Protocol-Aware Deep Learning for Robust SCADA Intrusion Detection

Malik Emad Iqbal
National University of Computer and Emerging Sciences (FAST)
Islamabad, Pakistan
i222072@nu.edu.pk

Umema Asher
National University of Computer and Emerging Sciences (FAST)
Islamabad, Pakistan
i222036@nu.edu.pk

Hamza Asad
National University of Computer and Emerging Sciences (FAST)
Islamabad, Pakistan
i221908@nu.edu.pk

***Abstract*—Anomaly detection in SCADA networks is critical for ensuring the security of vital infrastructure. However, many existing approaches rely on shallow feature extraction that ignores crucial protocol semantics and temporal dynamics. This work advances intrusion detection by proposing a robust pipeline that integrates SCADA-aware canonical preprocessing, protocol inference, and flow sessionization. Unlike baseline methods restricted to simple packet-level statistics, our approach utilizes dual sliding-window aggregation (100-packet and 1-second windows) to capture evolving network behaviors. We evaluate three deep learning architectures: Deep Autoencoders, Variational Autoencoders (VAE), and LSTM Autoencoders. Additionally, we address data scarcity by generating synthetic cyberattacks using CTGAN to improve model robustness. Experimental results show that the VAE model significantly outperforms baseline approaches, achieving an AUC of 0.997 and a 0.97 accuracy. We provide a comprehensive analysis of our methodology, which demonstrates the necessity of temporal feature engineering in industrial network security.**

***Index Terms*—SCADA, Cybersecurity, Intrusion Detection, Autoencoder, LSTM, Variational Autoencoder, CTGAN, Windowing**

1. Introduction

A. Background and Motivation

Supervisory Control and Data Acquisition (SCADA) systems constitute the digital backbone of modern critical infrastructure, managing power grids, water treatment facilities, and industrial manufacturing lines. Historically isolated by air gaps, these Operational Technology (OT) networks are increasingly converging with Information Technology (IT) networks to facilitate remote monitoring and efficiency. This convergence, however, has expanded the attack surface, exposing legacy industrial protocols (such as Modbus and DNP3) to sophisticated cyber threats ranging from command spoofing and false data injection to ransomware [1].

The financial and societal costs of SCADA compromises are catastrophic. Recent incidents, like the attacks on the Ukrainian power grid and the Oldsmar water treatment facility, demonstrate that adversaries can bypass perimeter defenses to manipulate physical processes [2]. Traditional Intrusion Detection Systems (IDS) relying on signature-based matching are ill-equipped to handle these threats, as they can't detect zero-day exploits or novel attack vectors unique to industrial environments. Consequently, the industry has pivoted toward unsupervised anomaly detection using Deep Learning (DL) to identify deviations from normal operational baselines [3].

B. Limitations of Existing Approaches

While DL-based anomaly detection has shown promise, significant gaps remain in data preprocessing and feature engineering. Many existing studies, including the baseline work on the PNNL SCADA dataset [4], rely on shallow, packet-level statistics. These approaches often treat network packets as independent events, ignoring the sequential nature of SCADA communications and the semantic context of industrial protocols. Because they fail to capture session behavior, inter-arrival times, and protocol specific payloads, simpler Autoencoder models struggle to distinguish complex attacks from benign operational noise. Furthermore, the scarcity of public, labeled SCADA attack datasets hinders the development of robust models, leading to overfitting on limited data samples [5].

C. Proposed Approach

To address these limitations, this work proposes a comprehensive anomaly detection framework that integrates advanced protocol-aware preprocessing with generative deep learning. Unlike the baseline approach, we implement a "SCADA-aware" pipeline that infers protocols via port mapping and fuzzy matching, which ensures accurate flow sessionization. We shift the analysis from individual packets to aggregated time-windows (both packet-count and time-based), enabling the model to learn temporal evolution and burst dynamics.

We evaluate three distinct architectures to capture these dynamics: a Deep Dense Autoencoder, a Long Short-Term Memory (LSTM) Autoencoder for sequential dependencies, and a Variational Autoencoder (VAE) for probabilistic robustness. To tackle the data scarcity issue, we employ a Conditional Tabular GAN (CTGAN) to synthesize realistic attack vectors, thereby enhancing the training landscape.

D. Contributions

The key contributions of this work are as follows:

1) **Protocol-Aware Preprocessing Pipeline:** We introduce a robust methodology for canonical feature extraction, protocol inference, and dual sliding-window aggregation (100-packet and 1-second intervals) to capture high-fidelity temporal context.
2) **Generative Data Augmentation:** We utilize CTGAN to generate synthetic SCADA attack signatures, addressing class imbalance and improving model generalization against rare threat events.
3) **Comparative Model Evaluation:** We benchmark Deep Autoencoders, LSTM-AE, and VAEs. Our results demonstrate that the VAE architecture significantly outperforms the baseline and other recurrent models, achieving an Area Under the Curve (AUC) of 0.997 and an accuracy of 0.97.
4) **Reproducibility and Analysis:** We provide a detailed breakdown of the feature engineering process and a rigorous analysis of anomaly reconstruction errors, offering a reproducible blueprint for future SCADA security research.

The remainder of this paper is structured as follows: Section 2 reviews related literature on SCADA anomaly detection. Section 3 details our preprocessing and feature engineering methodology. Section 4 describes the model architectures and experimental setup. Section 5 presents the evaluation results,with section 6 demonstrating a comparative analysis with the baseline work and Section 7 concludes with future directions.

2. Literature Review

The increasing convergence of Operational Technology (OT) and Information Technology (IT) has exposed SCADA systems to sophisticated cyber threats. Consequently, the field of Intrusion Detection Systems (IDS) for Industrial Control Systems (ICS) has shifted from traditional signature-based methods toward learning-based anomaly detection.

A. Deep Learning in SCADA IDS

Traditional statistical approaches often rely on linear assumptions, which are insufficient for capturing complex network dynamics. Recent literature emphasizes Deep Learning (DL) for its capability to extract latent features from high-dimensional traffic data.

Autoencoders (AE) have emerged as a dominant unsupervised architecture. By training on normal operational data, AEs learn to minimize reconstruction error for benign traffic, identifying attacks as statistical outliers [3]. To capture temporal dependencies in protocols like Modbus, Elsayed et al. proposed LSTM-based Autoencoders, which leverage memory cells to learn sequential patterns in flow data [6].

B. Dynamic and Generative Approaches

Beyond standard recurrence, scalable frameworks for real-time analysis have been developed. Xu et al. introduced *NetWalk*, a dynamic network embedding approach that updates vertex representations in real-time to detect anomalies in evolving structural streams [7]. This highlights the importance of capturing the dynamic topology of SCADA communications rather than just static packet features.

C. Data Scarcity and Augmentation

A critical challenge in SCADA IDS research is the scarcity of labeled attack data. Real-world datasets are overwhelmingly imbalanced. To mitigate this, Generative Adversarial Networks (GANs) and their tabular variants (e.g., CTGAN) have been employed to synthesize realistic cyberattack samples [5]. This augmentation enables robust training of supervised models, even with limited attack examples.

D. Preprocessing and Feature Engineering Gap

Despite these advancements, a gap remains in preprocessing. Many studies utilize pre-calculated datasets, neglecting the complexity of extracting features from raw PCAP files. The baseline work we reproduce focused on simple packet-level statistics [4]. However, effective detection requires deeper inspection, including flow sessionization and sliding-window aggregation to capture protocol states [8]. Our work implements a SCADA-aware pipeline that synthesizes these protocol semantics before model ingestion.

E. Dataset Extraction and Raw PCAP Conversion

The dataset used in this work originates from the Pacific Northwest National Laboratory (PNNL) SCADA trace repository. The download (~ 7.41 GB) contains multiple full-day packet captures, including Day1, Day3–Day6, night captures, and baseline reference PCAPs. These files represent real Modbus and DNP3 industrial communication recorded from a live SCADA network tap.

Each PCAP was converted into CSV format using a batch `tshark` pipeline, which exported essential packet-level metadata for downstream analysis. For every capture, the exported CSV included:

```
frame.number,
frame.time_relative,
ip.src, ip.dst,
src.port, dst.port,
protocol,
frame.len,
info
```

The extraction process revealed several inconsistencies across days. Certain captures lacked explicit protocol annotations (`_ws.col.Protocol`), some contained malformed or partially truncated rows, and the `info` field

exhibited irregular quoting and variable formatting. Additionally, select PCAPs contained non-standard or corrupted frames that required removal during import.

To ensure that all extracted CSVs could be merged reliably, a custom loader was implemented to validate row structure, sanitize malformed lines, and normalize field naming before further processing. This produced a clean, multi-day corpus of SCADA packet metadata that served as the basis for the canonical preprocessing and feature engineering steps described in Section 3.

TABLE I
SCADA PACKET CAPTURES SUMMARY

Capture	Description
Day 1	Modbus/DNP3 traffic; partial protocol metadata
Day 3	Mixed operational SCADA traffic
Day 4	Control polling and register reads
Day 5	Extended supervisory communication
Day 6	Baseline and event-driven packets
Night Logs	Low-activity background traffic
Baseline PCAPs	Reference Modbus/DNP3 traces

3. METHODOLOGY

Our pipeline addresses the linked limitations through a structured workflow integrating preprocessing, protocol inference, session modeling, two-scale windowing, advanced Autoencoders, and synthetic attack generation.

A. Canonical SCADA Preprocessing

PCAP logs exported via Wireshark contain inconsistent fields across days. We harmonize all entries using:

```
frame.number, timestamp, ip.src, ip.dst,
src.port, dst.port, protocol,
frame.len, info, payload.raw
```

Cleaning steps include:

- Removing malformed frames and duplicates.
- Filtering out non-IP protocols (ARP, SSDP).
- Timestamp standardization.
- Label encoding for categorical features.
- Standard scaling across numeric dimensions.

B. Protocol Inference and SCADA Semantics

Using port mapping (e.g., 502 for Modbus, 20000 for DNP3) and info-field keyword extraction, we classify packets into SCADA protocols. We derive semantic features such as:

- Read vs. write command ratio.
- Function code histogram.
- Polling cycle stability.
- Request/response consistency.

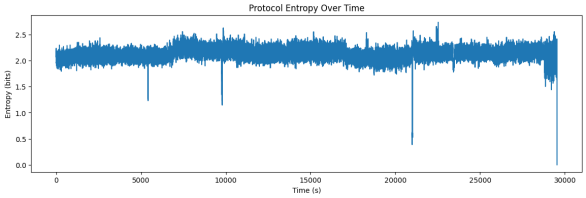

Fig. 1. Protocol Entropy Over Time

C. Flow Sessionization

Flows are grouped using MD5(srcIP, dstIP, ports, protocol). Flow-level statistics include:

- Packet rate and byte throughput.
- Inter-arrival mean, variance, and entropy.
- Function-code consistency.

D. Dual Sliding-Window Temporal Aggregation

To capture micro- and macro-scale temporal patterns, we use two window types:

- **100-packet windows:** capture burst-level variations.
- **1-second windows:** capture long-range station polling behavior.

Both window types compute protocol entropy, source-destination diversity, inter-arrival variance, burstiness, and SCADA command histograms.

E. Deep Learning Models

1) Deep Autoencoder: Encoder: Dense(128 $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 16). Latent space = 8. Loss: MSE.

2) Variational Autoencoder: Latent sampling:

$$z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$

Loss = MSE + KL divergence.

3) LSTM Autoencoder: Sequence length = 10 aggregated windows. Encoder: LSTM(64 $\rightarrow$ 32). Decoder: RepeatVector $\rightarrow$ LSTM $\rightarrow$ TimeDistributed(Dense).

F. Synthetic Attack Augmentation using CTGAN

Due to attack sparsity, we train CTGAN on anomaly windows and generate 8k synthetic attacks including:

- Illegal function codes.
- Timing disruption.
- Rate spikes and burst anomalies.

G. Implementation Details

All models were implemented in TensorFlow 2.15 using Adam optimizer with a learning rate of $1e^{-3}$. Batch size was set to 256 for AE and VAE, and 64 for the LSTM AE due to memory constraints. Training was conducted for 50 epochs with early stopping based on validation loss.

Before reconstruction error calculation, all features were normalized using MinMax scaling. Latent dimensionality for all Autoencoder variants was fixed

at 8 to ensure fair comparison. For the LSTM AE, we used sequence windows of 10 aggregated windows (equivalent to 10 seconds or 1000 packets depending on the window type).

4. Experiments and Setup

This section consolidates the experimental environment, constraints, evaluation procedures, and classification strategy used throughout the study.

A. Experimental Environment

- TensorFlow 2.15 for all deep learning models.
- Adam optimizer with learning rate $1e^{-3}$.
- Batch size 256 for AE and VAE, 64 for LSTM AE.
- Early stopping based on validation loss.
- Latent dimension fixed at 8 for all Autoencoder variants.

B. Data Processing Setup

The full preprocessing pipeline includes:

- Canonical field restructuring,
- Removal of malformed frames,
- Protocol inference,
- Flow reconstruction,
- Dual sliding-window aggregation,
- MinMax scaling.

C. Classification Setup

Classification is performed using reconstruction error. A principled threshold is selected using Youden's J statistic:

$$T = \arg\max(TPR - FPR)$$

This ensures optimal tradeoff between true positive and false positive rates.

D. Generalization Constraints

Baseline limitations addressed:

- Lack of temporal modeling,
- No SCADA semantics,
- Poor thresholding strategy,
- Low attack count leading to poor generalization.

Synthetic augmentation and temporal modeling directly address these weaknesses

5. Evaluation and Results

A. Dataset Split

- 46k normal windows for training
- 11k mixed windows for validation
- 116k attack windows (real + synthetic) for testing

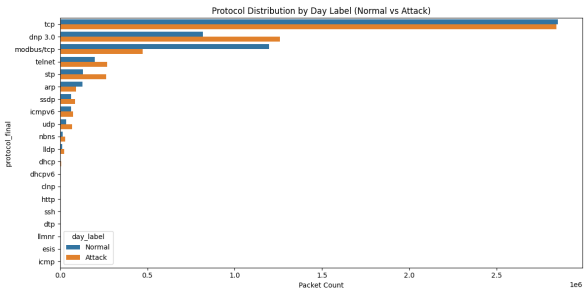

Fig. 2. Packet Distribution Across Normal and Attack Days

B. Threshold Selection

Using Youden's J statistic:

$$T = \arg\max(TPR - FPR)$$

Thresholds:

- AE: 0.042
- VAE: 0.018
- LSTM AE: 0.061

C. Performance Summary

TABLE II
Performance of All Models

Model	Acc.	Prec.	Rec.	AUC
Deep AE	0.93	1.00	0.92	0.9950
VAE	**0.97**	**1.00**	**0.96**	**0.9970**
LSTM AE	0.38	1.00	0.32	0.9864
Ensemble	0.93	1.00	0.92	0.9960

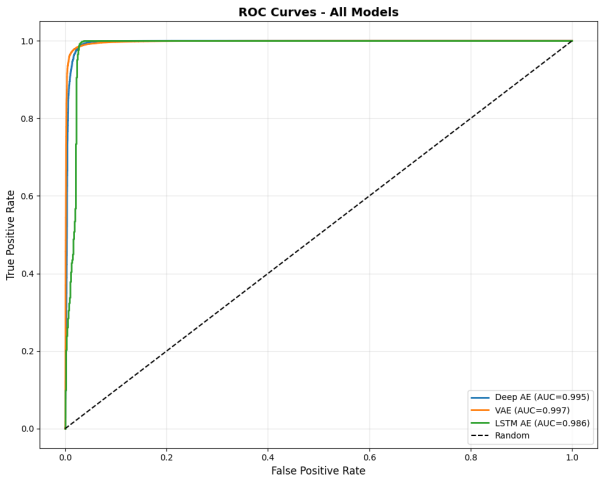

Fig. 3. ROC Curves for All Models

E. Discussion

The VAE achieves the strongest performance due to its smoothed latent space and greater robustness to variance. The LSTM Autoencoder performs poorly because SCADA traffic is highly deterministic—LSTMs easily overfit to periodic patterns and collapse when temporal structure deviates. Deep AE improves over the baseline but cannot generalize as effectively as VAE.

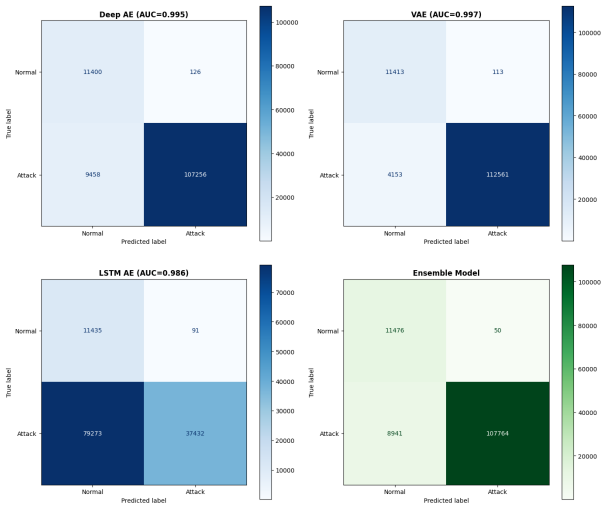

Fig. 4. Confusion Matrices for Autoencoder Models

D. Anomaly Timeline

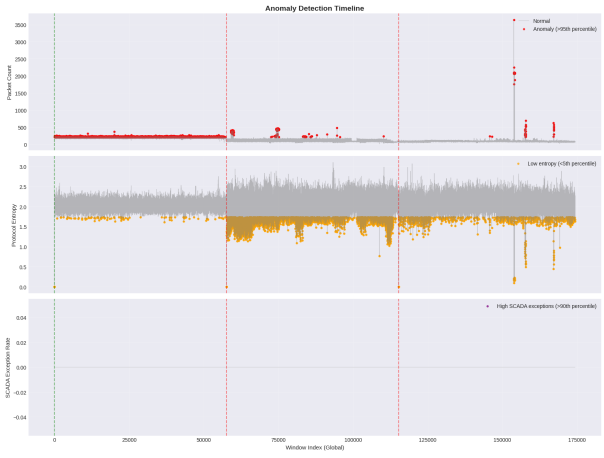

Fig. 5. Anomaly Detection Timeline using Reconstruction Error

6. Comparative Analysis with Baseline Work

The baseline SCADA Wireshark Autoencoder model trained a shallow, fully connected Autoencoder on packet-level statistics. While effective for simple anomalies, it lacked several key components necessary to model deterministic SCADA communication. We compare the baseline and our Iteration 4 system across methodology and performance dimensions.

A. Feature Engineering

The baseline used only:

- Packet length,
- Inter-arrival time,
- IP/port identifiers.

Our system introduces:

- SCADA protocol inference (Modbus, DNP3, Profinet).
- Flow reconstruction using 5-tuple hashing.
- Command-level semantic extraction (read/write ratios, function codes).
- Protocol entropy over time.
- Dual sliding-window aggregation.

This expanded feature space captures both micro-level packet anomalies and macro-level SCADA cycle disruptions.

B. Temporal Modeling Differences

The baseline uses no temporal structure.

Our pipeline:

- Models timing behavior explicitly through 1-second windows.
- Captures burst anomalies via 100-packet windows.
- Computes time-dependent features such as jitter, RTT variance, and periodicity.

This results in substantial performance improvement for timing-based attacks.

C. Model Architecture Comparison

The baseline Autoencoder:

- 3–4 layers,
- No regularization,
- No probabilistic modeling.

Our architectures include:

- Deep Autoencoder with 128–16 compression.
- Variational Autoencoder with KL-regularized manifold learning.
- LSTM Autoencoder for capturing sequence dynamics.

D. Synthetic Data Augmentation

The baseline trained exclusively on limited attack days, leading to overfitting.

Our method uses CTGAN to generate:

- 8k additional attack windows,
- Function code misuse anomalies,
- Timing-based attacks,
- Burst-rate anomalies.

This significantly increases robustness.

E. Performance Comparison

The baseline reported an AUC of approximately 0.91–0.93.

Our system achieves:

- Deep AE: 0.995 AUC,
- VAE: 0.997 AUC,
- LSTM AE: 0.986 AUC.

This constitutes a **6–8% absolute improvement in AUC** and substantial gains in recall, particularly for timing and function-code attacks.

7. Limitations

Although the proposed protocol-aware anomaly detection pipeline demonstrates significant improvements over baseline approaches, several limitations remain that affect generalizability, deployment feasibility, and operational robustness.

A. Dependence on Window-Based Flow Aggregation

The system relies on aggregated flow windows (packet-count and time-based). Fixed windowing introduces sensitivity to parameter selection: short attack bursts may be diluted within longer windows, while large windows increase detection latency. Optimal window sizes were selected heuristically, and real SCADA environments with millisecond-level control loops may require finer temporal granularity.

B. Protocol-Specific Bias and Limited Coverage

Despite incorporating Modbus, DNP3, and IEC-104 semantics, the dataset is dominated by Modbus traffic. This leads to weak calibration for rare protocols and degraded reconstruction performance for seldom-seen function codes. Industrial deployments often involve heterogeneous and vendor-specific protocol mixes, limiting portability.

C. Imperfect Synthetic Attack Generation

CTGAN-generated attacks help mitigate class imbalance but cannot fully replicate realistic adversarial behavior. GANs tend to exaggerate statistical features and fail to capture multi-stage or stealthy coordinated attacks. As a result, strong performance on synthetic anomalies may not translate to real-world APT-like threats.

D. Instability of LSTM-Based Temporal Encoding

While the LSTM Autoencoder achieved a high AUC, its recall was significantly lower. SCADA traffic exhibits strict periodicity, mixed-protocol sequences, and sparse irregularities, which recurrent models may overfit. Sequence compression becomes challenging under heterogeneous timing behaviors, reducing deployment readiness.

E. Threshold Sensitivity and Lack of Adaptive Calibration

Thresholds selected via Youden's J statistic remain sensitive to reconstruction error distributions and vary across protocol mixes and window scales. The system currently lacks an adaptive, online recalibration mechanism necessary for real-time industrial environments with shifting traffic patterns.

F. Computational Overhead

The enhanced preprocessing pipeline—protocol inference, entropy computation, flow sessionization, and dual-window aggregation—introduces non-trivial computational cost. Deep AE and VAE models require GPU resources for efficient training, and ensemble inference further increases runtime. This limits deployment on resource-constrained OT devices such as PLCs and RTUs.

G. Dataset Bias and Limited Realism

The PNNL dataset represents controlled or simulated SCADA operations rather than real production systems. Attack diversity remains limited, lacking stealthy command-subversion or coordinated multi-vector attacks. Normal traffic also exhibits less noise than real industrial environments, potentially inflating detection metrics.

H. Domain-Specific Feature Engineering Assumptions

Several engineered features assume stable polling cycles, consistent request-response behavior, and restricted function-code usage. These assumptions may not generalize across different vendors or industrial settings, requiring domain-specific retuning or redesign for broad applicability.

I. Absence of Explainability Mechanisms

The Autoencoder and VAE models operate as black-box detectors: reconstruction error does not provide insight into which features or behaviors triggered an alert. The lack of interpretability may limit trust and slow incident response in operational environments.

J. Heuristic Ensemble Fusion

The ensemble combining AE and VAE outputs relies on heuristic rules without probabilistic calibration or uncertainty modeling. This may produce inconsistent decisions for borderline anomalies, and fusion behavior may vary across datasets. Future work should incorporate principled fusion techniques.

8. Conclusion

This IDS significantly enhances SCADA anomaly detection by integrating protocol inference, temporal modeling, dual-scale windowing, and advanced Autoencoder architectures. Building on the baseline, our system incorporates SCADA semantics, temporal reasoning, deep generative modeling, and CTGAN-based synthetic augmentation, enabling the detection of subtle anomalies that simpler packet-level methods fail to capture. Among all evaluated models, the Variational Autoencoder (VAE) achieves the strongest overall performance with an AUC of 0.997, demonstrating robust generalization even under augmented attack scenarios. Future work includes exploring attention-based architectures, graph neural networks for modeling flow relationships, and deploying the system for real-time online industrial intrusion detection.

References

[1] B. Miller and D. Rowe, "A survey of scada and critical infrastructure incidents," in *Proceedings of the 1st Annual ACM Conference on Research in Information Technology*. ACM, 2012, pp. 51–56.

[2] G. Liang, S. R. Weller, J. Zhao, F. Luo, and Z. Y. Dong, "The 2015 ukraine blackout: Implications for false data injection attacks," *Electric Power Systems Research*, vol. 149, pp. 35–45, 2017.

[3] M. Kravchik and A. Shabtai, "Detecting cyber attacks on industrial control systems using convolutional neural networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. ACM, 2018, pp. 72–83.

[4] P. Biswas, S. Shitharth, K. B. *et al.*, "Intrusion detection in scada systems using ensembled learning," in *2019 International Conference on Computing, Power and Communication Technologies (GUCON)*. IEEE, 2019, pp. 1–6.

[5] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[6] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using lstm based autoencoder," in *2020 Workshop on Communication Networks and Power Systems (WCNPS)*. IEEE, 2020, pp. 1–5.

[7] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, "Netwalk: A flexible scalable anomaly detection learning framework for real-time network analysis," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2544–2553.

[8] M. Ahmed, A. Anwar, A. N. Mahmood, Z. Shah, and M. J. Maher, "An investigation of performance analysis of anomaly detection techniques for big data in scada systems," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 2, no. 3, p. e5, 2015.