From Red Flags to Detection Rules: An LLM-driven Pipeline for Real-Time GOOSE Intrusion Detection and Prevention

Lucas A. Martins[1], **Camilla B. Quincozes**[1], **Giovanni Siervo**[1]
Silvio E. Quincozes[1,2], **Marcelo Caggiani Luizelli**[2]

[1] Universidade Federal de Uberlândia (UFU) – Uberlândia, Brasil
[2] Universidade Federal do Pampa (UNIPAMPA) – Alegrete, Brasil

{lucas.martins, camillaquincozes, sequincozes, gsiervo}@ufu.br

marceloluizelli@unipampa.edu.br

Abstract. *Specification-based Intrusion Detection Systems (IDSs) are widely used in IEC 61850 substations, but rely on manually crafted rules derived from expert knowledge. This paper presents an LLM-driven pipeline that automates the generation of detection rules for real-time GOOSE intrusion detection and prevention. The approach uses labeled communication samples to identify red flags and transform them into intrusion-detection rules. The pipeline supports plug-and-play execution, reproducibility, and parameterized control. A proof-of-concept using the ERENO dataset shows that the generated rules effectively detect anomalous behavior with low operational overhead, indicating that LLM-driven automation is a viable approach for specification-based IDS.*

1. Introduction

Digital substations based on the IEC–61850 standard face several cybersecurity challenges and are increasingly exposed to evolving cyberattacks, including Denial-of-Service (DoS), Message Injection, and Masquerade attacks [Malik et al. 2022]. In this context, Intrusion Detection Systems (IDSs) play a fundamental role in ensuring the reliability, safety, and operational continuity of such critical infrastructures [Quincozes et al. 2021].

Among the existing detection approaches, specification-based IDSs are widely adopted in IEC–61850 environments due to their low computational overhead and interpretability. These systems rely on expert-defined rules that describe the expected behavior of network communications, flagging deviations as potential intrusions [Quincozes et al. 2021]. Several works have proposed rule sets targeting the Generic Object Oriented Substation Event (GOOSE)-based communication, covering aspects such as protocol integrity, temporal consistency, and network behavior [Hong and Liu 2019,Hong et al. 2014a,Hong et al. 2014b, Yang et al. 2016a, Yang et al. 2016b, Kwon et al. 2015].

Despite their relevance, specification-based IDSs present a fundamental limitation: the dependence on manually crafted rules derived from domain expertise. This process is inherently time-consuming, difficult to scale, and hard to adapt to new attack patterns or evolving system behaviors. Moreover, prior works have primarily focused on proposing and qualitatively analyzing such rules, with limited efforts toward systematic, data-driven rule generation or automated validation pipelines.

Recently, the availability of realistic intrusion detection datasets, such as ERENO–IEC–61850 [Quincozes et al. 2022], has enabled more structured experimentation by

providing labeled GOOSE communication traces containing both normal behavior and multiple attack classes. These datasets create an opportunity to bridge the gap between data-driven analysis and specification-based IDS design, enabling the extraction of behavioral patterns directly from observed communication samples.

In parallel, Large Language Models (LLMs) have emerged as powerful tools for pattern extraction, reasoning, and structured knowledge generation from complex data representations. While widely explored in natural language processing and software engineering tasks, their potential for automating the generation of specification-based detection rules from network data remains largely unexplored.

This work presents an LLM-driven pipeline that automates the generation of detection rules for real-time GOOSE intrusion detection and prevention. The approach consumes labeled communication samples to derive *red flags*, which are transformed into executable intrusion-detection rules and deployed in a simulated programmable switching environment using realistic traffic generated by the ERENO framework. The pipeline supports plug-and-play execution, reproducibility, and parameterized control, reducing reliance on expert-defined rules while improving adaptability to evolving attack scenarios.

2. Specification-based IDSs for GOOSE Communication in Substations

The IEC–61850-8-1 standard specifies communication networks and systems for digital substations, including the GOOSE protocol, which supports time-critical protection and control operations through a publisher/subscriber model over Ethernet [Commission 2003]. A typical GOOSE frame is shown in Figure 1. These messages carry operational fields such as `stNum`, `sqNum`, timestamps, and dataset references, whose consistency is essential for correct system behavior. However, since GOOSE was not originally designed with strong native security mechanisms, it is vulnerable to cyberattacks such as

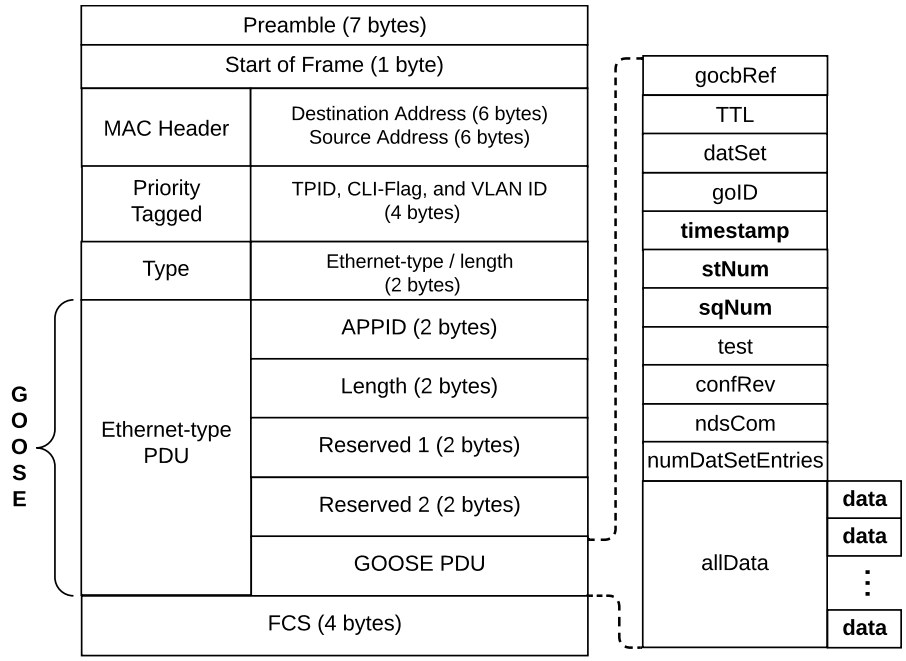

Figura 1. Ethernet frame of GOOSE messages.

injection, replay, and denial-of-service, which may compromise the coordination of protection functions [Malik et al. 2022].

In this context, specification-based detection is particularly suitable because many GOOSE fields naturally induce behavioral invariants that can be encoded as rules. Prior works explored this idea by checking structural fields such as destination MAC, `TPID`, `ethType`, `gooseAppid`, and `timeAllowedToLive` [Hong and Liu 2019, Hong et al. 2014a, Hong et al. 2014b, Yang et al. 2016a, Yang et al. 2016b], as well as consistency-related fields such as `gocbRef`, `datSet`, `goID`, `t`, `StNum`, and `SqNum` [Hong and Liu 2019, Hong et al. 2014a, Hong et al. 2014b, Yang et al. 2016b].

The literature also includes rules over communication dynamics, such as message frequency, recency, bytes per second, and packets per second [Kwon et al. 2015, Yang et al. 2016b]. Together, these works show that protocol fields, inter-message relationships, and traffic-level metrics can all serve as the basis for GOOSE intrusion-detection rules.

Because such rules are typically lightweight and directly evaluated over message attributes and simple traffic statistics, they are particularly attractive for real-time environments such as digital substations, where detection latency and operational overhead must remain low. The main challenge, however, is that these rules still depend on expert knowledge to be manually specified and tailored to specific attack types [Quincozes et al. 2021]. To support controlled experimentation, the ERENO–IEC–61850 dataset provides labeled samples of normal and malicious GOOSE traffic generated under realistic substation conditions [Quincozes et al. 2022]. This enables the systematic analysis of communication patterns and offers a practical basis for automatically deriving detection logic from observed data rather than specifying rules entirely by hand.

3. Proposed Architecture

In contrast to traditional specification-based IDSs, in which detection rules are manually defined by domain experts, our proposal automates the derivation of detection logic directly from labeled IEC 61850 GOOSE communication samples. The architecture implements an end-to-end LLM-driven pipeline that transforms observed benign and malicious traffic patterns into executable intrusion-detection rules and evaluates them in a simulated real-time environment. Its workflow is organized into four stages: *source ingestion*, *red flag extraction*, *rule generation*, and *simulated deployment*.

Figure 2 presents the overall workflow. The pipeline starts from a labeled GOOSE dataset, which is used as the sole input source for the generation process. These samples are first organized and converted into structured descriptions that preserve protocol-level, temporal, and derived features. The LLM then analyzes these descriptions to identify suspicious conditions, behavioral inconsistencies, and discriminative cues associated with malicious or benign traffic. Rather than producing code directly from the raw samples, the architecture first extracts these intermediate *red flags*, which improves traceability and makes the generation process easier to inspect.

Next, the extracted red flags are translated into executable Python detection rules. Each rule is expressed as a lightweight conditional statement over one or more packet attributes or derived metrics, following the same general spirit as traditional specification-based IDS rules. The difference is that, instead of being manually handcrafted, these rules

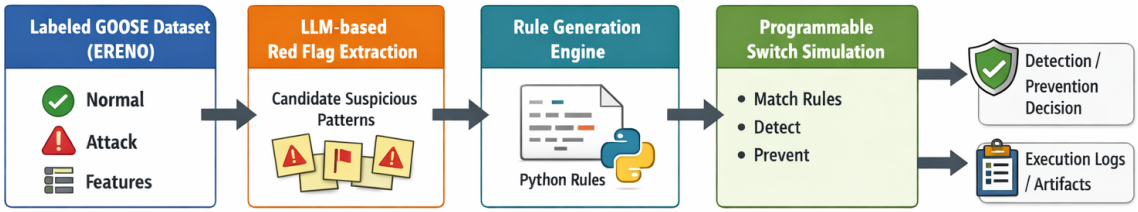

Figura 2. Overview of the proposed architecture.

are automatically inferred from labeled observations, reducing the dependence on expert-written specifications while still preserving interpretability.

After generation, the rule set is deployed into a simulated programmable-switch environment that sequentially inspects incoming GOOSE samples and applies the synthesized logic to each packet. For every processed sample, the simulator determines whether the traffic should be considered benign or suspicious and records the corresponding decision. Although this component does not aim to reproduce the full behavior of a hardware switch, it provides a controlled environment for assessing the feasibility of rule execution under real-time constraints in time-sensitive scenarios such as digital substations.

Besides the online decisions, the architecture also produces artifacts that support inspection and reproducibility, including the extracted red flags, the final `rules.py` file containing all generated rules, and annotated traces with per-rule activation flags and global `BLOCK`/`ALLOW` decisions. Overall, the proposed architecture defines a complete path from labeled GOOSE samples to executable detection logic, bridging data-driven analysis and specification-based detection in a form that is both reproducible and inspectable.

4. Tool Demonstration: Parameters, Execution and Outputs

This section describes how to execute the proposed pipeline, from environment setup to LLM-based rule generation and offline evaluation over GOOSE traces. The prototype implementation, including Jupyter notebooks, auxiliary modules, and configuration files, is publicly available at `https://github.com/lucastuxnet/SBRC_2026`. The same repository also provides installation, configuration, and execution instructions, as well as a step-by-step guide to reproduce the experimental results reported in this paper.

The prototype is implemented in Python 3.12 as a Jupyter notebook supported by auxiliary modules. Its dependencies, including `numpy`, `pandas`, `groq`, and `python-dotenv`, are installed from a `requirements.txt` file inside a virtual environment. Configuration parameters are read from a local `.env` file, which must define the LLM API key (`GROQ_API_KEY`); otherwise, execution is interrupted with a runtime error. To obtain a Groq API Key, please visit `https://console.groq.com/keys`. The LLM backend is a Groq-hosted model configured with deterministic decoding (temperature equal to zero), which ensures reproducible rule-generation runs.

For the proof-of-concept, the notebook uses the `ERENO-2.0-100k.csv` trace [1] generated by the ERENO Framework [Quincozes et al. 2023]. This dataset contains labeled GOOSE messages collected under normal operation and multiple attack sce-

[1]The ERENO-2.0-100k dataset: `https://www.kaggle.com/datasets/silvioerenoquincozes/ereno-2-0-100k-csv`

narios. Each row includes protocol fields such as `StNum`, `SqNum`, `gooseAppid`, and `APDUSize`, along with temporal and derived features such as `stDiff`, `sqDiff`, `timestampDiff`, and length differences. During preprocessing, the code loads a reduced version of the dataset and keeps only a semantic subset of relevant columns, including the `class` label. Columns expected by the prompts but absent from the CSV are automatically reported and ignored, ensuring that both prompts and generated rules refer only to available features.

From these labeled samples, the notebook builds structured prompts describing normal and malicious traffic, then asks the LLM to generate Python detection rules in the format `def rule<attack_class>_<name>(packet: dict) -> bool`. Each rule inspects the fields available in the selected dataset representation, such as `StNum`, `SqNum`, `timestampDiff`, and `timeFromLastChange`, and returns `True` when a packet is considered suspicious. The LLM interaction is encapsulated in a wrapper function that strips optional markdown fences from the response and applies simple retry logic in the event of rate-limit errors. The generated code is appended to a single `rules.py` file, which progressively accumulates rule functions.

For evaluation, the notebook provides utilities that read a GOOSE CSV file, restrict it to the same semantic columns used during rule generation, and convert each row into a `dict`. At runtime, the `rules` module is inspected to collect all callables whose names begin with `rule`, and each rule is applied to every packet. For each rule, a Boolean column records whether it fired on a given sample, while a global `decision` column labels packets as `BLOCK` if at least one rule returns `True`, or `ALLOW` otherwise. The resulting annotated DataFrame is then used in the detection and real-time feasibility analyses presented in Section 5. The main artifacts produced by the tool are therefore the final `rules.py` file and the `rules_output.csv`, which contains per-rule outputs. Based on this output, we also produce analytical charts and derive the global decisions.

For the SBRC demonstration, the tool will be executed on a single laptop equipped with a quad-core CPU and 16 GB of RAM, running Python 3.12 with the dependencies listed in `requirements.txt`, following the flow illustrated in Figure 2.

5. Proof-of-Concept (PoC)

This section demonstrates the feasibility of the proposed architecture using the ERENO GOOSE dataset [Quincozes et al. 2023]. We used Groq's Compound system as the language model backbone, which integrates GPT-OSS 120B and Llama 4 models with external tools such as web search and code execution, enabling access to real-time information through a unified orchestration interface.[2] The prompt was constructed through iterative interactions between the authors and the Perplexity tool,[3] without systematic prompt engineering; the results presented here can and should be improved in future work through exhaustive prompt optimization and model comparison.

5.1. Detection Performance

Table 1 summarizes the per-class results obtained by the synthesized rule set in terms of support, true positives (TP), false negatives (FN), and recall (TPR). For attack classes,

[2] `https://groq.com/compound-ai/`

[3] `https://www.perplexity.ai`

recall corresponds to the fraction of malicious samples correctly classified as *BLOCK*. For the *normal* class, it represents the fraction of benign samples classified as *ALLOW*.

Tabela 1. Per-class detection metrics on the ERENO GOOSE trace.

Class	Support	TP	FN	TPR (Recall)
grayhole	19,999	17,476	2,523	0.87
highStNum	20,000	19,982	18	1.00
injection	20,000	14,869	5,131	0.74
inverse replay	20,000	14,445	5,555	0.72
masq. fake fault	20,000	9,032	10,968	0.45
masq. fake normal	20,000	7,464	12,536	0.37
poisoned high rate	20,000	1,890	18,110	0.09
random replay	20,000	12,881	7,119	0.64
normal	39,999	16,226	23,773	0.41[4]

The best result is obtained for *highStNum* (TPR = 1.00), which directly violates a strong protocol-level invariant, making the anomaly explicit in the GOOSE state progression. *Grayhole* also achieves strong recall (0.87), confirming that the generated rules can capture attacks that introduce noticeable temporal or sequential inconsistencies. Intermediate recalls are observed for *injection* (0.74), *inverse replay* (0.72), and *random replay* (0.64), suggesting that the synthesized rules are effective at identifying a relevant portion of abnormal behavior, but still miss cases in which packets remain sufficiently compatible with expected GOOSE dynamics.

The lowest recalls are observed for the masquerade classes (*masq. fake fault*, 0.45; *masq. fake normal*, 0.37) and for *poisoned high rate* (0.09). Masquerade attacks are purposely designed to mimic legitimate `StNum`, `SqNum`, retransmission bursts, and control semantics, making them partially indistinguishable from benign communication when detection relies solely on GOOSE-side observations. For *poisoned high rate*, the poor result suggests that the malicious traffic preserves a large portion of the expected GOOSE dynamics, exposing a fundamental limit of lightweight rule conditions when the attacker behaves in a protocol-aware manner. These findings are consistent with the original ERENO assessment [Quincozes et al. 2023], in which masquerade and rate-based attacks were already the hardest cases when restricted to basic GOOSE features alone, and where the main gains came from combining GOOSE and SV data and from cross-protocol feature enrichment.

The low recall for the *normal* class (0.41) reflects a conservative rule set in which many benign packets are flagged by at least one rule. This highlights a key trade-off of the current PoC: while the synthesized rules capture attacks with strong protocol-level deviations well, improving selectivity for normal traffic and mimicry-based attacks will likely require rule refinement or richer cross-domain evidence. Figure 3 summarizes the per-class recall distribution discussed above.

5.2. Explainability of the Generated Rules

A distinctive property of the proposed approach is that, despite relying on a LLMs (typically a black-box system), the output it produces is fully interpretable. Rather than retur-

[4]For normal traffic, TPR is the fraction of samples that are *allowed* rather than *blocked*.

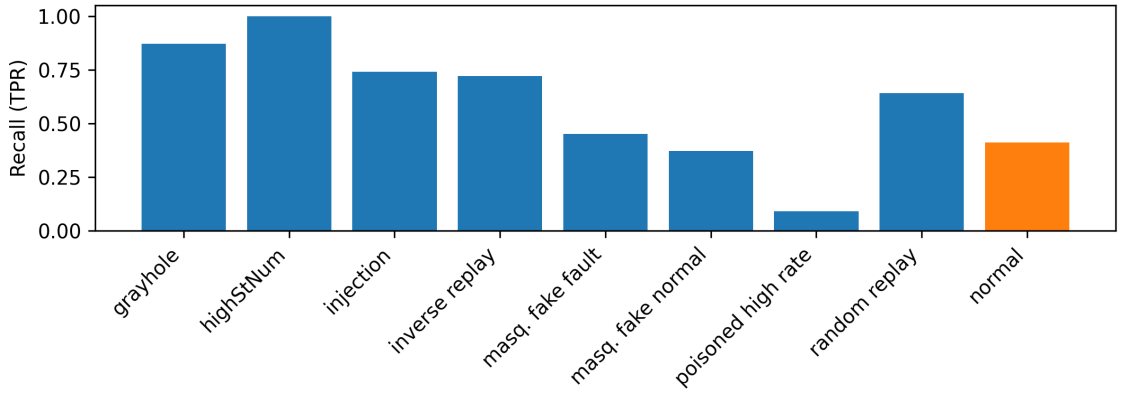

Figura 3. Per-class recall for the LLM-generated rule set on the ERENO trace.

ning a classification decision, the tool generates explicit, human-readable rules grounded in the semantic of the IEC 61850 GOOSE protocol. For instance, for the *grayhole* class, the LLM identified red flags such as:

> **RED FLAG: Sequence Reset** — Fields: `SqNum`, `StNum`. The *grayhole* class presents packets with reset `SqNum` and `StNum` values, indicating a sequence reset. This is anomalous according to GOOSE/IEC 61850 semantics, which requires sequences to be continuous and coherent.

These red flags are then translated into executable rule conditions (See Figure 4) that a human operator can read, validate, and audit. This property is particularly relevant in critical infrastructure contexts, where accountability and auditability are not optional requirements but regulatory and operational imperatives. An operator must be able to

```python
def rule_grayhole_sqnum_reset_pattern(packet: dict) -> bool:
    sq_num = packet.get('SqNum', 0)
    st_num = packet.get('StNum', 0)
    prev_sq_num = packet.get('prev_sq_num', 0)
    prev_st_num = packet.get('prev_st_num', 0)

    return (sq_num < prev_sq_num and st_num < prev_st_num)
```

Figura 4. Example of rule generated from a Red Flag (more examples can be found at *rules.py* in the project repository.

understand *why* a packet was blocked, trace that decision back to a specific protocol violation, and justify it to a security auditor or incident response team. Rule-based outputs satisfy this requirement in a way that probabilistic scores from neural classifiers do not. Furthermore, explainability supports iterative improvement: a domain expert who disagrees with a generated rule can correct it directly, without retraining a model or understanding internal weight representations.

5.3. Real-time Feasibility

The generation of red flags for 100 samples (80 attack instances, 10 per category, and 20 normal) required 3 minutes and 40.1 seconds, followed by the creation of rules in

1 minute and 33.6 seconds. As the tool's rules can be produced in any programming language, extensive compatibility with programmable switches is achieved through the integration of Generative AI. In this demonstration, Python was employed since it best suited the illustrative purpose. The overall processing time for executing these rules on the complete set of 200,000 samples is examined below in terms of per-packet latency.

Across multiple benchmark runs, the average per-packet processing latency is approximately 1.08 μs, with a minimum of 0.54 μs, a maximum of 547.29 μs, and a 99th percentile close to 2.87 μs. These values confirm that the generated rules can be evaluated within a very small time budget, remaining in the microsecond range under typical conditions. This latency profile is compatible with the time-sensitive nature of GOOSE-based protection and control traffic, supporting the feasibility of using automatically synthesized specification-style rules in real-time inspection settings. Notably, once produced, the rules behave as lightweight conditional checks, showing that LLM-assisted rule generation does not imply expensive online execution.

Compared to prior GOOSE-focused IDS approaches [Hong et al. 2014b, Yang et al. 2016a, Kwon et al. 2015, Malik et al. 2022], which typically rely on manually engineered rules or supervised models tailored to specific attack scenarios, the proposed approach automatically synthesizes interpretable rules from labeled traces, reducing manual engineering effort. It is important to emphasize that these results reflect a proof-of-concept aimed at demonstrating the feasibility of the proposed architecture, not at optimizing its performance. No fine-tuning was applied, and the prompt was constructed through iterative interactions between the authors without exhaustive prompt engineering. The results presented here therefore represent a lower bound on the tool's potential: with systematic prompt optimization, model selection, and feature enrichment, substantially better detection rates are expected in future work.

6. Final Remarks

This paper presented an LLM-driven pipeline that automates the generation of detection rules for specification-based IDS in IEC–61850 substations. Starting from labeled GOOSE samples provided by the ERENO framework, the approach extracts behavioral patterns and translates them into executable, human-readable rules evaluated in a simulated programmable switch.

The PoC results demonstrate two key differentials of the proposed tool: real-time feasibility, with per-packet latency in the microsecond range compatible with GOOSE traffic constraints, and explainability, since the generated rules are fully auditable and traceable to specific protocol violations — a critical requirement in safety-critical infrastructure where accountability and auditability are operational imperatives. Lower recall for masquerade and rate-based attacks points to the need for richer feature sets and more systematic prompt engineering, which are left for future work.

As next steps, we plan to validate the pipeline on actual programmable switch hardware, extend the evaluation to additional ERENO attack scenarios and other IEC–61850 datasets, and investigate cross-protocol feature enrichment — particularly GOOSE and Sampled Values intercorrelation — to address the harder masquerade cases. Comparison against supervised learning baselines and incremental rule-set updates in the presence of evolving attack patterns are also planned.

Referências

Commission, I. E. (2003). *Communication networks and systems in substations - Part 8-1: Specific communication service mapping (SCSM) - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3*. IET.

Hong, J. and Liu, C. (2019). Intelligent electronic devices with collaborative intrusion detection systems. *IEEE Transactions on Smart Grid*, 10(1):271–281.

Hong, J., Liu, C., and Govindarasu, M. (2014a). Detection of Cyber Intrusions Using Network-Based Multicast Messages for Substation Automation. In *Innovative Smart Grid Technologies (ISGT)*, pages 1–5. IEEE.

Hong, J., Liu, C.-C., and Govindarasu, M. (2014b). Integrated anomaly detection for cyber security of the substations. *IEEE Transactions on Smart Grid*, 5(4):1643–1653.

Kwon, Y., Kim, H. K., Lim, Y. H., and Lim, J. I. (2015). A behavior-based intrusion detection technique for smart grid infrastructure. In *2015 IEEE Eindhoven PowerTech*, pages 1–6. IEEE.

Malik, H., Alotaibi, M. A., and Almutairi, A. (2022). Cyberattacks identification in iec 61850 based substation using proximal support vector machine. *Journal of Intelligent & Fuzzy Systems*, 42(2):1213–1222.

Quincozes, S. E., Albuquerque, C., Passos, D., and Mossé, D. (2021). A survey on intrusion detection and prevention systems in digital substations. *Computer Networks*, 184:107679.

Quincozes, S. E., Albuquerque, C., Passos, D., and Mossé, D. (2023). ERENO: A framework for generating realistic IEC–61850 intrusion detection datasets for smart grids. *IEEE Transactions on Dependable and Secure Computing*, 21(4):3851–3865.

Quincozes, S. E., Passos, D., Albuquerque, C., Mossé, D., and Ochi, L. S. (2022). *ERENO: AN EXTENSIBLE TOOL FOR GENERATING REALISTIC IEC–61850 INTRUSION DETECTION DATASETS*. PhD thesis, Universidade Federal Fluminense.

Yang, Y., McLaughlin, K., Gao, L., Sezer, S., Yuan, Y., and Gong, Y. (2016a). Intrusion detection system for IEC 61850 based smart substations. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5. IEEE.

Yang, Y., Xu, H.-Q., Gao, L., Yuan, Y.-B., McLaughlin, K., and Sezer, S. (2016b). Multidimensional intrusion detection system for IEC 61850-based SCADA networks. *IEEE Transactions on Power Delivery*, 32(2):1068–1078.