via high reconstruction error; and (iii) Kitsune ML[2] (KIT-ML) [63]—an ensemble of DAs trained on correlated feature subsets to exploit feature dependencies and learn richer representations.

5.1 Attack Detection Performance at Low FPR

We evaluate our Knowledge-Augmented Features against alternative feature sets across three anomaly detection models under strict operational settings where minimizing the false positive rate is essential. The evaluation covers 3,384 scenarios, combining 3 models, 94 parameters, 4 thresholds, and 4 feature spaces (detailed in Appendix D). Each model was tuned to maximize recall while keeping the FPR at or below 0.1%. Methods were optimized at their best point within this bound, rather than fixed at a single threshold, to avoid "benchmark lottery" effects where performance differences stem from arbitrary parameter choices [32].

Table 2 presents the results. Benign FPR_{te} denotes the FPR on the holdout test set for the corresponding training environment. For instance, Benign FPR_{te} (CIoT-23) refers to the value obtained when trained and tested on CIoT-23 benign data. Our features achieve higher F1-scores in most cases while consistently yielding the lowest false positive rate. KIT occasionally exceeds our F1-score but only at cost of higher FPR. The results also show the effect of background traffic complexity. KIT performs well in simpler environments such as CIoT-23, achieving an F1-score of 99.91% for DA and 97.48% for KIT-ML on HTTP DoS attacks. Its performance collapses in complex environments, for example falling to nearly 0% F1-score on the GoldenEye HTTP DoS variant in the enterprise-like CIDS-17 dataset. These findings support our claim in §2. When background traffic is diverse, models that rely on non-discriminative features cannot separate benign from attack traffic, which causes detection to fail. By constraining models away from non-discriminative patterns, KNOWML's features guide learning toward semantic indicators of attack activity. This yields both fewer false positives and more discriminative representations of malicious behavior. Overall, the balance of recall and low false positive rate shows that KNOWML outperforms existing methods under realistic deployment conditions.

5.2 Generalization on Attack Variants

To evaluate generalization of KNOWML to variants and to highlight the limitations of existing approaches (§2), we conduct experiments on diverse attack categories. The experiment is conducted using the same model parameters as in Table 2 (hence FPR omitted as it remains unchanged). Results in Table 3 show that KNOWML outperforms contemporary meth-

[2]We observed an exponential increase in feature-extraction runtime, as the number of packets grew. We identified the cause, implemented a patch, and verified that the original results were reproducible (see Appendix G). All experiments are performed using the patched Kitsune for a fair comparison.

ods on most variants, independent of the underlying anomaly model; in the few cases where KNOWML does not outperform baselines, the baselines result in a higher FPR (check with Table 2). Even a single two-layer DA trained with our Knowledge-Augmented Features achieves performance comparable to KIT-ML's ensemble of autoencoders. This suggests that well-chosen features can enable models to remain effective against new attack variants, an important property for deployment in resource-constrained environments.

Table 3 also highlights how incorporating only narrow or incomplete attack knowledge into A-NIDS feature design severely limits their capacity to detect unseen variants. First, Table 3 shows that SFS and CIC consistently perform poorly on M-N endpoint aggregation. In the rare cases where their performance appears higher, we found that the attack traffic involved little variation in source IPs, reducing the scenario to what is effectively a 1-1 attack rather than a true M–N setting. This outcome is consistent with the design of SFS and CIC: both rely on single-conversation features and therefore lack the ability to aggregate evidence across multiple sources. As a result, they fail to capture the global, distributed nature of M–N attacks. The effect is striking: SFS, for example, records an F1-score of 0% on TCP DDoS across all anomaly models. Moreover, changing the model architecture does not remedy this limitation, SFS achieves higher scores on HTTP DoS across models and SYN DoS performance remains fixed at 0% F1-score, indicating that feature space, not model type, is the primary determinant of detection capability. In contrast, KIT aggregates features at the source level, giving the model a global view that yields stronger results on M–N attacks.

Second, Table 3 confirms that existing feature spaces are highly susceptible to Out-of-Dimension blindness, with contemporary approaches often collapsing to 0% F1-score across all models. This further underscores that increasing model type alone does not improve detection when the feature space fails to capture attack-relevant dimensions. An exception arises with CICFlowMeter on the Nkiller2 variant: because it monitors TCP window statistics, the model can detect anomalies in window size values and achieve higher performance.

Similarly, models lacking attack-relevant semantics fail to generalize to non-throughput mimicry attacks. CIC attains an F1-score of 86.96% on Low-rate TCP only because its features include several TCP-level indicators (e.g., SYN flag counts) directly tied to the attack's mechanics. This attack exploits carefully timed bursts of SYN packets synchronized with the TCP retransmission-timeout mechanism [55], and CIC's feature space captures these semantics. These findings emphasize that generalization hinges less on architecture than on whether the feature space encodes attack-relevant semantics. By embedding this knowledge, KNOWML overcomes these limitations and achieves strong results at low FPR.

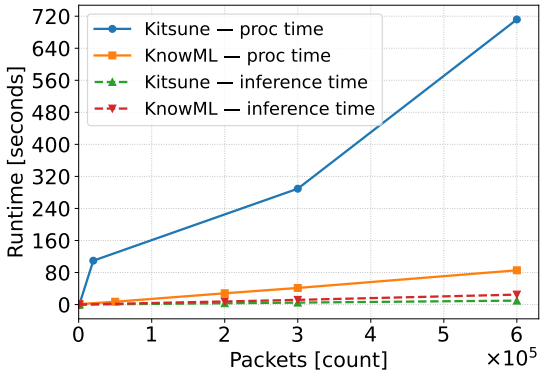


Figure 6: **Computational Efficiency.** KNOWML vs. Kitsune [63]: (a) feature extraction and (b) inference time. X-axis: input packets; Y-axis: runtime (s) with DA model.

5.3 Computational Efficiency

To assess whether our feature set and extraction algorithm are suitable for practical deployment, rather than a method that sacrifices efficiency for effectiveness. We evaluated computational efficiency along two dimensions: 1) feature extraction runtime and, 2) model inference time. We compare against Kitsune [63], which explicitly claims deployability as an online NIDS. Since the C++ implementation of Kitsune is not publicly available, we use the released Python version [62]. In addition, our evaluation is constrained to a single model rather than the full ensemble. In Kitsune, the Feature Mapper (FM) allocates the number of denoising autoencoders dynamically based on feature correlations (see [63] for details). To ensure a fair and unified comparison, we evaluate only the single underlying model for both evaluated feature spaces. KNOWML is also implemented in Python, which avoids bias from implementations in different programming language. All experiments were conducted on an Ubuntu 20.04 VM (2.6 GHz CPU, single thread, 4 GB RAM).

Figure 6 presents the results. Kitsune scales poorly with traffic volume: processing 1k packets required 0.42s, while 600k packets required 711.85s, showing a non-linear runtime increase. By contrast, KNOWML processed 600k packets in 85.59s (≈7,010 pps), despite having more than twice the feature dimensionality (214 vs. 100[3]). This throughput is comparable to the 7,500 pps reported for Kitsune's C++ implementation on a faster 3.6 GHz CPU [63]. Given that C++ implementations are typically 8–29× faster than Python under identical conditions [57], a C++ version of KNOWML would likely surpass Kitsune in feature extraction speed. In terms of inference, KNOWML is slower due to the higher feature dimensionality, achieving ≈24,836 pps compared to Kitsune's ≈61,166 pps. However, this overhead is offset by its

[3]The original paper [63] claims 115 features, whereas its artifact 100 [90].

Table 3: **Generalization on Attack Variants.** F1-scores of KNOWML and baselines. **Bold** marks the best feature space per detection model, ↑ indicates KNOWML improvements, and FPR values are presented in percentage (%).

Category	Attack	GMM				DA				KIT-ML			
		SFS	CIC	KIT	KNOWML	SFS	CIC	KIT	KNOWML	SFS	CIC	KIT	KNOWML
Out-of-Dimension	HTTP Mal (SIM)	0.0000	0.0000	0.0000	↑ **0.9844**	0.0000	0.0000	0.0000	↑ **0.9985**	0.0000	0.0000	0.0000	↑ **0.9985**
	HTTP Overflow (SIM)	0.0000	0.0000	0.0000	↑ **0.9817**	0.0000	0.0000	0.0000	↑ **0.9770**	0.0000	0.0000	0.0000	↑ **0.9800**
	Nkiller2 (SIM)	0.0000	0.8696	0.0000	↑ **0.8841**	0.0000	0.7792	0.0000	↑ **0.9190**	0.0000	0.8451	0.0000	↑ **0.9276**
	Brute Force P=0 (CAP_{ex})	0.0000	0.0000	0.0002	↑ **0.9998**	0.0000	0.0000	0.0000	↑ **0.9998**	0.0000	0.0000	0.0000	↑ **0.9998**
Non-Throughput	Fiberfox AVB (CAP)	**0.9622**	0.0000	0.0000	0.8523	0.0042	0.0000	0.0000	↑ **0.7649**	0.0042	0.0000	0.0000	↑ **0.8638**
	Fiberfox BYPASS (CAP)	0.3205	0.0000	0.2786	↑ **0.9185**	0.0000	0.5509	0.0045	↑ **0.8576**	0.0000	0.0000	0.0030	↑ **0.8431**
	Fiberfox GET (CAP)	0.0321	0.1623	0.0064	↑ **0.9766**	0.2770	0.3205	**0.9957**	0.9742	0.0298	0.1619	**0.9997**	0.9742
	Brute Force P=1 (CAP_{ex})	0.0000	0.0000	0.0000	↑ **0.9994**	0.0000	0.0000	0.0000	↑ **0.9994**	0.0000	0.0000	0.0000	↑ **0.9994**
	DDoS-Slowloris (CIoT-23)	0.0287	0.0027	0.0001	↑ **0.8114**	0.0000	0.0000	0.0015	↑ **0.7548**	0.0302	0.0001	0.5198	↑ **0.7560**
	DoS-Slowhttptest (CIDS-17)	0.0031	0.0000	0.0000	↑ **0.9214**	0.0002	0.3117	0.0018	↑ **0.9652**	0.0000	0.0281	0.0000	↑ **0.9647**
	DoS-SlowLoris (CIDS-17)	0.0568	0.0000	0.0031	↑ **0.9803**	0.0000	0.0131	0.0028	↑ **0.9714**	0.0000	0.0000	0.0000	↑ **0.9818**
	Low rate TCP (SIM)	0.0000	**0.8696**	0.0000	0.7699	0.0000	0.7792	0.0000	↑ **0.9194**	0.0000	0.8451	0.0000	↑ **0.9277**
M-N End-Point	DDoS-HTTP (CIoT-23)	0.8360	0.0022	0.0002	↑ **0.9696**	0.8502	0.0101	0.2909	↑ **0.9476**	0.8418	0.0033	0.8506	↑ **0.9476**
	DDoS-SYN (CIoT-23)	0.0000	0.9099	0.1292	↑ **0.9979**	0.0000	0.0004	0.9620	↑ **0.9979**	0.0000	0.0000	0.9873	↑ **0.9979**
	DDoS-TCP (CIoT-23)	0.0000	0.0000	0.4294	↑ **0.9997**	0.0000	0.0000	0.9992	↑ **0.9997**	0.0000	0.0000	0.9986	↑ **0.9999**
	DDoS-PSHACK (CIoT-23)	0.0000	0.2405	0.0728	↑ **0.9778**	0.0000	0.0008	0.9656	↑ **0.9753**	0.0000	0.0078	0.9831	↑ **0.9772**
	DDoS-RSTFIN (CIoT-23)	0.0000	0.7057	0.5166	↑ **1.0000**	0.0000	0.0000	0.9983	↑ **1.0000**	0.0000	0.0000	0.9950	↑ **1.0000**
	DDoS-SynIP (CIoT-23)	0.0000	0.0000	0.9992	↑ **1.0000**	0.0000	0.8000	0.9999	↑ **1.0000**	0.0000	0.8001	0.9999	**1.0000**
	DDoS-ACK Frag (CIoT-23)	**0.9768**	0.8534	0.0006	0.9438	**0.9803**	0.8442	0.5375	0.9438	**0.9802**	0.8237	0.9476	0.9438
	Fiberfox SLOW (CAP)	0.0000	0.1828	0.2637	↑ **0.9825**	0.0000	0.2514	0.9959	↑ **0.9796**	0.0000	0.1610	0.9997	↑ **0.9796**
	Fiberfox STRESS (CAP)	0.9312	0.2492	0.0000	↑ **0.9605**	0.9157	0.9914	0.9869	↑ **0.9291**	0.8814	0.9930	**0.9992**	0.9295

more efficient extraction pipeline, supporting the suitability of KNOWML's feature space for practical deployment.

5.4 Comparing with Data-Driven Approaches

We evaluate whether purely data-driven frameworks for generalization can substitute for the Knowledge-Augmented Features. This experiment tests if advances in representation learning alone are sufficient, or whether semantic features remain necessary. We focus on two recent methods that represent strong SotA approaches. *AOC-IDS* [99] employs a Cluster Repelling Contrastive loss within an autoencoder and reports over 91% zero-day detection. *Trident* [100] is an incremental learning framework that operates on existing feature spaces to improve generalization, using tScissors for outlier thresholding and tMagnifier for clustering-based class discovery. Trident and AOC-IDS have demonstrated improved generalization on existing models [24, 40, 41], making them suitable as competitive baselines for evaluating whether data-driven generalization alone can replace semantic knowledge.

We applied both methods to the Kitsune feature space (KIT), chosen because it performs well on M–N endpoint attacks but shows poor results on Out-of-Dimension and Non-Throughput variants; specifically, we focus on scenarios in which KIT performed 0% across all three detection models, to test whether advanced data-driven learning can compensate for KIT's limitations. Due to computational costs (even A100 GPUs with 80GB memory were insufficient) we trained on a sub-sampled dataset that remained 125 times larger than the original AOC-IDS dataset and comparable in size to Trident, with hold-out test data to avoid data snooping. Results in Table 4 show that both methods fail to generalize to attack variants. In the few cases where performance improves, it comes at the cost of exceptionally high false positive rates, for example 20.67% with Trident. These findings reveal a key limitation: reshaping feature space geometry without incorporating attack semantics does not achieve generalization, leaving semantic-driven attack variants effectively undetected.

Table 4: **Comparing with Data-Driven Approaches.** Per-attack F1-scores of AOC-IDS, Trident, Kitsune, and KNOWML, along with each approach's FPR on benign traffic on the same sub-sampled dataset.

Attack	F1-score			
	AOC-IDS [99]	Trident [100]	Kitsune	KnowML
Low rate TCP (SIM)	0.0000	0.0000	0.0000	**0.9247**
Fiberfox AVB (CAP)	0.1659	0.7952	0.0000	**0.8702**
Fiberfox BYPASS (CAP)	0.0536	0.9302	0.0000	**0.9565**
HTTP Overflow (SIM)	0.0000	0.0000	0.0000	**0.9703**
Nkiller2 (SIM)	0.0000	0.0000	0.0000	**0.9348**
Brute Force P=1 (CAP)	0.0180	0.9744	0.0000	**1.0000**
Benign FPR_{te}	**0.0043**	**0.2067**	**0.0011**	**0.0000**

5.5 Ablation Study

We perform an ablation study on the rule-derived features from §4.3. Using the DA model, we removed one rule-derived feature set at a time, with results shown in Table 5. Atomic Features (R1) achieved the highest average performance

Table 5: **Ablation Study**. Average F1-scores on the DA model when using features derived from individual rules. A tick (✓) denotes inclusion of features from the corresponding rule, while a cross (✗) denotes their removal.

Category	R1 (✓, ✗, ✗)	R2 (✗, ✓, ✗)	R3* (✗, ✗, ✓)	R1,R2,R3 (✓, ✓, ✓)
Out-of-Dimension	0.9373	0.7799	0.0000	0.9761
Non-Throughput-based	0.7706	0.4239	0.0000	0.9219
M-N Endpoint	0.6678	0.9158	0.1049	0.9815
Benign FPR_{te}	0.0006	0.0000	0.0000	0.0000

*R3 achieves the best performance on the composite attack, i.e., 94.38% F1 on the DDoS Ack Fragmentation attack, compared to 31.36% for R1 and 52.65% for R2. Note that R2's relatively higher performance on this attack is expected, since it also belongs to the M-N category.

(79.19% F1-score), excelling on out-of-dimension attacks by directly encoding strategy-specific dimensions. Invariant Features (R2) performed best on M–N endpoint attacks, where multi-source behavior must be aggregated, and also improved detection of non-throughput mimicry attacks (e.g., 41.46% F1-score on Slowloris) by amplifying weak but meaningful signals such as inter-arrival time invariants. Composite Features (R3) yielded the lowest overall average due to the limited number of composite attacks in the dataset, but proved highly effective when such cases appeared—for example, reaching 94.38% F1-score on the DDoS ACK Fragmentation variant. Overall, each rule contributes complementary strengths: R1 ensures broad strategy coverage, R2 encodes distributed or stealthy behaviors, and R3 captures multi-step combinations. Their integration yields improved generalization than any rule in isolation, underscoring the value of systematically embedding threat knowledge into feature design. We also verified that pairs of rules R1-R2, R2-R3, R1-R3 (for more ablation results and discussion see Appendix E).

6 Discussion

Manual Validation for Feature Mapping.. The process of mapping strategies into features, as discussed in §4.4, still requires expert input and manual validation, but can be carried out efficiently. KNOWML introduces a general framework for structural strategy analysis and, within this paper, we propose a set of features that best capture the intended strategy and corresponding attack behaviors. The feature mappings KNOWML allowed us to define already proved to be effective in our experiments. However, we encourage future research to explore alternative mappings within KNOWML. As LLMs continue to improve, we also plan to investigate their use to automate this component in future work.

Choice of Attack Families.. In this paper, we focus on TCP DoS, HTTP DoS, and SSH Brute Force. These families are both common in real-world deployments [29] and widely studied [46, 50, 85], allowing us to demonstrate that existing solutions fail to generalize even within well-studied families. While it is easier to show gaps in A-NIDS using obscure or uncommon attacks, doing so with well-known and heavily researched variants demonstrates the practical utility of our approach. Additionally, our objective is not to limit the framework to these families, but to establish a general methodology: KNOWML enables the systematic enumeration of attack strategies, the examination of relationships between strategies, and the analysis of cross-family connections. This design makes KNOWML extensible to the broader threat landscape.

Considerations on Sophisticated Adversaries.. We evaluate KNOWML in the context of generalization of detection tasks with attack variants, and do not directly assess its resilience to adversarial ML. This remains an important direction for future research. Our intuition, based on the fact that KNOWML distills behavioral features, is that knowledge-augmented features are more robust to adversarial manipulation as attackers would need to satisfy problem-space constraints [75]. We plan to explore how knowledge-augmented systems mitigate adversarial threats in future work.

Other A-NIDS Feature-Space Baselines.. Other feature-space baselines include FlowLens [24], Whisper [40], and HyperVision [41]. These methods extract few packet-level features such as size and inter-packet timing (throughput-based), then transform them through quantization, Fourier analysis, or flow-level statistics. They are like data-driven feature enhancement techniques. Our results show that Kitsune, which monitors 100 throughput-based statistics, cannot detect Out-of-Dimension or Non-Throughput attacks even when combined with SotA enhancement or incremental learning frameworks (§5.4). This suggests other approaches face the same limits. That being said, HyperVision, which uses connection degree for detection, might detect M-N endpoint attack but would suffer from the same constraint as Kitsune in detecting the other two categories of attacks. It is important to note that these systems were designed for efficiency and Tbps-scale deployment, sometimes on programmable switches, and intentionally use very few features, whereas KNOWML focuses on semantics generalization across attack variants. We plan to explore approaches to reduce feature dimensionality, further abstract behaviors, embed attack knowledge directly into models, and neuro-symbolic approaches to balance generalization with efficiency as future work.

Detecting Zero-Days.. KNOWML 's KG models diverse attack instances to distill generalizable methodology patterns rather than overfitting to specific cases. As a result, it detects variants that extend or combine known strategies but cannot anticipate entirely new vulnerabilities outside this space. When such a vulnerability emerges, however, integrating it requires only a constant-time update to the KG (e.g., adding a new edge), enabling the framework to evolve quickly as new threat intelligence becomes available.

Cost of Knowledge Graph Construction.. KnowML enables scalable, cost-efficient extraction from large codebases. For instance, extracting data (including text embeddings) for TCP DoS (from 2,333 repositories) using GPT-4o-mini costs approximately US$13. KnowML also significantly reduces human labor. Compared to an estimated 157 workdays of expert analysis for TCP DoS, HTTP DoS and Brute-Force (from a total of 7,853 repositories, assuming 10 minutes-analysis each, 8-hour workdays), KnowML completes the automated task in 26 hours with expert validation taking about a week, cutting both human effort and financial cost.

7 Related Work

KG Applications in Security.. KGs have gained traction in cybersecurity, with a range of applications from detection to knowledge management [104]. The most relevant to our work are those that directly integrate KGs into detection pipelines. For example, Yang et al. [95] construct a KG from NSL-KDD feature co-occurrence and embed it into a CNN-BiLSTM. Their approach enhances raw features through data-driven correlations, whereas KnowML introduces an attack-implementation-based feature space grounded in domain knowledge, rather than manipulations of existing features. Another approach is KnowGraph [102], which models buyer–seller interactions as a KG for anomaly detection, learning embeddings via GNNs and refining predictions with expert-provided probabilistic rules. However, KnowGraph depends on external corrections (expert-defined rules) and does not expand the knowledge base. By contrast, KnowML embeds foundational attack knowledge directly into the feature space, allowing models to capture the broader attack landscape to improve generalization.

Automated Feature Engineering in Security.. Automated feature engineering in security follows two directions. Representation learning derives latent features from data to improve detection [89, 93, 98]. Feature mining uses external knowledge to refine an existing feature space, as in FeatureSmith [103], which enhances Drebin [21] by selecting useful features. KnowML instead expands the feature space with attack-relevant features grounded in implementation details, creating new semantics rather than filtering existing ones. This approach outperforms prior methods focused only on feature enhancement (§5.4).

Works Exploring Limitation of ML-NIDS.. Recent critiques emphasize two issues: 1) dataset inadequacies [17, 38, 39, 72] and 2) flawed ML design practices [20, 35]. Flood et al. [39] describe "bad smells" in benchmark datasets, such as *poor diversity*, which produces highly dependent features. Such flaws prevent models from learning meaningful representations, limit generalization, and inflate reported performance. In contrast, we analyze limitations from a semantic perspective. We reason about how well features capture attack semantics i.e., its ability to generalize, how this affects their ability to separate benign from malicious traffic, and how it influences the number of false positive alarms.

8 Conclusion

This paper identified fundamental gaps in the SotA feature spaces adopted by Anomaly-based ML-NIDS (A-NIDS). These limitations stem from the predominant reliance on features that overfit attack-specific patterns, rather than capture behaviors shared across attack variants and families. This hinders generalization, and thus limits the effectiveness of the model. To address this gap, we proposed KnowML, a framework that builds a knowledge graph of attack strategies and applies symbolic reasoning to derive Knowledge-Augmented Features grounded in attack semantics. Our evaluation, on established benchmarks and new attack variants, revealed three categories of knowledge gaps in existing approaches. We demonstrated that KnowML closes these gaps, achieving effective generalization while maintaining low false positive rates. These results highlight the importance of incorporating attack-relevant semantics into the feature space of A-NIDS. The findings also have broader implications. They encourage the exploration of semantic reasoning in other security applications and motivate the development of adaptive defenses that evolve with attacker strategies.

Ethical Considerations

We conducted our research ethically, with the principle of beneficiance in mind. To the best of our knowledge, this work and its experiments raise no ethical concerns. We did not collect any traffic without user consent. We primarily used two public datasets (CIDS-17 and CIoT-23), and then generated two attack variants dataset: SIM via manipulation with the Scapy library of CIDS-17 and CIoT-23; CAP via traffic generation on controlled lab machines, on which launching the attacks caused no harm. For attack implementations, we crawled publicly available repositories. Our approach and methodology follows established best practices in network security research, consistent with prior work [17, 20, 39, 84].

Open Science

All code and resources required to reproduce our work and to support the claims outlined in the Contribution section are provided at https://anonymous.4open.science/r/KnowML-82B9/README.md. Specifically, we release:

- **KG Construction:** Source code for knowledge graph construction and reasoning (https://anonymous.4open.science/r/KnowML-82B9/KG_construction/README.md).

- **Feature Extraction:** Source code for extracting features from `.pcap` files, along with unit tests to verify technical correctness (https://anonymous.4open.science/r/KnowML-82B9/Feature_extraction/README.md).
- **Labeling Scripts:** Scripts to label feature spaces at different levels of granularity for CIC-IDS2017 (https://anonymous.4open.science/r/KnowML-82B9/Label_CIC_IDS2017/README.md) and CICIoT2023 (https://anonymous.4open.science/r/KnowML-82B9/Label_CIC_IoT23/README.md).
- **Kitsune Patch:** Modified Kitsune code to enable fair evaluation (https://anonymous.4open.science/r/KnowML-82B9/FE_Baseline/Kitsune_FE/README.md).
- **Simulation and Data Collection:** Scripts to simulate attacks, collect traffic, and generate labeled datasets (https://anonymous.4open.science/r/KnowML-82B9/Attack_simulation_collection/README.md).
- **Reproduction Files:** All model checkpoints and configuration files necessary to reproduce our experiments and results, ensuring transparency and reproducibility (https://anonymous.4open.science/r/KnowML-82B9/Experiments/Models_configs/README.md).
- **Dataset Access:** All experiments on CIoT-23 and CIDS-17 can be fully reproduced (including on baseline A-NIDS and feature spaces) by applying our feature extraction, labeling and detection code. CIoT-23 and CIDS-17 packet captures can be downloaded from [3,5]. For artifact evaluation and upon acceptance of the paper, we will also release all pre-extracted feature spaces for all methods and packet captures `.pcap` for SIM and CAP; we could not find a way to share all this data anonymously at the time of submission, both due to their large size (approx. 2TB in total), and the high risk of author-deanonymizing information in the packet captures or file metadata.

References

[1] CVE-2007-6750: Apache http server partial http requests dos (slowloris). https://nvd.nist.gov/vuln/detail/CVE-2007-6750, 2007. National Vulnerability Database; published December 27, 2011; last updated April 11, 2025.

[2] CVE-2009-1926 detail, 2009.

[3] Cic-ids2017 intrusion detection evaluation dataset. https://www.unb.ca/cic/datasets/ids-2017.html, 2017. Canadian Institute for Cybersecurity, University of New Brunswick.

[4] CVE-2020-35863 detail. https://nvd.nist.gov/vuln/detail/CVE-2020-35863, December 2020. Accessed: 2025-02-26.

[5] Cic iot dataset 2023. https://www.unb.ca/cic/datasets/iotdataset-2023.html, 2023. Canadian Institute for Cybersecurity, University of New Brunswick.

[6] Cve-2024-35296 detail. https://nvd.nist.gov/vuln/detail/CVE-2024-35296, July 2024. Accessed: 2025-02-26.

[7] CVE-2025-32472: Denial-of-service via slowloris-type attack on multiscan and picoscan. https://nvd.nist.gov/vuln/detail/CVE-2025-32472, April 2025. National Vulnerability Database; published 2025-04-28; last modified 2025-04-29.

[8] Emad E Abdallah, Ahmed Fawzi Otoom, et al. Intrusion detection systems using supervised machine learning techniques: a survey. *Procedia Computer Science*, 201:205–212, 2022.

[9] Devrim Akgun, Selman Hizal, and Unal Cavusoglu. A new ddos attacks intrusion detection model based on deep learning for cybersecurity. *Computers & Security*, 118:102748, 2022.

[10] Fahad Alotaibi and Sergio Maffeis. Mateen: Adaptive ensemble learning for network anomaly detection. In *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 215–234, 2024.

[11] Suresh Kumar Amalapuram, Akash Tadwai, Reethu Vinta, Sumohana S Channappayya, and Bheemarjuna Reddy Tamma. Continual learning for anomaly based network intrusion detection. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 497–505. IEEE, 2022.

[12] Suresh Kumar Amalapuram, Bheemarjuna Reddy Tamma, and Sumohana S Channappayya. Spider: A semi-supervised continual learning-based network intrusion detection system. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 571–580. IEEE, 2024.

[13] Peng An, Zhiyuan Wang, and Chunjiong Zhang. Ensemble unsupervised autoencoders and gaussian mixture model for cyberattack detection. *Information Processing & Management*, 59(2):102844, 2022.

[14] Apache Software Foundation. CVE-2025-31650: Improper Input Validation in Apache Tomcat. https://nvd.nist.gov/vuln/detail/CVE-2025-31650, 2025. Published: 2025-04-28, Last Modified: 2025-05-06.

[15] Giovanni Apruzzese, Aurore Fass, and Fabio Pierazzi. When adversarial perturbations meet concept drift: an exploratory analysis on ml-nids. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security*, pages 149–160, 2024.

[16] Giovanni Apruzzese, Pavel Laskov, Edgardo Montes de Oca, Wissam Mallouli, Luis Brdalo Rapa, Athanasios Vasileios Grammatopoulos, and Fabio Di Franco. The role of machine learning in cybersecurity. *Digital Threats: Research and Practice*, 4(1):1–38, 2023.

[17] Giovanni Apruzzese, Pavel Laskov, and Johannes Schneider. Sok: Pragmatic assessment of machine learning for network intrusion detection, 2023.

[18] Giovanni Apruzzese, Luca Pajola, and Mauro Conti. The cross-evaluation of machine learning-based network intrusion detection systems. *IEEE Transactions on Network and Service Management*, 19(4):5152–5169, 2022.

[19] Ignacio Arnaldo and Kalyan Veeramachaneni. The holy grail of" systems for machine learning" teaming humans and machine learning for detecting cyber threats. *ACM SIGKDD Explorations Newsletter*, 21(2):39–47, 2019.

[20] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988, 2022.

[21] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[22] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000.

[23] Kurt Baker. 12 most common types of cyberattacks. https://www.crowdstrike.com/en-gb/cybersecurity-101/cyberattacks/common-cyberattacks/, May 2024. Accessed: 2025-08-06.

[24] Diogo Barradas, Nuno Santos, Luís Rodrigues, Salvatore Signorello, Fernando MV Ramos, and André Madeira. Flowlens: Enabling efficient flow classification for ml-based network security applications. In *NDSS*, 2021.

[25] Sergei Bogdanov, Alexandre Constantin, Timothée Bernard, Benoit Crabbé, and Etienne Bernard. Nuner: entity recognition encoder pre-training via llm-annotated data. *arXiv preprint arXiv:2402.15343*, 2024.

[26] Marta Catillo, Antonio Pecchia, and Umberto Villano. Multicids: Anomaly-based collective intrusion detection by deep learning on iot/cps multivariate time series. *Internet of Things*, 30:101519, 2025.

[27] Cisco Systems, Inc. Cisco ios netflow, 2025. Accessed: 2025-05-25.

[28] Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Aidmar Wainakh, Max Mühlhäuser, and Simin Nadjm-Tehrani. On generating network traffic datasets with synthetic attacks for intrusion detection. *ACM Transactions on Privacy and Security (TOPS)*, 24(2):1–39, 2021.

[29] CrowdStrike, Inc. Common cyberattacks: Definitions, examples and prevention tips. https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/common-cyberattacks/, 2025. Accessed: 2025-06-04.

[30] National Vulnerability Database. CVE-2009-1926: Tcp/ip null pointer dereference vulnerability. https://nvd.nist.gov/vuln/detail/CVE-2009-1926, 2009. Accessed: 2024-10-20.

[31] Michael Dean and Guus Schreiber. Owl web ontology language guide: Feature synopsis. https://www.w3.org/TR/2004/REC-owl-features-20040210/#s4, 2004. W3C Recommendation, Accessed: 2025-06-01.

[32] Mostafa Dehghani, Yi Tay, Alexey A Gritsenko, Zhe Zhao, Neil Houlsby, Fernando Diaz, Donald Metzler, and Oriol Vinyals. The benchmark lottery. *arXiv preprint arXiv:2107.07002*, 2021.

[33] Kathrin Dentler, Ronald Cornet, Annette Ten Teije, and Nicolette De Keizer. Comparison of reasoners for large ontologies in the owl 2 el profile. *Semantic Web*, 2(2):71–87, 2011.

[34] Jairo Diaz-Rodriguez. k-llmmeans: Summaries as centroids for interpretable and scalable llm-based text clustering. *arXiv preprint arXiv:2502.09667*, 2025.

[35] Laurens D'hooge, Miel Verkerken, Bruno Volckaert, Tim Wauters, and Filip De Turck. Establishing the contaminating effect of metadata feature inclusion in machine-learned network intrusion detection models. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 23–41. Springer, 2022.

[36] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2024.

[37] Gints Engelen. Cicflowmeter: Flow-based network traffic generator. https://github.com/GintsEngelen/CICFlowMeter/tree/master, 2023. Accessed: 2024-09-23.

[38] Gints Engelen, Vera Rimmer, and Wouter Joosen. Troubleshooting an intrusion detection dataset: the cicids2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12, 2021.

[39] Robert Flood, Gints Engelen, David Aspinall, and Lieven Desmet. Bad design smells in benchmark nids datasets. In *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, pages 658–675. IEEE, 2024.

[40] Chuanpu Fu, Qi Li, Meng Shen, and Ke Xu. Realtime robust malicious traffic detection via frequency domain analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3431–3446, 2021.

[41] Chuanpu Fu, Qi Li, and Ke Xu. Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis. *arXiv preprint arXiv:2301.13686*, 2023.

[42] Álvaro García-Barragán, Alberto González Calatayud, Oswaldo Solarte-Pabón, Mariano Provencio, Ernestina Menasalvas, and Víctor Robles. Gpt for medical entity recognition in spanish. *Multimedia Tools and Applications*, pages 1–20, 2024.

[43] Hamed Babaei Giglou, Jennifer D'Souza, Felix Engel, and Sören Auer. Llms4om: Matching ontologies with large language models. *arXiv preprint arXiv:2404.10317*, 2024.

[44] Guofei Gu, Junjie Zhang, and Wenke Lee. Botsniffer: Detecting botnet command and control channels in network traffic. 2008.

[45] Dongqi Han, Zhiliang Wang, Ying Zhong, Wenqi Chen, Jiahai Yang, Shuqiang Lu, Xingang Shi, and Xia Yin. Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors. *IEEE Journal on Selected Areas in Communications*, 39(8):2632–2647, 2021.

[46] Brendon Harris and Ray Hunt. Tcp/ip security threats and attack methods. *Computer communications*, 22(10):885–897, 1999.

[47] Hiett. serverless-redis-http. https://github.com/hiett/serverless-redis-http, 2020. Accessed: 2025-06-02.

[48] Yuelin Hu, Futai Zou, Jiajia Han, Xin Sun, and Yilei Wang. Llm-tikg: Threat intelligence knowledge graph construction utilizing large language model. *Computers & Security*, 145:103999, 2024.

[49] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

[50] Mobin Javed and Vern Paxson. Detecting stealthy, distributed ssh brute-forcing. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 85–96, 2013.

[51] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. Netdiffusion: Network data augmentation through protocol-constrained traffic generation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 8(1):1–32, 2024.

[52] Andrey Kachayev. Fiberfox. https://github.com/kachayev/fiberfox. Accessed: 2025-04-18.

[53] Elisa F Kendall and Deborah L McGuinness. *Ontology engineering*. Morgan & Claypool Publishers, 2019.

[54] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. In search of needles in a 10m haystack: Recurrent memory finds what llms miss. *arXiv preprint arXiv:2402.10790*, 2024.

[55] Aleksandar Kuzmanovic and Edward W Knightly. Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 75–86, 2003.

[56] Lanjelot. Patator: A multi-purpose brute-forcer. https://github.com/lanjelot/patator/blob/master/src/patator/patator.py, 2025. Accessed: 2025-04-20.

[57] David Lion, Adrian Chiu, Michael Stumm, and Ding Yuan. Investigating managed language runtime performance: Why {JavaScript} and python are 8x and 29x slower than c++, yet java and go can be faster? In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 835–852, 2022.

[58] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[59] Ziadoon Kamil Maseer, Robiah Yusof, Nazrulazhar Bahaman, Salama A Mostafa, and Cik Feresa Mohd Foozy. Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. *IEEE access*, 9:22351–22370, 2021.

[60] Syed Ali Mehdi and Syed Zeeshan Hussain. Survey on intrusion detection system in iot network. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2022, Volume 2*, pages 721–732. Springer, 2022.

[61] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Comput. Surv.*, 56(2), September 2023.

[62] Yisroel Mirsky. Kitsune-py: A network intrusion detection system based on incremental statistics (afterimage) and an ensemble of autoencoders (kitnet). https://github.com/ymirsky/Kitsune-py, 2018. Accessed: 2025-04-15.

[63] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.

[64] MITRE Corporation. CVE-2008-4609: Tcp implementation vulnerability. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4609.

[65] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[66] National Institute of Standards and Technology. CVE-2024-32984: Yamux stream multiplexer unbounded pending frames vulnerability. https://nvd.nist.gov/vuln/detail/CVE-2024-32984.

[67] National Institute of Standards and Technology. Nist: National institute of standards and technology, 2023. Accessed on September 23, 2024.

[68] National Vulnerability Database. CVE-2021-31166: HTTP Protocol Stack Remote Code Execution Vulnerability. https://nvd.nist.gov/vuln/detail/CVE-2021-31166, 2021. Accessed: 2025-04-18.

[69] National Vulnerability Database. CVE-2024-39316: Regular Expression Denial of Service (ReDoS) in Rack::Request::Helpers. https://nvd.nist.gov/vuln/detail/CVE-2024-39316, 2024. Accessed: 2025-04-10.

[70] Harendra Singh Negi, Sushil Chandra Dimri, Bhawnesh Kumar, and Mangey Ram. Support vector machine and classification, kernel trick for separating of data points. *Mathematics in Engineering, Science & Aerospace (MESA)*, 15(2), 2024.

[71] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A Ghorbani. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13):5941, 2023.

[72] Benoit Nougnanke, Gregory Blanc, and Thomas Robert. How dataset diversity affects generalization in ml-based nids. In *ESORICS 2025-30th European Symposium on Research in Computer Security*, 2025.

[73] OpenAI. Structured outputs with language models. https://cookbook.openai.com/examples/structured_outputs_intro, 2023. Accessed: 2025-06-02.

[74] OpenAI. Embedding models, 2024. Accessed: 2025-06-06.

[75] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[76] Naveen Mohan Prajapati, Atish Mishra, and Praveen Bhanodia. Literature survey-ids for ddos attacks. In *2014 Conference on IT in Business, Industry and Government (CSIBIG)*, pages 1–3. IEEE, 2014.

[77] David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. Call me maybe: Eavesdropping encrypted {LTE} calls with {ReVoLTE}. In *29th USENIX security symposium (USENIX security 20)*, pages 73–88, 2020.

[78] Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. Gollie: Annotation guidelines improve zero-shot information-extraction. *arXiv preprint arXiv:2310.03668*, 2023.

[79] Mohanad Sarhan, Siamak Layeghy, and Marius Portmann. Towards a standard feature set for network intrusion detection system datasets. *Mobile networks and applications*, pages 1–14, 2022.

[80] J. Seidl. Goldeneye - layer 7 (keepalive+nocache) dos testing tool. https://github.com/jseidl/GoldenEye, 2014. Accessed: 2025-04-01.

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

[82] Mohammad Shurman, Rami Khrais, Abdulrahman Yateem, et al. Dos and ddos attack detection using deep learning and ids. *Int. Arab J. Inf. Technol*, 17(4A):655–661, 2020.

[83] Mahdi Soltani, Behzad Ousat, Mahdi Jafari Siavoshani, and Amir Hossein Jahangir. An adaptable deep learning-based intrusion detection system to zero-day attacks. *Journal of Information Security and Applications*, 76:103516, 2023.

[84] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[85] Cristian-Alexandru Staicu and Michael Pradel. Freezing the web: a study of {ReDoS} vulnerabilities in {JavaScript-based} web servers. In *27th USENIX security symposium (USENIX Security 18)*, pages 361–376, 2018.

[86] Wesley Joon-Wie Tann, Jackie Jin Wei Tan, Joanna Purba, and Ee-Chien Chang. Filtering ddos attacks from unlabeled network traffic data using online deep learning. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 432–446, 2021.

[87] Ankit Thakkar and Ritika Lohiya. A review of the advancement in intrusion detection datasets. *Procedia Computer Science*, 167:636–645, 2020.

[88] Ankit Thakkar and Ritika Lohiya. A review on machine learning and deep learning perspectives of ids for iot: recent updates, security issues, and challenges. *Archives of Computational Methods in Engineering*, 28(4):3211–3243, 2021.

[89] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3552–3570, 2024.

[90] GitHub user ella collins. Issue #12: The number of features in the code is not the same as in the paper. https://github.com/ymirsky/Kitsune-py/issues/12, 2021. Accessed: 2025-02-25.

[91] Mathy Vanhoef. Fragment and forge: breaking {Wi-Fi} through frame aggregation and fragmentation. In *30th USENIX security symposium (USENIX Security 21)*, pages 161–178, 2021.

[92] David Wagner and Paolo Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 255–264, 2002.

[93] Wei Wang, Songlei Jian, Yusong Tan, Qingbo Wu, and Chenlin Huang. Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions. *Computers & Security*, 112:102537, 2022.

[94] Barry Payne Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[95] Xiuzhang Yang, Guojun Peng, Dongni Zhang, and Yangqi Lv. An enhanced intrusion detection system for iot networks based on deep learning and knowledge graph. *Security and Communication Networks*, 2022(1):4748528, 2022.

[96] Hongpo Zhang, Lulu Huang, Chase Q Wu, and Zhanbo Li. An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset. *Computer Networks*, 177:107315, 2020.

[97] Jing Zhang, Bo Chen, Lingxi Zhang, Xirui Ke, and Haipeng Ding. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35, 2021.

[98] Xinchen Zhang, Running Zhao, Zhihan Jiang, Zhicong Sun, Yulong Ding, Edith C. H. Ngai, and Shuang-Hua Yang. AOC-IDS: Code for "AOC-IDS: Autonomous Online Framework with Contrastive Learning for Intrusion Detection". https://github.com/xinchen930/AOC-IDS, 2024. Accessed: 2025-08-20.

[99] Xinchen Zhang, Running Zhao, Zhihan Jiang, Zhicong Sun, Yulong Ding, Edith C.H. Ngai, and Shuang-Hua Yang. Aoc-ids: Autonomous online framework with contrastive learning for intrusion detection. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, pages 581–590, 2024.

[100] Ziming Zhao, Zhaoxuan Li, Zhuoxue Song, Wenhao Li, and Fan Zhang. Trident: A universal framework for fine-grained and class-incremental unknown traffic detection. In *Proceedings of the ACM Web Conference 2024*, pages 1608–1619, 2024.

[101] Ziming Zhao, Zhaoxuan Li, Zhuoxue Song, Fan Zhang, and Binbin Chen. Rids: Towards advanced ids via rnn model and programmable switches co-designed approaches. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 591–600. IEEE, 2024.

[102] Andy Zhou, Xiaojun Xu, Ramesh Raghunathan, Alok Lal, Xinze Guan, Bin Yu, and Bo Li. Knowgraph: Knowledge-enabled anomaly detection via logical reasoning on graph data. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 168–182, 2024.

[103] Ziyun Zhu and Tudor Dumitraş. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 767–778, 2016.

[104] Xiaohan Zou. A survey on application of knowledge graph. In *Journal of Physics: Conference Series*, volume 1487, page 012016. IOP Publishing, 2020.

A Feature Extraction Algorithm

For completeness, we provide the two core algorithms used in KNOWML's feature extraction procedure mentioned in Equation 4.4. Algorithm 1 specifies the update rule for a single feature statistic. Given a new observation x_i, it recursively updates the cumulative sum (CS), sample count (W), running mean (r), sum of squared residuals (SSR), and standard deviation (σ), without storing the entire history of observations. This formulation combines Welford's algorithm [94] with the approach in Kitsune [63], enabling constant-time updates with minimal memory requirements. The complete update algorithm is given in 2.

Algorithm 1 Register Network Statistics at Time t

function UPDATE($CS_{t-x}, W_{t-x}, SSR_{t-x}, \sigma_{t-x}, x_i$)
  $CS_t \leftarrow CS_{t-x} + x_i$
  $W_t \leftarrow W_{t-x} + 1$
  $r_t \leftarrow \frac{CS_t}{W_t}$
  $SSR_t \leftarrow SSR_{t-x} + (x_i - r_t)^2$
  $\sigma_t \leftarrow \sqrt{\frac{SSR_t}{W_t}}$
  return $CS_t, W_t, r_t, SSR_t, \sigma_t$
end function

Algorithm 2 Online Algorithm to Extract Relevant Feature Values for Packet at Time t

function EXTRACTFEATURES(packet p)
  $H_1, H_2 \leftarrow p.identifiers$ ▷ H_1: source identifier, H_2: destination identifier
  $V_{H_1,H_2} \leftarrow V_{channel}(H_1, H_2)$
  $V_{H_2} \leftarrow V_{destination}(H_2)$
  $\Delta V \leftarrow [\,]$
  for all v in V_{H_1,H_2} **do**
    $x_i \leftarrow p.f_i$ ▷ Get corresponding value to update
    $W_{t-x}, CS_{t-x}, r_{t-x}, SSR_{t-x}, \sigma_{t-x} \leftarrow v$
    $\Delta CS \leftarrow CS_{t-x}$ ▷ Store value at previous timestep
    $W_t, CS_t, r_t, SSR_t, \sigma_t \leftarrow$ UPDATE($CS_{t-x}, W_{t-x}, SSR_{t-x}, \sigma_{t-x}, x_i$)
    $v \leftarrow W_t, CS_t, r_t, SSR_t, \sigma_t$
    $\Delta CS \leftarrow |CS_t - \Delta CS|$
    $\Delta V \leftarrow \Delta V \cup \{\Delta CS\}$
  end for
  for all (i, v) in ENUMERATE(V_{H_2}) **do**
    $x_i \leftarrow \Delta V[i]$ ▷ Get change in value
    $W_{t-x}, CS_{t-x}, r_{t-x}, SSR_{t-x}, \sigma_{t-x} \leftarrow v$
    $W_t, CS_t, r_t, SSR_t, \sigma_t \leftarrow$ UPDATE($CS_{t-x}, W_{t-x}, SSR_{t-x}, \sigma_{t-x}, x_i$)
    $v \leftarrow W_t, CS_t, r_t, SSR_t, \sigma_t$ ▷ Store updated statistics
  end for
  return $V_{H_1,H_2} \| V_{H_2}$ ▷ Return updated feature values
end function

Building on this primitive, Algorithm 2 describes the end-to-end packet-level workflow for online feature extraction. For each packet, it retrieves the relevant feature sets associated with the source–destination channel and the destination host, updates their statistics by repeatedly invoking Algorithm 1, and computes incremental values to capture short-term dynamics. The updated feature vectors are then returned for subsequent analysis. In summary, Algorithm 1 provides the atomic update mechanism for individual features, while Algorithm 2 integrates this mechanism into the complete packet-processing pipeline, enabling efficient and scalable extraction of descriptive statistics from network traffic in real time.

B Overview of Extracted Features

Table 6 summarizes the extracted feature set used in our experiments. The features were designed according to the rules specified in KNOWML, covering a broad range of traffic characteristics including flow-level metrics, packet size statistics, temporal dynamics, and protocol-specific behaviors. In addition to low-level features (e.g., TCP flags, inter-arrival times, retransmissions), we also incorporate higher-level protocol semantics (e.g., HTTP request methods, SSH authentication counts) to capture application-layer behaviors.

Table 6: **Categorization of the Extracted Features.** The feature set is organized according to their underlying characteristics and descriptive statistics.

Feature Characteristic	Example Statistics	# Features
Flow Features	Duration, packet counts, size, rates, throughput, transmission rate	42
Timing Features	Inter-arrival time (IAT), Time-To-Live (TTL)	32
TCP Specific	TCP flags, header values, payload stats, TCP state values	58
HTTP Specific	Request methods, payload/header stats, response codes	42
SSH Specific	Packet size stats, authentication counts, key exchange	20
ICMP Specific	Ping requests, packet count, error count	6
Connection/State Features	Idle connections, establishment/termination attempts, request/response intervals	6
Error/Retransmission	Invalid checksums, retransmissions, packet loss	8

Table 7: Grid search parameters and number of combinations for each model, tested at FPR_{te}=0, 0.01, 0.1%.

Model	Parameters	Combinations
GMM	`n_components`: [3, 5, 7, 10, 20, 30, 40] `covariance_type`: [diag, spherical]	14
DA	`hidden_ratio`: [0.1, 0.3, 0.5] `learning_rate`: [0.0001, 0.001] `corruption_level`: [0.05, 0.1, 0.2] `activation`: [relu, tanh] `l2_reg`: [0.0001, 0.001]	72
KIT-ML (Ensemble of DA)	`maxAE`: [1, 10] `learning_rate`: [0.001, 0.01] `hidden_ratio`: [0.25, 0.5]	8

Table 8: **Ablation Study.** Average F1-scores on the DA model when using features derived from individual rules. A tick (✓) denotes inclusion of features from the corresponding rule, while a cross (✗) denotes their removal.

Category	R1,R2 (✓, ✓, ✗)	R1,R3 (✓, ✗, ✓)	R2, R3 (✗, ✓, ✓)	R1,R2,R3 (✓, ✓, ✓)
M-N Endpoint	0.9335	0.6813	0.9643	0.9815
Out-of-Dimension	0.9716	0.9378	0.7834	0.9761
Non-Throughput-based	0.8531	0.6839	0.4542	0.9219
Benign FPR_{te}	0.0000	0.0000	0.0000	0.0000

C Additional Dataset Details

In this paper, we evaluate attack variants organized by the categories defined in §2. The evaluated and collected variants breakdown are presented in Table 9. Note that some variants overlap and do not fall into a single category. In the remainder of this section, we describe the collection and simulation of the variants used in this study.

C.1 Simulated Attacks

We simulated 6 attack scenarios for both CIoT-23 and CIDS-17 datasets, including the aforementioned Nkiller2 attack. In total, we simulated 6 attacks. Each attack represents a real-world *known* vulnerability. The simulations are conducted by taking an hour of benign traffic. For each simulation we force at least 2 times increase in rate compared to the original benign traffic (if not stated otherwise). The goals here is to shows that even with an increase in the average byte rate, those attacks cannot still be captured due to insufficient disciminative power of feature space. The configuration and set up details is given as follows:

1. **Low-rate TCP**: Throttling the TCP SYN attack by sending packets in bursts of 1 second and an idle time of 1 second. Simulating a well-established Low-Rate TCP Targeted attack [55] targeting the TCP Retransmission Timeout (RTO) mechanism.

2. **Nkiller2**: Setting TCP window size of TCP packets to 0 and sending packets at 2.2x rate as the benign traffic. Simulating the vulnerability [2, 64, 66].

3. **HTTP DoS (HTTP Overflow)**: Setting HTTP Accept-Language to overflow a long valid string, causing the server to spend excessive time processing (as described in [69]). The packets are sent at 2.2x rate as benign traffic.

4. **HTTP DoS (Mal Header)**: Setting HTTP Accept-Encoding to manifold of invalid values which can cause the kernel to crash, simulating vulnerability [14, 68]. The packets are sent at 2.2x rate as benign traffic.

Table 9: Summary of Attack Variants used for evaluation.

Category	Variant Name	Dataset
Out-of-Dimension	HTTP Overflow	SIM
	HTTP Mal	SIM
	Nkiller2	SIM
	Brute Force	IoT-23
	Brute Force	CIDS-17
	Brute Force (P=0)	CAP_{ex}
	DoS Golden Eye	CIDS-17
	DoS Hulk	CIDS-17
Non-Throughput	DoS-Slowloris	CIDS-17
	DDoS-Slowloris	CIoT-23
	DoS-Slowhttptest	CIDS-17
	Fiberfox AVB	CAP
	Fiberfox BYPASS	CAP
	Low-rate TCP	SIM
	Brute Force (P=1)	CAP_{ex}
	Fiberfox GET	CAP
M-N End-Point	DoS-HTTP	CIoT-23
	DDoS-HTTP	CIoT-23
	DoS-SYN	CIoT-23
	DDoS-SYN	CIoT-23
	DoS-TCP	CIoT-23
	DDoS-TCP	CIoT-23
	DDoS-PSHACK	CIoT-23
	DDoS-RSTFIN	CIoT-23
	DDoS-SynIP	CIoT-23
	DDoS-ACK Frag	CIoT-23
	Fiberfox SLOW	CAP
	Fiberfox STRESS	CAP

C.2 Captured Attacks

The CAP dataset was collected in a controlled but realistic setting using two heterogeneous hosts: a MacBook running Sonoma 14.1 and a Dell PC running Ubuntu 14. The Dell PC was configured with an `nginx` HTTP service when HTTP DoS attacks were executed. The Dell machine acted as the victim, and all traffic was captured directly on the victim using Wireshark to minimize measurement artifacts and ensure that the traces reflect the actual network load experienced by the service. The MacBook acted as the attacker and generated malicious traffic by replaying five different HTTP DoS attack strategies using Fiberfox [52].

To avoid dataset bias and ensure realism, we (i) used commodity hardware and standard operating systems instead of emulators, (ii) deployed widely used services (nginx for HTTP) with default configurations, and (iii) allowed background system processes and benign traffic to coexist with attacks during collection. This setup prevents artificial isolation of attack traffic and preserves natural protocol dynamics such as connection establishment, response codes, and timing behavior. In addition, we avoided synthetic traffic generation tools beyond the attack scripts themselves, thereby reducing the risk of artifacts in packet structure or timing distributions. Each attack strategy was executed for 20 minutes (as specified in the `README.md` of Fiberfox [52]), resulting in six distinct attack scenarios:

1. `--AVB`: Issues HTTP GET packets into an open connection with long delays between send operations.
2. `--GET`: Sends randomly generated HTTP GET requests over an open TCP connection.
3. `--STRESS`: Sends a sequence of HTTP requests with a large body over a single open TCP connection.
4. `--BYPASS`: Sends HTTP GET requests over an open TCP connection and reads responses back.
5. `--SLOW`: Similar to STRESS, but issues HTTP requests while keeping the connection active by reading back a single byte and sending additional payload with timed delays.

CAP_{ex}. This dataset was provided by external collaborators (not produced as part of this work) and was collected using SSH Patator [56] to evaluate generalization against SSH variants.

`--SSH_PERSISTENT`: If set to 0, then it create a new connection for each attempt; otherwise, it will send multiple brute force attempt through the same connection until termination.

D Hyperparameter Selection

We performed grid search over the parameters listed in Table 7. Each model was tuned to maximize Recall under the constraint FPR $\leq 0.1\%$, selecting the lowest FPR that achieved at least 90% Recall. This avoids the "benchmark lottery" [32], ensuring that performance reflects the method rather than arbitrary hyperparameters. For GMM, we varied *n_components* $\in 3, 5, 7, 10, 20, 30, 40$ and covariance type $\in$ `diag`, `spherical` (14 combinations). For DA, we searched over hidden ratios $0.1, 0.3, 0.5$, learning rates $0.0001, 0.001$, corruption levels $0.05, 0.1, 0.2$, activations `relu`, `tanh`, and l_2 regularization $0.0001, 0.001$ (72 combinations). For KIT-ML, we tuned maxAE $1, 10$, learning rate $0.001, 0.01$, and hidden ratio $0.25, 0.5$ (8 combinations).

E Ablation Study

To further assess the effectiveness of the proposed detection rules and quantify their individual as well as combined contributions, we perform an ablation study by selectively including or excluding rule-based features. The results, summarized in Table 8, report the average F1-scores across multiple attack categories identified in §2. Each column corresponds to a different combination of rule sets, where a tick ✓indicates

inclusion and a cross ✗indicates removal of features derived from that rule. The study highlights several important findings. First, the removal of certain rules (e.g., R2 in the second column) leads to a significant drop in performance for specific attack categories, underscoring their critical role in detection. Second, while individual rules provide partial detection capability, their joint use consistently boosts performance across categories, achieving the highest scores when all three rule sets are included (last column). This demonstrates that the rules capture complementary aspects of attack behavior, and their synergy is essential for achieving near-optimal detection accuracy without increasing false positives.

F Evaluating LLM Effectiveness in Strategy Extraction

We evaluated KNOWML 's ability to enumerate and extract attack strategies through a Named Entity Recognition (NER) task, as introduced in §4.2, because accurate identification of strategy entities from heterogeneous repository documentation is essential for constructing the Knowledge Graph of attack strategies. From a corpus of 7,853 open-source implementations across three attack families, we sampled 150 repositories for detailed evaluation. This sample size balances annotation feasibility with diversity and captures four representative scenarios observed in practice: structured documentation, unstructured documentation, empty inputs to test robustness against hallucinations [49], and irrelevant content such as auxiliary repositories mistakenly retrieved (e.g., an HTTP Redis pooler returned for an HTTP DoS query [47]). We compared three models: GPT-3.5 and GPT-4o, which are general-purpose LLMs with demonstrated effectiveness in security-related extraction tasks [48], and Gollie [78], a task-specific NER model that is fine-tuned. Following standard evaluation methodology [25, 42, 78], we report Precision, Recall, and F1-Score (Table 10), where Recall quantifies the number of human-annotated strategy entities correctly identified (coverage), Precision measures the fraction of model-extracted entities that match human annotations (exactness), and F1-Score is their harmonic mean. Results show that GPT-4o achieved the highest performance with 90.91% Recall, indicating strong coverage of relevant strategies across diverse documentation formats. Precision was lower at 53.65%, but this is not a concern for our pipeline because most false positives correspond to loosely related entities such as proxies or configuration files, which are filtered efficiently during clustering and manual validation (§4.2.3), e.g., 184 FPs reduce to only four clusters for human review. High Recall is decisive, since missing strategies at this stage would introduce permanent gaps in the KG and degrade downstream feature space generalization, whereas excess entities can be pruned with low labor cost.

Table 10: Performance results for the Strategy-NER task.

Model	Precision	Recall	F1-Score
GPT-4o mini	0.5316	**0.9091**	0.6709
GPT-3.5 Turbo	0.5981	0.5541	0.5753
Gollie 13B	0.5417	0.0563	0.1020

G Patching Kitsune

When implementing baselines, we referred to the Kitsune open-source implementation at [62]. We observed an exponential increase in runtime during the feature extraction process in Kitsune. Our analysis revealed that the covariance update underwent an unintended double decay, which deviates from the algorithm described in the original paper [63], leading to the runtime growth. To ensure fair comparison, we corrected this issue, and all experiments in this work were conducted using the patched version of Kitsune. Table 12 presents performance results on the original Mirai dataset. Our results are comparable to those reported by Arp et al. [20], but differ from those in the original Kitsune paper, as a number of features had been removed by the authors [90].

H Repositories Extracted for KG

To systematically investigate the attack mutation space, we conducted a comprehensive search of GitHub repositories containing relevant implementations. We began by constructing a base set of keywords for each attack, then expanded these keywords with variations to capture different naming conventions and implementation styles. Each combination was used as a query to search GitHub, and the resulting repositories were collected into a candidate set. This structured process ensured broad coverage of potential attack implementations and minimized the risk of overlooking relevant variants. A list of searched keywords are given in Table 11.

Table 11: Attack keywords used in search and number of repositories found

Attack Name	**Base Keywords**	**Variations**	**Repos**
TCP DoS	TCP	Flood, DoS, Denial of Service, Attack	2333
HTTP DoS	HTTP, HTTP GET, HTTP POST, HTTP Request	Flood, DoS, Denial of Service, Attack, Bombardment, Overload	2935
Brute Force	SSH Dictionary, SSH Brute Force, SSH Brute-Force, SSH Password Guessing, SSH Password Cracking, SSH, SSH Login Guessing, SSH Password Spraying	Attack, Technique, Method, Tool, Hack, Crack, Attempt, Threat, Vulnerability, Exploit, Breach, Brute Force, Dictionary Attack	2585

Table 12: Comparison of Kitsune Before and After Fix at **(FPR=0)** on the Kitsune Mirai dataset

Metric	Kitsune Before Fix	Kitsune After Fix
True Positives (TP)	560,735	565,390
True Negatives (TN)	66,620	66,620
False Positives (FP)	0	0
False Negatives (FN)	81,781	77,126
Precision	1.0000	1.0000
Recall	0.8727	0.8791 ↑ **(+0.0064)**
F1-score	0.9320	0.9361 ↑ **(+0.0041)**