IDSCor: A Hybrid Intrusion Detection System Framework for Industrial Networks

Kumar Ashutosh Anand*†, Joshua Geiger‡, Santhosh Nataraj†, Martin Aman‡,
Stefanie Merz*†, Dalibor Fiala*§, Michael Heigl†
*University of West Bohemia, Pilsen, Czech Republic
{kanand, stmerz, dalfia}@kiv.zcu.cz
†Deggendorf Institute of Technology, Deggendorf, Germany
{kumar.anand, stefanie.merz, michael.heigl}@th-deg.de
‡TG alpha GmbH, Deggendorf, Germany
{j.geiger, m.aman}@tgalpha.de
§CCA Group a.s., Pilsen, Czech Republic

***Abstract*—Industrial Control Systems (ICS) are increasingly vulnerable to sophisticated cyber threats due to expanding connectivity and convergence with IT infrastructures. Traditional intrusion detection approaches for Operational Technology (OT) environments often rely either on static signatures, which are limited to known threats or supervised machine learning models, which require labeled attack data that is rarely available in real-world industrial settings. This paper presents IDSCor, a hybrid intrusion detection framework that combines packet-level signature-based inspection with flow-level unsupervised anomaly detection. The framework integrates a Snort-based IDS with whitelisting mechanism and a Kolmogorov-Arnold Network Autoencoder trained exclusively on benign network flows. To improve operational usability, a temporal alert correlation mechanism is incorporated to fuse heterogeneous alerts across packet and flow granularities into unified, correlated alerts. The proposed framework is evaluated on the ICS-Flow dataset for validating the hybrid framework for OT setting. While the unsupervised anomaly detector alone achieves an F1-score of 0.98, the detection performance of the framework is evaluated through the correlated alerts. The correlation mechanism reduced the overall alert volume by 33% and decreased false positives by approx. 84%. The correlated alert analysis further shows that all evaluated attack scenarios are detected reasonably well in the dataset. These results demonstrate that combining flow-level anomaly detection with packet-level rule-based analysis can improve detection coverage and alert quality in ICS network environments.**

***Index Terms*—Intrusion Detection, Hybrid IDS, Artificial Intelligence, Machine Learning, Anomaly Detection, Kolmogorov-Arnold Networks, Alert Correlation, Cybersecurity, Industrial Networks, Operational Technology**

I. Introduction

Cybersecurity in industrial production environments has evolved from a peripheral concern to a core research challenge. Unlike most traditional IT systems, industrial control systems (ICS) interact directly with physical processes, making cyber potentially safety-critical [1]. From a research perspective, ICS security poses a unique combination of challenges: strict real-time constraints, heterogeneous legacy systems, deterministic communication and control actions, and a shortage of labeled attack data representative of Operational Technology (OT) environments (apparently more so due to the sensitive nature of critical infrastructures [2]). And the convergence of IT and OT networks, combined with the proliferation of Industrial Internet of Things (IIoT) devices, has expanded the attack surface well beyond traditional network perimeters [3], [4].

Intrusion Detection System (IDS) serves as a crucial defensive layer in ICS environments. They are broadly categorized into two paradigms: signature-based detection, which matches observed traffic against existing databases of known attack patterns, and anomaly-based detection, which models normal system behavior and flags deviations. Signature-based IDS excel at accurately detecting known threats with lower false positive rates, yet they are fundamentally unable to detect novel or zero-day attacks whose signatures have not been catalogued [5]. Anomaly-based methods, particularly those leveraging Machine Learning (ML) and Deep Learning (DL), offer the complementary ability to detect previously unseen threats by identifying statistical deviations from learned normal baselines. However, they often suffer from elevated false positive rates and limited interpretability, which hinder their practical adoption [6].

Recent research has demonstrated the potential of Artificial Intelligence (AI) and ML algorithms to detect unknown or stealthy attacks in ICS networks [2], [7], [8]. However, a critical analysis of the current state-of-the-art reveals several persistent gaps. First, the majority of ML-based IDS research for ICS environments relies on supervised learning algorithms such as decision trees, random forests, support vector machines, and neural network classifiers that require both normal and attack data in terms of labeled data for training [9], [10]. In OT networks, such labeled datasets are exceedingly rare, making purely supervised approaches difficult to deploy. Second, while hybrid IDS architectures that combine signature-based and anomaly-based detection have been employed in the general IT domain, their application to OT-specific network flow data with unsupervised anomaly detection remains limited [11]. Third, although numerous alert correlation techniques have been explored, effectively correlating alerts from heterogeneous IDS

components remains a significant challenge [12], a capability that is essential for real-world operational deployments [13].

This research study addresses these gaps by proposing an AI-augmented hybrid IDS concept designed to improve detection performance in modern cyber environments. The main contributions of this work can be summarized as follows. First, a hybrid IDS framework is proposed which integrates a signature-based IDS at the packet level, along with an unsupervised ML-based IDS for flow-level anomaly detection, specifically targeting ICS network flow data. Here, the signature-based IDS, i.e., Snort, is augmented with a whitelisting strategy, where normal communication patterns are learned and enforced as a baseline, and deviations are treated as anomalies. This approach aligns well with ICS environments, which are characterized by relatively stable and deterministic communication patterns, making whitelisting an effective detection strategy [14]. Unlike traditional blacklist-based IDS that just rely on known attack signatures, whitelisting enforces a "deny-by-default" policy, allowing only approved interactions and thereby reducing the attack surface. Second, building up further on the studies [15]–[18], we investigate the use of a Kolmogorov-Arnold Network Autoencoder (KAN-AE) for unsupervised anomaly detection in network flows, termed as ML-IDS. These studies indicate that KAN-based models can achieve strong performance while remaining compact, data-efficient, and computationally less demanding, making them particularly suitable for resource-constrained environments. To the best of our knowledge, this study represents the first application of a KAN-based Autoencoder on OT domain network data. Finally, we introduce a temporal alert correlation mechanism that combines outputs from both the signature-based IDS and ML-IDS across different levels of network granularity. This approach enhances detection reliability through cross-validation while improving the quality of alerts by aggregating and organizing them into unified correlated alerts.

The remainder of this paper is organized as follows. Section II provides background on related work in (hybrid) IDS for industrial and general network environments. Section III details the methodology, including the hybrid architecture, signature-based IDS, ML-based IDS, and alert correlation mechanism. Section IV provides the experimental evaluation basis, including test setup, data sources, model training & testing and evaluation metrics. Section V discusses the framework's evaluation results. Finally, Section VI concludes the research findings and outlines the future work.

II. Related Work

In this research work, a hybrid IDS is defined as a combination of a signature-based IDS and an ML-based anomaly detection system designed to leverage the strengths of both approaches [6], [10], [11]. The integration of such systems into a unified detection framework enhances detection performance, reduces the incidence of false positives (FPs), and enables more effective detection in industrial settings [19]. Signature-based IDS rely on predefined patterns of known threats for rapid detection; however, these signatures require frequent updates because cyber threats and attack tactics continuously evolve over time [20] and, increasingly so, through the application of generative artificial intelligence (GenAI) by malicious actors [21]. Conversely, an unsupervised anomaly-based IDS learns normal network characteristics and identifies deviations from typical behavior, making it inherently better suited for detecting zero-day attacks or unknown malicious activities in a network.

Dehlaghi-Ghadim et al. [9] introduced the ICS-Flow dataset and evaluated supervised models such as Decision Tree (DT), Random Forest (RF), and Artificial Neural Network (ANN), achieving up to 99.5% binary detection accuracy with RF. However, the reliance on labeled training data for supervised learning models limits applicability in real OT environments. Ali et al. [22] showcased an adaptive IDS for ICS combining 'Convolutional Neural Network - Long Short-Term Memory' (CNN-LSTM) with reinforcement learning-guided transfer learning, which partially addresses data scarcity but still requires an initial labeled source. Hasan [23] presented a two-stage hybrid real-time IDS for IoT environments, where K-means clustering is followed by RF classifier. However, both stages are confined within a machine learning pipeline, rather than combining ML techniques with a separate signature-based detection system, and the second stage relies on labeled data. Similarly, Varma et al. [24] put forward a hybrid deep learning IDS that integrates a Convolutional Neural Network with the Suricata IDS. While effective, they highlight practical limitations such as high computational overhead, scalability concerns, and the risk of potential false positives. Maseno et al. [6], in a systematic review, confirm that hybrid IDS approaches can reduce false positives, but also emphasize the ongoing challenge of balancing detection accuracy with false positive rates. Srivastav et al. [11] introduced HYRIDE, a framework that combines multiple feature selection techniques with unsupervised outlier detection methods such as Local Outlier Factor (LOF) among others, and evaluated across datasets including TON_IoT[1], and CICIDS2017[2]. However, in this case, the term hybrid primarily refers to the integration of feature selection strategies rather than the combination of fundamentally different detection paradigms, such as signature-based and anomaly-based approaches. Meanwhile, our approach focuses on ICS/OT network flow data and integrates alert correlation with the hybrid framework, enabling a more cohesive system that addresses both detection performance and alert handling through temporal-oriented similarity-based correlation [25].

III. Methodology

A. Hybrid IDS Concept

The proposed hybrid framework is designed around the principle that different intrusion detection mechanisms can provide complementary insights when applied at different

[1] https://www.kaggle.com/datasets/arnobbhowmik/ton-iot-network-dataset Attribution 4.0 International (CC BY 4.0)

[2] https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset CC0: Public Domain

levels of network abstraction, while still operating on a shared data source. Figure 1 exemplies the proposed hybrid IDS architecture with its each individual components. In this framework, both the signature-based IDS and the ML-IDS analyze the same underlying network traffic, but from two distinct perspectives. The signature-based component operates at the packet level, inspecting individual packets in detail and identifying known patterns or deviations from expected communication rules. In contrast, the ML-IDS operates at the flow level, where network traffic is aggregated into flows that capture higher-level communication behavior over time. To enable this dual expression, the system employs the ICSFlowGenerator [9], which processes raw packet captures into structured network flow records. This module is positioned directly in front of the ML-based IDS, ensuring that it receives aggregated flow features while preserving consistency with the packet-level observations of the signature-based system. As a result, both detection components are logically aligned, yet operate at different granularities: one focusing on fine-grained packet semantics, and the other on higher-level behavioral patterns. After each subsystem independently performs its analysis, their outputs are combined through a correlation mechanism (described in Section III-D). This correlation stage plays a crucial role in the overall framework, as it integrates packet-level alerts and flow-level anomalies into a unified interpretation.

B. Signature-based IDS

The signature-based component of the hybrid IDS is implemented using Snort 3[3] and is responsible for detecting both known attack patterns and deviations from established network behavior.

1) Signature-based Detection: A set of static Snort rules is used to identify well-known malicious activities and insecure protocol usage. These include, for example, the detection of Telnet communication, TCP scan techniques (e.g., ACK scans), and traffic anomalies such as DNS flooding. This approach follows the traditional blacklist paradigm, where known attack signatures are explicitly defined and matched against observed network traffic.

2) Baseline-driven Whitelisting: In addition to static rules, a whitelisting-based detection mechanism is employed, specifically suited for industrial OT environments: A short recording of network traffic is used to establish a baseline of normal behavior. Based on this baseline, Snort rules are generated that define valid hosts, permitted communication relationships, and expected traffic characteristics. Any deviation from this baseline is considered anomalous and triggers an alert. This effectively inverts the traditional blacklist approach into a whitelist-based model.

3) Combined Detection and Benefits: By combining signature-based detection with baseline-driven whitelisting, the system is positioned to detect both known attacks and unexpected network behavior. Furthermore, the rule-based nature of Snort ensures high transparency, as each alert can be directly traced back to a specific rule, facilitating efficient analysis and interpretation by security analysts.

[3] https://www.snort.org/ GNU GPLv2

C. ML-based IDS

The ML-based component of the hybrid IDS serves as the anomaly detection branch, designed to identify previously unseen or zero-day attacks by learning the normal behavioral patterns of OT network traffic without requiring labeled attack data. In real-world industrial environments, labeled attack data is scarce or entirely unavailable, as real-world ICS networks are expected to carry predominantly normal traffic, and deliberate attack injection for data collection is impractical and potentially hazardous. This operational constraint motivates the selection of an unsupervised learning paradigm for the anomaly detection component. The ML-IDS model is trained exclusively on normal network flow data, learning a compact latent representation of benign traffic characteristics. The ML-IDS employs a Kolmogorov-Arnold Network Autoencoder, an autoencoder architecture built upon Kolmogorov-Arnold Network layers, as suggested in [16]. During inference, network flows that deviate significantly from this learned normal baseline, as measured by reconstruction error are flagged as anomalous. Unlike traditional Multi-Layer Perceptrons (MLPs), that apply fixed activation functions at each neuron and learn only the linear weight parameters, KAN architectures place learnable univariate activation functions on the network edges, thereby enabling more expressive and interpretable function approximation rooted in the Kolmogorov-Arnold representation theorem [26].

D. Alert Correlation

1) Problem Setting and Design Rationale: While the ML-based IDS detects deviations from learned normal flow behavior, its alerts are not observed in isolation; instead, they are fused with packet-level alerts from the rule-based Snort branch through a stateful alert-correlation layer that transforms heterogeneous raw outputs into a more contextualized, and operationally meaningful representation of security events. This design is motivated by a well-established limitation of standalone IDS deployments, as depicted in these studies [13], [27]–[29]: heterogeneous detectors generate alerts with different semantics, temporal granularity, and lack of contextual association, which makes direct comparison difficult and often leads to redundant, weakly contextualized alarms. To address this, the proposed correlation framework explicitly applies temporal similarity-based matching, and produces individually distinct unified outcomes, referring to the same communication instance to form corroborated events.

2) Methodology: In this work, a simple hybrid alert correlation framework is proposed that integrates the events from i) Snort alerts, representing packet-level signature matches from signature-based IDS, and, ii) Flow alerts, representing anomalous network flows classified by ML-IDS. The methodology prioritizes contextual association and enrichment, over complex probabilistic fusion approaches [30], [31]. To fuse these alerts with ML-based anomalous

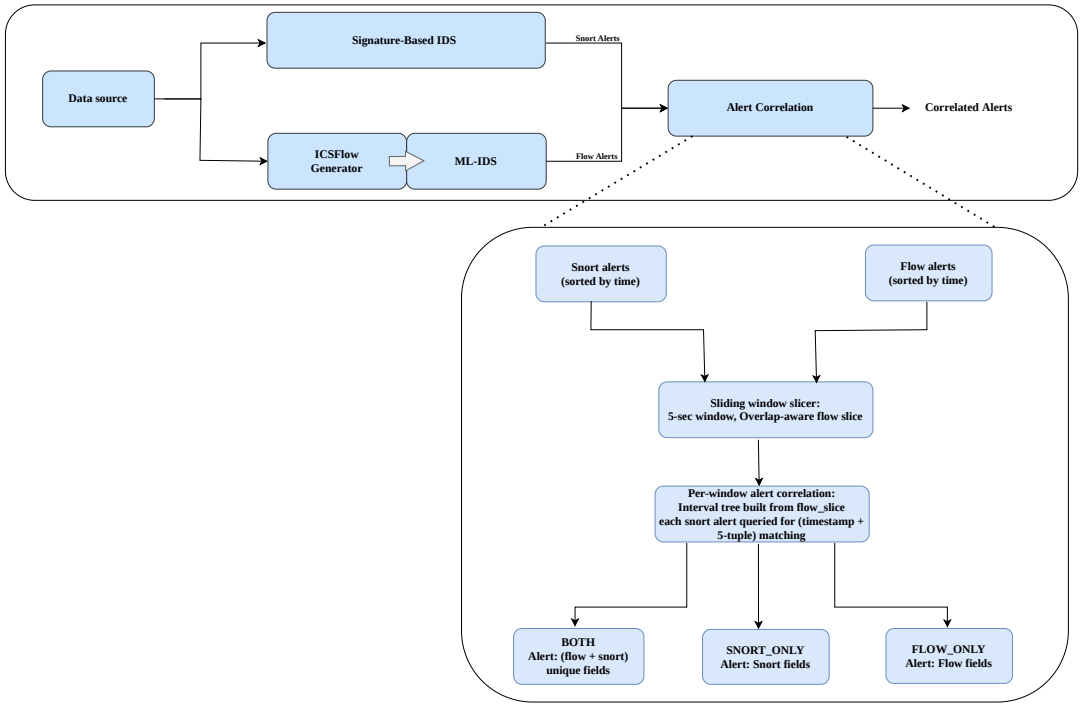


Fig. 1: Hybrid IDS Framework Schematic Overview

flow detections, a sliding-window correlation framework is utilized. This sliding-window approach is inspired by prior works on alert and event correlation for alert reduction and contextualization [29], [32], [33]. The correlation mechanism operates by partitioning the available alert timeline into fixed-duration tumbling windows and associating packet-level Snort alerts with temporally overlapping flow-level detections using normalized network attributes and tolerance-aware temporal matching. Each alert or flow is ultimately classified as jointly detected (`BOTH`), detected only by Snort (`SNORT_ONLY`), or detected only by the flow-based ML model (`FLOW_ONLY`). The goal of this categorization is to identify whether an event is flagged by both detection modalities, or whether it is observed exclusively by one source. It enables the construction of a unified alert stream while preserving the provenance of each detection. Although the present implementation is being used for offline analysis, the correlation logic is designed to be directly transferable to a real-time setting, where input data could arrive instead from streaming processors (e.g., Kafka[4]) and could be modeled and managed as windowed intervals. The various steps of the proposed alert correlation mechanism are outlined in the subsequent paragraphs.

a) Data Ingestion and Normalization: Prior to correlation, both alert datasets are first normalized into a common schema. For Snort alerts, timestamps are parsed into absolute datetimes and the relevant network fields, including source and destination IP addresses, source and destination ports, and protocol identifiers, are standardized into a consistent format. Flow alerts are transformed similarly by conforming source-specific fields, parsing flow start and end timestamps, filtering invalid records and standardizing the same communication attributes. The normalized Snort alerts are sorted by alert timestamp, whereas the flow alerts are sorted by flow start time.

b) Window-Based Processing and Flow Ownership: The correlation mechanism determines the entire temporal extent covering both alert sources at hand, which is then traversed using a sliding window of duration Δ (default: `5.0` seconds). For each window $[t, t+\Delta)$, three subsets are extracted: (i) a *Snort slice* containing all Snort alerts whose timestamps fall inside the window, (ii) a *flow match slice* containing all flows overlapping the tolerance-expanded interval $[t-\tau, t+\Delta+\tau]$, where τ is the configurable tolerance limit and (iii) a *flow owned slice* containing all flows whose start times lie within $[t, t+\Delta)$. The separation between flow match and flow ownership slices is essential: The *flow match slice* ensures that Snort alerts near window boundaries can still be matched with flows that begin slightly before or end slightly after the window due to a predefined allowed tolerance. In contrast, the *flow owned slice* defines a unique ownership rule for flows, guaranteeing that each flow is emitted as a flow-only alert in exactly one window, thereby preventing duplicate flow-only detections across adjacent windows.

c) Correlation via Interval Trees: This temporal similarity-based technique extends previous time-windowed

[4] https://kafka.apache.org/ Apache License 2.0

alert correlation frameworks such as those in [32], [34], [35] which select and correlate alerts based on temporal windows. This proposed technique employs an interval-tree[5] indexing to enable more efficient and scalable temporal matches within each tumbling window. Within each tumbling window, the set of candidate flow alerts is organized using an interval-tree-based index to support efficient temporal matching. Each flow is represented by its canonical 5-tuple, consisting of source address, destination address, source port, destination port, and protocol. This tuple is used as the flow identifier, while the corresponding time span is defined by a tolerance-expanded interval extending from the flow start time minus τ to the flow end time plus τ. Organizing the flows in this manner enables efficient retrieval of temporally relevant candidate matches for a given Snort alert. The value of τ could be adapted for the allowable timing tolerance depending upon the target network environment. For our offline analysis, it is assumed as `0.0` seconds.

For every Snort alert in the current window, the correlation mechanism derives both a forward and a reverse 5-tuple representation. The reverse representation is included to account for cases in which a Snort signature is triggered on return traffic rather than on the original packet direction. The alert timestamp is then queried against the interval tree under both tuple orientations. If no corresponding flow is identified, the alert is classified as `SNORT_ONLY`. Conversely, if one or more matching flows are found, the alert is classified as `BOTH`, and a separate correlation record is generated for each matched Snort-flow pair. The identifiers of all matched flows are retained to ensure that subsequent processing can distinguish already correlated flows from those that remain unmatched. e.g., `FLOW_ONLY`.

d) Flow-Only Emission and Duplicate Prevention: After processing the Snort alerts in a given window, the correlation mechanism evaluates the set of flows owned by that window, where ownership is determined by the flow start time. Any owned flow that has not been matched to a Snort alert is labeled `FLOW_ONLY`. Because each flow is assigned to a single owner window, a flow that extends across multiple consecutive windows can still be emitted only once. This ownership-based strategy eliminates cross-window duplication while preserving complete representation of unmatched flow detections.

e) Correlation Outcome: For each window, the proposed framework computes summary statistics that characterize the local correlation outcome, including the number of Snort alerts, the number of flows owned by the window, the number of emitted `BOTH` records, the number of uniquely matched flows, and the numbers of `SNORT_ONLY` and `FLOW_ONLY` detections. Importantly, the framework distinguishes between emitted `BOTH` records and uniquely matched flows. While the former represents the total number of Snort-flow correlation pairs, the latter counts each matched flow only once, thereby avoiding metric inflation when several Snort alerts map to the same flow. Under this formulation, the number of uniquely matched flows plus the number of `FLOW_ONLY` detections equals the number of flows owned by the window. The correlation outputs and window-level statistics are appended then to a log file incrementally after each window is processed. This streaming-oriented design bounds memory consumption and supports scalability to long-running traces and high-volume alert streams. The pseudocode for the proposed alert correlation has been depicted through the algorithms 1 and 2. Algorithm 1 generates the per-window Snort alert set S_w, the tolerance-expanded flow matching set F_m, and the owned-flow set F_o. These entities are then passed to Algorithm 2 for the actual alert correlation task.

[5] https://pypi.org/project/intervaltree/ Apache-2.0 license

Algorithm 1 Sliding-Window Slicing

Input: Snort alert set S, flow alert set F, window size Δ, tolerance τ
Output: Sequence of tuples $(w, S_w, F_m, F_o, t_s, t_e)$

$t_{\min} \leftarrow \min\left(\min_{s\in S} s.ts,\ \min_{f\in F} f.start\right)$
$t_{\max} \leftarrow \max\left(\max_{s\in S} s.ts,\ \max_{f\in F} f.end\right)$
$t_s \leftarrow t_{\min}$
$w \leftarrow 0$
while $t_s < t_{\max}$ **do**
  $t_e \leftarrow t_s + \Delta$
  $S_w \leftarrow \{s \in S \mid t_s \le s.ts < t_e\}$
  $F_m \leftarrow \{f \in F \mid f.start < t_e + \tau \ \wedge\ f.end > t_s - \tau\}$
  $F_o \leftarrow \{f \in F \mid t_s \le f.start < t_e\}$
  if $|S_w| > 0$ **or** $|F_m| > 0$ **then**
    yield $(w, S_w, F_m, F_o, t_s, t_e)$
  end if
  $t_s \leftarrow t_e$
  $w \leftarrow w + 1$
end while

IV. Evaluation Framework

A. Test Setup

A virtual Ubuntu 22.04.5 LTS server, equipped with 32 cores, 32 GB memory, and 500 GB disk, was set up as a testing environment for benchmarking our experiments. A virtual environment was created with Python 3.10.12 to have all the required packages installed for executing Python scripts.

B. Data Sources

The experimental evaluation was conducted using the ICS-Flow dataset, generated with the ICSSIM framework for intrusion detection in industrial control systems [9]. The testbed models a bottle-filling factory composed of two Programmable Logic Controllers (PLCs), three Human-Machine Interfaces (HMIs), and one attacker node, all connected through an emulated network using the Modbus TCP protocol. Additionally, four specific types of attack scenarios such as i) Reconnaissance (consists of IP-Scan & Port-Scan), ii) Replay, iii) Distributed Denial of Service (DDoS), and Man-in-the-Middle (MitM), were carried out, exploiting the vulnerabilites in ICS networks [9].

Algorithm 2 Sliding Window-Based Correlation of Snort and Flow Alerts

Input: Window identifier w, Snort alerts S_w, match-flow set F_m, owned-flow set F_o, window bounds (t_s, t_e), tolerance τ

Output: Correlated alert records R and window statistics Γ

$R \leftarrow \emptyset$
$M \leftarrow \emptyset$ ▷ indices of flows matched with Snort alerts
$I \leftarrow$ empty interval-tree index
for all $f \in F_m$ **do**
  $a \leftarrow (f.start) - \tau$
  $b \leftarrow (f.end) + \tau$
  $k \leftarrow (f.srcIP, f.dstIP, f.srcPort, f.dstPort, f.proto)$
  Insert interval $[a,b]$ with value $f.id$ into $I[k]$
end for
for all $s \in S_w$ **do**
  $x \leftarrow (s.ts)$
  $k_f \leftarrow (s.srcIP, s.dstIP, s.srcPort, s.dstPort, s.proto)$
  $k_r \leftarrow (s.dstIP, s.srcIP, s.dstPort, s.srcPort, s.proto)$
  $H \leftarrow \emptyset$, $V \leftarrow \emptyset$
  for all $iv \in I[k_f]$ such that $x \in iv$ **do**
    if $iv.value \notin V$ **then**
      $H \leftarrow H \cup \{iv.value\}$
      $V \leftarrow V \cup \{iv.value\}$
    end if
  end for
  for all $iv \in I[k_r]$ such that $x \in iv$ **do**
    if $iv.value \notin V$ **then**
      $H \leftarrow H \cup \{iv.value\}$
      $V \leftarrow V \cup \{iv.value\}$
    end if
  end for
  if $H = \emptyset$ **then**
    Append a `Snort-Only` record for s to R
  else
    for all $j \in H$ **do**
      $M \leftarrow M \cup \{j\}$
      $f \leftarrow F_m[j]$
      Append a `Both` record for pair (s, f) to R
    end for
  end if
end for
for all $f \in F_o$ **do**
  if $f.id \notin M$ **then**
    Append a `Flow-Only` record for f to R
  end if
end for
$B \leftarrow |\{r \in R : r.status = \texttt{Both}\}|$
$S \leftarrow |\{r \in R : r.status = \texttt{Snort-Only}\}|$
$F \leftarrow |\{r \in R : r.status = \texttt{Flow-Only}\}|$
$U \leftarrow |M|$
$\rho \leftarrow 100 \times B / \max(|S_w|, 1)$
$\Gamma \leftarrow \{w, t_s, t_e, |S_w|, |F_m|, |F_o|, B, S, F, U, \rho\}$
return (R, Γ)

For flow data generation, we used the publicly available ICSFlowGenerator tool[6], which supports offline conversion of packet capture (PCAP) traces into network flow records. In our study, we used raw network traffic from the file `traffic.pcap`, available in the ICSSIM dataset repository[7]. To explore the data briefly, the whole pcap file was processed in offline mode for flow-generation. The flow interval was set to `0.5` s (500 ms), following the configuration reported in the original study [9]. The distribution of benign and attack samples in the packet-level and flow-layer is summarized in Table I. It should be noted that the number of flow records generated in our study differs from that reported in the original ICS-Flow paper [9]. To ensure consistency and avoid discrepancies, the flow dataset generated locally using ICSFlowGenerator is used throughout model training and evaluation. Additionally, this study [9] also presents the information about the attack type and timings in a separate file, named as `attacker machine summary.csv`, which is used to evaluate the entire hybrid IDS and alert correlation setup in Section V.

TABLE I: ICS-Flow Data Distribution with their Benign & Attack content

Data	Total Count	Benign data	Attack data	Attack %age
`Packet`	25,161,400	1,043,645	24,117,755	95
`Flow`	8,609,062	49,423	8,559,639	99

At the packet level, the distribution of benign and malicious traffic within the pcap file was identified in Wireshark[8] using the filter `(ip.addr == 192.168.0.41) || (eth.addr == 02:42:c0:a8:00:29)`, which corresponds to the attacker node's IP and MAC addresses, respectively[7]. For the flow-level analysis, class labels were determined from the `NST_B_Label` field, following the methodology described in [9].

C. Model Training

a) Signature-based IDS with Whitelisting Approach: For the signature-based IDS, the rule generation process is based on a baseline-driven whitelisting approach combined with static detection rules. A subset of the first 100,000 packets of the recorded pcap trace is used as training data to establish a representation of benign network behavior. This subset was specifically selected to contain benign communication while excluding attack traffic.

During the training phase, the rule generator mechanism automatically analyzes the selected packets and derives a customized Snort 3 ruleset. Besides predefined static detection rules for known malicious activities, the generated rules describe expected network behavior such as valid hosts, communication relationships, traffic flow characteristics, overall flow volume, and maximum packet sizes. The rule generation process is implemented in Python and utilizes

[6] https://github.com/AlirezaDehlaghi/ICSFlow MIT license
[7] https://www.kaggle.com/datasets/alirezadehlaghi/icssim CC BY-NC-SA 4.0
[8] https://www.wireshark.org/ GNU GPLv2

TABLE II: Training Data Information

IDS	Benign
Snort	100,000
ML-IDS	23,276

TABLE III: Testing Data Information

IDS	Benign	Attack
Snort	553,047	24,117,755
ML-IDS	26,147	8,559,639

a high-performance packet parsing backend to efficiently process large pcap files. Table II shows the number of benign data points used for generating the rule set and for training KAN-AE model respectively.

b) KAN-AE Model: The KAN-AE model has been adapted from this source[9]. For the ML-IDS, the training set consists exclusively of normal flows, consistent with the unsupervised learning paradigm. Due to the insufficient number of flow instances generated from the initial 100,000 packets that were used for training the Snort-based component, an extended set of benign traffic was required for the ML-based model. Using the `attacker machine summary.csv` file as a reference for attack timing, network flows were generated from benign traffic captured up to 12:15:00 PM, ensuring that the training data did not contain any attack-related patterns. This training flow subset is used subsequently for model training. And the flows generated from the remaining PCAP data (from 12:15:00 PM onward) were used as the test set. During model development, the training subset is further divided into training and validation sets using an additional 80:20 split. Feature selection partly follows [9] approach: the label features are dropped; network identifiers (e.g., IP and MAC addresses) and temporal features are excluded in line with the original study. The protocol feature is encoded using Ordinal encoding[10], and all numerical features are normalized with MinMaxScaler[11].

For hyperparameter optimization, the open-source framework Optuna[12] is employed over 10 trials, with the best-performing model comprising hidden layers of sizes [48, 24, 12]. To determine the anomaly threshold for the ML model, we employ the Median Absolute Deviation (MAD), following an approach similar to the MAD implementation in PyThresh[13]. As a non-parametric method, MAD estimates the threshold directly from the model's decision scores without assuming any underlying distribution. After model fitting, the training data are evaluated to obtain decision scores for all samples. These scores are then normalized and provided to the MAD function, which computes the threshold as mentioned above. During testing/evaluation phase, samples with decision scores exceeding this MAD-based threshold are classified as anomalies (i.e., assigned label 1).

D. Model Testing

Before inference, the test set samples, as defined in Section IV-C0b, are processed using the same fitted encoder and scaler employed during training to ensure consistent feature transformation and normalization. The resulting decision scores are then compared with the predefined threshold with the trained model to produce the predicted class labels. This evaluation scheme ensures consistency between training and testing conditions and enables a reliable assessment of a model's classification performance. To maintain consistency with the corresponding flow-based test set, the remaining pcap data from 12:15:00 PM onward was used for evaluating the Snort-based IDS. Table III shows the information about benign and attack data size in the test set for both Snort and ML-IDS. The resulting Snort alerts and ML-IDS alerts are then used for further correlation and evaluation within the hybrid IDS framework.

E. Evaluation Metrics

To assess ML model performance, Confusion Matrix (CM), a standard metric for binary classification, is being used. Based on the ground-truth labels, the CM comprises four outcomes: True Negatives (TN), False Negatives (FN), False Positives (FP), and True Positives (TP), which together characterize correctly detected events as well as missed and falsely raised IDS alerts. In addition, the classification performance is also evaluated via extensively used F1-score[14]. The F1-score ranges from 0 to 1, with higher values indicating better classification performance.

V. Evaluation Results

The evaluation is designed to fulfill two closely related research objectives. First, the evaluation asks whether a hybrid unsupervised detection framework that combines flow-based analysis and signature-based IDS alerts provides better predictive value than either source in isolation. Second, the evaluation discusses whether alert correlation make the combined output functionally useful by reducing duplication and exposing where the two detection approaches complement each other.

Table IV shows the number of alerts produced by each IDS module. Starting with the ML-IDS, it achieves an F1-score of 0.98, indicating strong detection performance. As evident from Table I, the attack class dominates the data in general. Hence, considering this class imbalance, Table V presents the confusion matrix elements for the ML-IDS flow-based predictions to provide a closer look at the ML-IDS predictions. It could be observed from Table V that the model does not trivially classify all data samples as anomalous which could lead to F1-score inflation. In contrast, the F1-score is not directly computed for the Snort IDS, since Snort reports only detected anomalous events (i.e., class 1 instances) rather than assigning labels to all analyzed samples by ML-IDS. However, the correlated outputs might provide a useful basis for evaluating the detection behavior of the overall hybrid framework and is discussed shortly after. Nevertheless, after the alert correlation procedure is performed, 12,618,982 records (74.9%) are jointly identified by both the flow-based

[9] https://github.com/SekiroRong/KAN-AutoEncoder MIT License

[10] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html BSD License

[11] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html BSD License

[12] https://optuna.org/ MIT License

[13] https://github.com/KulikDM/pythresh BSD 2-Clause License

[14] https://scikit-learn.org/0.15/modules/generated/sklearn.metrics.f1_score.html BSD License

TABLE IV: Hybrid IDS Alerts

IDS	Alert count
Snort	16,739,316
ML-IDS	8,360,867
Total	25,100,183

TABLE V: ML-IDS Confusion Matrix

TN:	25,595	FP:	552
FN:	199,324	TP:	8,360,315

TABLE VI: Alert Correlation Summary

Category	Alert Count	Share%
BOTH	12,618,982	74.9
SNORT_ONLY	4,120,334	24.5
FLOW_ONLY	103,163	0.6
Total	16,842,479	100

model and Snort (BOTH), while 4,120,334 records (24.5%) are reported exclusively by Snort (SNORT_ONLY) and 103,163 records (0.6%) are detected only by the flow-based model (FLOW_ONLY), as summarized in Table VI. These results indicate a substantial overlap between the two detection components. Furthermore, when we compare Tables IV and VI, it already shows a significant drop (approx. 33% alarm reduction) in the total alert count achieved by this alert correlation itself.

Here onwards, the correlated outputs are utilized for evaluating the detection behavior of the overall hybrid framework. The aforementioned `attacker machine summary.csv` is being used to provide the ground-truth attack schedule and its type to associate with the window-level statistics and correlated alerts produced by the alert correlation module in order to evaluate the entire framework. The attacker summary includes the attack label, attack start time, and attack end time, which allows correlated alerts to be mapped back to known attack episodes during offline evaluation. And the window-level file provides the temporal structure needed to align detector output with the attack timeline. This labeled setting is useful because it enables a controlled analysis of how correlated alerts behave during known attack intervals such as IP-Scan, Port-Scan, MitM, Replay, and DDoS. The attacker summary is just being used here as an evaluation tool, not as an assumption that such labels would be available in realistic scenarios.

The evaluation begins by establishing the temporal structure and context of the hybrid framework. Figure 2 represents the correlation windows in this framework aligned with the attack timeline. It indicates not only the presence of specific attack classes but also their temporal distribution with respect to the framework's 5-second correlation windows. In Figure 2, the upper ribbon provides a compact view of attack coverage across the entire correlation timeline, while the lower panel separates the ground-truth attack episodes by class, revealing the temporal distribution of DDoS, IP-Scan, MitM, Port-Scan, and Replay events over the same timeline. This acts as the temporal backbone of the evaluation, making the assessment of complementarity both interpretable and reproducible. The figure also makes clear that the attack schedule is not uniform: some behaviors, such as IP-Scan, occur more frequently, whereas others, such as DDoS and Replay, appear in fewer but more distinct sessions. However, it is important to note here that the attacks were apparently performed in mutually non-overlapping temporal sessions.

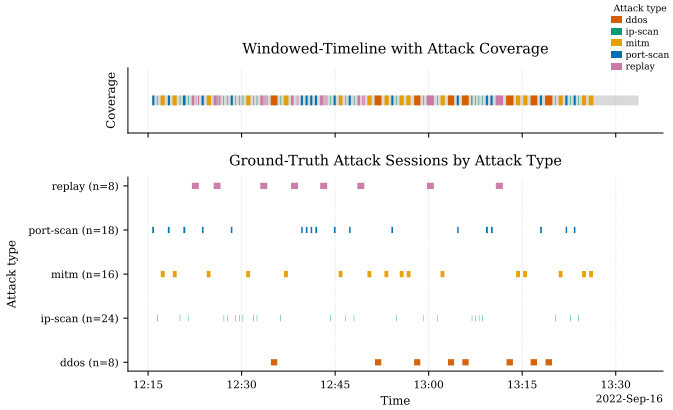


Fig. 2: Correlation Windows aligned with Attack Coverage

The central technical premise of the hybrid framework is that flow-based ML-IDS and signature-based IDS alerts do not merely replicate the same information; rather, they provide distinct and partially complementary detection perspectives. To evaluate how the hybrid framework behaves under known attack activity, the correlated alerts are first aligned with the ground-truth attack schedule. The attack schedule is provided by `attacker machine summary.csv`, which records the start and end time of each attack session together with its attack label, while window-level file provides the broader 5-second temporal structure used throughout the experiment. In this approach, each correlated alert is assigned an event timestamp using either by flow start time or Snort timestamp whichever available. This choice ensures that the labeling step remains robust even when one timing field is missing. Each alert is then matched against the attack intervals, so that alerts falling inside a known attack episode inherit the corresponding attack label, while alerts outside all labeled intervals are treated as normal activity. This step transforms the correlated alert stream into an attack-aware evaluation set, which is essential for assessing hybrid-system behavior across different classes of adversarial activity.

Leveraging the attack schedule, the correlated alerts presented in Table VI are aligned with the temporal boundaries of the attack sessions. Alerts whose timestamps fall within any attack session are classified as true positives (TP), whereas those occurring outside these intervals are considered false positives (FP). A comparative analysis between Table V and Table VII reveals a meaningful reduction in false alarms, decreasing from 552 to 85, which corresponds to approx. 84% FP reduction from ML-IDS alarms alone.

TABLE VII: True Positives and False Positives after Alert Correlation

Category	Alert Count
TP	16,842,394
FP	85
Total	16,842,479

Next, we evaluate how these two monitoring modalities contribute to the detection of each attack class. As stated before, each correlated alert is categorized according to its correlation outcome: `FLOW_ONLY`, `SNORT_ONLY`, or `BOTH`. This categorization is fundamental to the evaluation because it makes it possible to ask whether the hybrid framework is genuinely combining complementary observations.

To make the results interpretable, these outcomes are aggregated by attack type, producing a heatmap visualization. This visualization provides a compact visual representation, making it easier to see at a glance which attack classes are predominantly flow-visible, which are signature-visible, and which benefit most from cross-IDS confirmation.

Figure 3 provides clear empirical evidence for this interpretation. The heatmap shows that several attack classes are predominantly flow-visible, indicating that the flow-analysis component contributes substantial unique coverage. In particular, IP-Scan appears entirely `FLOW_ONLY` (100.0%), while DDoS is also almost entirely `FLOW_ONLY` (99.53%) with a minor shared or Snort-only contribution. MitM attack follows a similar pattern, with 94.95% of correlated alerts categorized as `FLOW_ONLY`, only 4.78% jointly observed by both subsystems, and 0.28% visible only to Snort. These results suggest that for several events, especially scan-like traffic behaviour, the flow-level anomaly detection functions as the dominant observation layer and reveals their occurrences that a signature-based IDS alone would either miss or detect only sparsely.

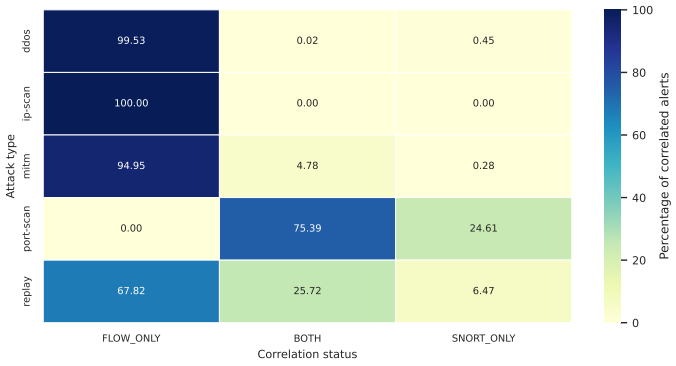


Fig. 3: Complementarity between ML-IDS and signature-based IDS

At the same time, the heatmap also shows that the hybrid framework is not merely a flow-dominated system with a marginal signature-IDS impact. Port-Scan exhibits the strongest cross-IDS confirmation, with 75.39% of correlated alerts classified as `BOTH` and the remaining 24.61% as `SNORT_ONLY`, while `FLOW_ONLY` contribution is effectively absent in this class which is similarly reflected from False Negatives in Table V. Replay attack scenario occupies a more intermediate position, with 67.82% `FLOW_ONLY`, 25.72% `BOTH`, and 6.47% `SNORT_ONLY`. This variation across attack classes is particularly important because it demonstrates that the relative contribution of each IDS depends on the nature of the behavior being observed. The hybrid framework's effectiveness is not due to any single detector performing optimally in all cases, but rather because different behaviors are more distinctly captured in flow-based features and packet-based (signature-based) alerts. A large `FLOW_ONLY` proportion indicates that flow-level analysis extends the observable attack surface beyond what signature rules alone provide. A non-trivial `SNORT_ONLY` proportion shows that signature-based IDS still contributes detections with no corresponding flow-only match. Most importantly, a substantial `BOTH` proportion provides cross-IDS confirmation, which increases the confidence in the detections. These three categories therefore could provide a far more informative view of system behavior than a mere aggregate accuracy score. They reveal not only whether the framework detects suspicious activity, but also how the two subsystems interact to produce that detection. These results show that the value of the hybrid system lies not in simple redundancy, but in its ability to combine broader flow-side coverage with viable signature-based confirmation.

VI. Conclusion & Future Work

This study presented a hybrid intrusion detection framework that integrates packet-level signature-based monitoring with a flow-level machine learning IDS, augmented by a temporal similarity-based alert correlation mechanism. The proposed approach addresses open key challenges identified in prior work, including the application of unsupervised binary classification and the reduction of false alarms [9]. Experimental evaluation using the ICS-Flow dataset demonstrates that the hybrid framework achieves strong overall detection performance while reducing alert volume by 33% through effective aggregation and deduplication. Additionally, the framework significantly minimizes false positives by approx. 84% in this case, thereby enhancing the reliability of generated alerts. The analysis of correlated alerts further confirms that all attack scenarios in the dataset are successfully identified, underscoring the complementary detection capabilities of packet-level and flow-level monitoring. Overall, the results highlight the potential of hybrid and correlation-driven approaches for improving the effectiveness and practicality of IDS in ICS network environments.

In this study, the evaluation is confined to the ICS-Flow dataset and KAN-AE model in an offline, hybrid setting and the results seem quite promising, thereby encouraging further investigation and comprehensive evaluation. Moving forward, we plan to extend this work to a streaming environment to better reflect real-world network conditions for real-time detection. In addition, future efforts will focus on conducting comprehensive benchmarking to examine the detection performance across multiple datasets and diverse machine learning models, including evaluations on enterprise-scale network data to further validate the versatility of the proposed framework. Moreover, given the inherent interpretability attributed to KAN-based models, we aim to explore and leverage this characteristic to gain deeper insights into the

model's decision-making process. Such analysis is expected to improve transparency and trust in the system, as well as support more informed alert triage and response strategies.

Acknowledgment

This research was conducted in the context of the CAIDAN project ("Cyberattack Attribution Using AI-Enhanced Intrusion Detection with Alert Correlation in Industrial Networks"), funded by the German Federal Ministry of Research, Technology and Space under grant number 16KIS1946.

References

[1] E. Byres and J. Lowe, "The myths and facts behind cyber security risks for industrial control systems," in *Proceedings of the VDE Kongress*, vol. 116, 2004, pp. 213–218.

[2] A. Termanini, D. Al-Abri, H. Bourdoucen, and A. Al Maashri, "Using machine learning to detect network intrusions in industrial control systems: a survey," *International Journal of Information Security*, vol. 24, no. 1, p. 20, 2025.

[3] R. Grubbs, J. Stoddard, S. Freeman, and R. Fisher, "Evolution and trends of industrial control system cyber incidents since 2017," *Journal of Critical Infrastructure Policy*, vol. 2, no. 2, pp. 45–79, 2021.

[4] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," National Institute of Standards and Technology, Tech. Rep. SP 800-82 Rev. 2, 2015.

[5] A. Shaikh and P. Gupta, "Advanced signature-based intrusion detection system," in *Intelligent Communication Technologies and Virtual Mobile Networks: Proceedings of ICICV 2022*. Springer, 2022, pp. 305–321.

[6] E. M. Maseno, Z. Wang, and H. Xing, "A systematic review on hybrid intrusion detection system," *Security and Communication Networks*, vol. 2022, no. 1, p. 9663052, 2022.

[7] Q. Gulzar and K. Mustafa, "Interdisciplinary framework for cyber-attacks and anomaly detection in industrial control systems using deep learning," *Scientific Reports*, vol. 15, no. 1, p. 26575, 2025.

[8] Q. Zhu, Y. Ding, J. Jiang, and S.-H. Yang, "Anomaly detection using invariant rules in industrial control systems," *Control Engineering Practice*, vol. 154, p. 106164, 2025.

[9] A. Dehlaghi-Ghadim, M. H. Moghadam, A. Balador, and H. Hansson, "Anomaly detection dataset for industrial control systems," *IEEE Access*, vol. 11, pp. 107 982–107 996, 2023.

[10] M. Z. Khan, A. A. Reshi, S. Shafi, and I. Aljubayri, "An adaptive hybrid framework for IIoT intrusion detection using neural networks and feature optimization using genetic algorithms," *Discover Sustainability*, vol. 6, no. 1, p. 382, 2025.

[11] S. Srivastav, A. K. Shukla, S. Kumar, and P. K. Muhuri, "HYRIDE: HYbrid and robust intrusion DEtection approach for enhancing cybersecurity in Industry 4.0," *Internet of Things*, vol. 30, p. 101492, 2025.

[12] O. Koucham, S. Mocanu, G. Hiet, J.-M. Thiriet, and F. Majorczyk, "Cross-domain alert correlation methodology for industrial control systems," *Computers & Security*, vol. 118, p. 102723, 2022.

[13] M. Heigl, E. Weigelt, A. Urmann, D. Fiala, and M. Schramm, "Exploiting the outcome of outlier detection for novel attack pattern recognition on streaming data," *Electronics*, vol. 10, no. 17, p. 2160, 2021.

[14] A. Paul, F. Schuster, and H. König, "Whitelisting for characterizing and monitoring process control communication," in *International Conference on Network and System Security*. Springer, 2023, pp. 23–45.

[15] E. Luna-Villagómez and V. Mahalec, "Exploring Kolmogorov-Arnold networks for unsupervised anomaly detection in industrial processes," *Processes*, vol. 13, no. 11, p. 3672, 2025.

[16] M. Ghorbani, A. Ghassemi, M. Alikhani, H. Khaloozadeh, and A. Nikoofard, "Using Kolmogorov-Arnold network for cyber–physical system security: A fast and efficient approach," *International Journal of Critical Infrastructure Protection*, vol. 50, p. 100768, 2025.

[17] Y. Wu, Z. Zang, X. Zou, W. Luo, N. Bai, Y. Xiang, W. Li, and W. Dong, "Graph attention and Kolmogorov-Arnold network based smart grids intrusion detection," *Scientific Reports*, vol. 15, no. 1, p. 8648, 2025.

[18] Y. Abudurexiti, G. Han, F. Zhang, and L. Liu, "An explainable unsupervised anomaly detection framework for industrial internet of things," *Computers & Security*, vol. 148, p. 104130, 2025.

[19] L. Rosa, T. Cruz, M. B. de Freitas, P. Quitério, J. Henriques, F. Caldeira, E. Monteiro, and P. Simões, "Intrusion and anomaly detection for the next-generation of industrial automation and control systems," *Future Generation Computer Systems*, vol. 119, pp. 50–67, 2021.

[20] H. Asad, S. Adhikari, and I. Gashi, "A perspective-retrospective analysis of diversity in signature-based open-source network intrusion detection systems," *International Journal of Information Security*, vol. 23, no. 2, pp. 1331–1346, 2023.

[21] W. Ibrar, D. Mahmood, A. S. Al-Shamayleh, G. Ahmed, S. Z. Alharthi, and A. Akhunzada, "Generative ai: a double-edged sword in the cyber threat landscape," *Artificial Intelligence Review*, vol. 58, no. 9, p. 285, 2025.

[22] J. Ali, S. Ali, T. Al Balushi, and Z. Nadir, "Intrusion detection in industrial control systems using transfer learning guided by reinforcement learning," *Information*, vol. 16, no. 10, p. 910, 2025.

[23] M. Hasan, "A hybrid real-time intrusion detection system for an internet of things environment with signature and anomaly based intrusion detection," Ph.D. dissertation, National College of Ireland, Dublin, Ireland, 2020.

[24] M. A. D. Varma, G. S. Vaasist, B. C. Reddy, M. P. Reddy, and R. Nair, "Intrusion detection system using signature and anomaly based algorithm," in *2025 International Conference on Inventive Computation Technologies (ICICT)*. IEEE, 2025, pp. 838–842.

[25] S. A. Mirheidari, S. Arshad, and R. Jalili, "Alert correlation algorithms: A survey and taxonomy," in *Cyberspace Safety and Security (CSS 2013)*, ser. Lecture Notes in Computer Science, vol. 8300. Springer, 2013, pp. 183–197.

[26] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold networks," in *International conference on learning representations*, vol. 2025, 2025, pp. 70 367–70 413.

[27] R. Sadoddin and A. Ghorbani, "Alert correlation survey: framework and techniques," in *Proceedings of the 2006 international conference on privacy, security and trust: bridge the gap between PST technologies and business services*, 2006, pp. 1–10.

[28] K. Zhang, F. Zhao, S. Luo, Y. Xin, and H. Zhu, "An intrusion action-based ids alert correlation analysis and prediction framework," *IEEE Access*, vol. 7, pp. 150 540–150 551, 2019.

[29] H. Albasheer, M. Md Siraj, A. Mubarakali, O. Elsier Tayfour, S. Salih, M. Hamdan, S. Khan, A. Zainal, and S. Kamarudeen, "Cyber-attack prediction based on network intrusion detection systems for alert correlation techniques: a survey," *Sensors*, vol. 22, no. 4, p. 1494, 2022.

[30] F. Jemili, "Towards data fusion-based big data analytics for intrusion detection," *Journal of Information and Telecommunication*, vol. 7, no. 4, pp. 409–436, 2023.

[31] K. Wu, H. Qu, and C. Huang, "A network intrusion detection method incorporating bayesian attack graph and incremental learning part," *Future Internet*, vol. 15, no. 4, p. 128, 2023.

[32] H. Maosa, K. Ouazzane, and M. C. Ghanem, "A hierarchical security event correlation model for real-time threat detection and response," *Network*, vol. 4, no. 1, pp. 68–90, 2024.

[33] D. Levshun and I. Kotenko, "A survey on artificial intelligence techniques for security event correlation: models, challenges, and opportunities," *Artificial Intelligence Review*, vol. 56, no. 8, pp. 8547–8590, 2023.

[34] J. Ma, Z.-t. Li, and W.-m. Li, "Real-time alert stream clustering and correlation for discovering attack strategies," in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 4. IEEE, 2008, pp. 379–384.

[35] M. Wu and Y. Moon, "Alert correlation for cyber-manufacturing intrusion detection," *Procedia Manufacturing*, vol. 34, pp. 820–831, 2019.