Project Report

CISSAN

Collective intelligence supported by security aware nodes

D5.1 Distributed Detection of Potentially Harmful Actions

Editor: Karoly Makonyi (Savantic AB)

Contact: karoly.makonyi@savantic.se

Abstract

Deliverable D5.1 presents CISSAN efforts and results at the intersection of AI and IoT, with anomaly detection as a key technique in use. In the domain of smart grids, AI-based anomaly detection algorithms are applied for analysing network traffic and operational data from substations to detect threats and faults at both communication and operational levels. The developed anomaly detection models run locally, in network nodes, reflecting the focus of CISSAN Task 5.1 on distributed detection and – more generally – the project's focus on enabling Collective Intelligence for improved cybersecurity. Another T5.1 effort included in this document is studies in the application of a modern Large Language Model (Llama 3.1) to IoT network traffic analysis, with a goal of detecting and analysing cyberattacks with minimal preprocessing and preparations.

D5.1 also briefly presents work that fully or partially belongs to WP4: (i) the use of generative AI for producing synthetic data for training and validating intrusion detection systems, addressing such critical challenges as data scarcity and privacy; (ii) analysis of GPS data in public transport systems to identify system faults and attacks and improve the safety and resilience of transport systems; (iii) the use of anomaly detection as an ingredient in a blockchain-based framework for securely logging IoT events and producing verifiable proofs of detected anomalies. While these lines of work will be presented in detail in future CISSAN deliverables, (i) clearly supports T5.1 and (ii) and (iii) show methodological similarities, so we find it helpful to articulate the connections.

Project **CISSAN**

Public

Participants in project CISSAN are:

- University of Jyväskylä
- Bittium Wireless Ltd
- Bittium Biosignals Ltd
- Geodata ZT Gmbh
- Mattersoft
- Mint Security Ltd
- Netox Ltd
- Nodeon Ltd
- Scopesensor Ltd
- Wirepas Ltd
- Councilbox Ltd
- Affärsverken Karlskrona AB
- Arctos Labs
- Clavister AB
- Blekinge Tekniska Högskolan
- Blue Science Park
- Savantic AB
- Techinova AB

CISSAN – Collective intelligence supported by security aware nodes

D5.1 Distributed detection of potentially harmful actions

Editor: Karoly Makonyi (Savantic AB)

Project coordinator: Alexey Kirichenko (University of Jyväskylä)

CELTIC published project result

Executive Summary

The convergence of artificial intelligence (AI) and IoT technologies lies at the core of innovative cybersecurity solutions for interconnected systems. Deliverable D5.1 presents CISSAN efforts and results which harness this synergy to address the pressing security and reliability challenges of modern systems, with anomaly detection as a key technique in use.

In the domain of smart grids (CISSAN Use Case 2), AI-based anomaly detection algorithms are applied for analysing network traffic and operational data from substations to detect threats and faults at both communication and operational levels. The developed anomaly detection models run locally, in network nodes, reflecting the focus of CISSAN Task 5.1 on distributed detection and – more generally – the project's focus on enabling Collective Intelligence for improved cybersecurity. Another T5.1 effort included in this document is studies in the application of a modern Large Language Model (Llama 3.1) to IoT network traffic analysis, with a goal of detecting and analysing cyberattacks with minimal preprocessing and preparations.

D5.1 also briefly presents work that fully or partially belongs to WP4: (i) the use of generative AI for producing synthetic data for training and validating intrusion detection systems, addressing such critical challenges as data scarcity and privacy; (ii) analysis of GPS data in public transport systems (in the scope of CISSAN Use Case 1) to identify system faults and attacks and improve the safety and resilience of transport systems; (iii) the use of anomaly detection as an ingredient in a blockchain-based framework for securely logging IoT events and producing verifiable proofs of detected anomalies. While these lines of work will be presented in detail in future CISSAN deliverables, (i) clearly supports T5.1 and (ii) and (iii) show methodological similarities, so we find it helpful to articulate the connections.

Anomaly Detection in IoT-based Smart Grids

This study investigates anomaly detection in IoT-enabled smart grids using AI-based algorithms, focusing on network traffic and operational data from grid substations. The experiments aimed to detect anomalies at both communication and operational levels.

Network data was captured in the PCAP format, and two levels of analysis were conducted on it:

- **Network-Level Analysis**: The AI algorithms processed the IP headers from the SCADA and MQTT packets, using features like Header Length, TTL, and a custom Inter-Arrival Time metric to account for timing.
- **Operational-Level Analysis**: The sensor values in the SCADA and MQTT payloads were filtered based on Information Object Addresses (IOA) and mapped to signals of interest.

At the communication (network) level, the AI models were trained on one million SCADA packets using selected IP headers. To test anomaly detection, the Total Length feature was manipulated during inference. The results showed that the models were able to identify deviations from normal behaviour via anomaly scores crossing the decision boundaries.

The experiments at the operational level targeted substation health monitoring through signal analysis to detect system failures and attacks undetected at the communication level. Anomalous signals of two types were tested:

- **Frozen-Value signal**: Signal values were frozen over multiple consecutive packets.
- **Mean-Shift Attack**: Signal values were manipulated using a weighted moving average and one standard deviation, mimicking stealthier attacks.

The AI models were trained on rolling-window statistical features (mean, standard deviation, etc.) from the measured signals (like Currents, Power Factor, etc). The results showed the successful detection of both types of anomalous signals, with the anomaly scores crossing the predefined thresholds during the manipulation periods.

The novelty of our approach is in performing analysis on data generated at secondary substations, close to the source where higher-frequency data samples are accessible, to detect even subtle deviations in operational and communication / network behaviours. Achieving this requires lightweight, efficient algorithms that can run locally on resource-constrained edge devices.

The primary objective in CISSAN Use Case 2 (UC2) is to automate grid monitoring and deliver intelligent, actionable insights to operators, addressing the challenges brought about by the changing dynamics of power grids.

While the current experiments are focusing on anomaly detection run locally within individual substations, future work aims to develop a global framework for aggregating and contextualizing observations across substations, enabling a holistic view of the grid performance. Advanced explainability features will be integrated to help operators identify root causes of anomalies and make informed decisions to improve grid reliability and efficiency.

Llama analysis in detecting IoT-targeting cyberattacks

The study explores the potential of Llama 3.1, a Large Language Model (LLM), to detect cyberattacks in IoT network traffic data, focusing on its ability to analyse raw, heterogeneous data without fine-tuning or data preprocessing. LLMs have shown promise in cybersecurity by understanding cyberattack patterns and identifying vulnerabilities. In CISSAN, we evaluated Llama 3.1's performance in detecting 12 attack types, leveraging its ability to interpret individual network packets (in the JSON format) extracted from the Edge-IIoTset dataset.

The study highlights the observed Llama 3.1's strengths, such as understanding network protocols, identifying suspicious IPs and ports, and analysing human-readable data. However, the limitations include hallucinations, particularly with encrypted payloads, and reduced accuracy in discerning attack signals amidst ambiguous data. Notably, the model's ability to correctly reason was high at 94%, but its occasional inaccuracies necessitate human oversight for validation.

In the qualitative analysis, the model demonstrated contextual awareness of Industrial Internet of Things (IIoT) environments and protocol-specific functions, underlining its ability to process raw network data meaningfully.

In cases when LLM adaptation or data preprocessing are not justified or infeasible, our work contributes to exploring the limits of LLMs in cyberattack detection and to understanding preferred ways of their application. We are planning to extend the scope of our experiments and the datasets used in those, including synthetic data generated in WP4 (briefly presented in Section 4 and the paragraphs that immediately follow).

Generative AI for Generating Cybersecurity Data

In T4.2, we investigated the use of generative AI, particularly Generative Adversarial Networks (GANs), to generate synthetic cybersecurity data aimed at addressing challenges with building intrusion detection systems (IDS). Real-world cybersecurity data is often limited, imbalanced, and sensitive to privacy concerns, making synthetic data a valuable alternative. This study evaluates various synthetic data generation methods, including non-AI techniques (such as Random Oversampling and SMOTE) and AI-based methods (like GANs and Variational Autoencoders). The focus is on their effectiveness, data fidelity, and ability to handle class imbalance.

Key findings indicate that while traditional methods, such as SMOTE and Cluster Centroids, effectively maintain data distribution and class balance, they often do not generate novel data points. In contrast, AI-based methods, such as Conditional Tabular GAN (CTGAN) and Tabular Variational Autoencoders (TVAE), excel at producing high-quality synthetic data closely resembling real-world statistical properties, although they may need additional techniques to address the class imbalance. The study also emphasizes the importance of mutual information for feature selection and the use of evaluation metrics like statistical similarity, performance, and class balance.

The research concludes that CTGAN and TVAE are particularly effective for cybersecurity applications, providing a balance between data realism and utility, and suggests leveraging GANs to simulate evolving cyberthreats, enhance IoT data privacy, and improve real-time IDS. This study offers a comprehensive framework for selecting synthetic data generation methods tailored to specific cybersecurity needs.

Analysis of GPS data in the public transport domain

GPS interference, spoofing, and jamming are growing cybersecurity concerns, particularly affecting aviation and public transport. Real-time passenger information and vehicle location systems in public transport depend on accurate, frequently updated positioning data, primarily from such GNSS technologies as GPS or alternative sources.

The CISSAN project has been exploring methods and models for detecting anomalies in GPS data, starting with the real-time bus position data from Tampere, Finland. Given the large volume of GPS data collected daily (5-7 million coordinates), efficient and robust anomaly detection methods are necessary. We are evaluating various algorithms, including statistical and ML models.

Initial results allowed us to identify in the data frequent small errors (likely due to environmental interference, device sensitivity, or route characteristics) and occasional large errors (likely resulting from signal loss, device malfunction, or data glitches). Future work will focus on developing and validating methods optimized for real-time anomaly detection in public transport data.

Network Logging System

Anomaly detection is used as an ingredient in a blockchain-based framework being developed in T4.3 for securely logging IoT events and producing verifiable proofs of detected anomalies, supporting, e.g., audits and incident investigations in IoT networks. The framework enables multiple agents to transmit IoT events in a standardized format to a custom blockchain network. These events are initially received and processed by anomaly detection models in so-called *Hub nodes*. A *Master node* creates and extends a canonical blockchain of events, which every *Hub node* synchronizes and verifies. As an additional assurance, the event blockchain is also timestamped on a public blockchain (e.g., Bitcoin).

The anomaly detection part of the framework, which belongs to T5.1, will be developed further in the 2nd half of the project. Connecting to the synthetic data generation efforts presented above, we note that GANs are planned to be used for improving the anomaly detection models for the IoT event logging framework.

List of Authors

In alphabetic order by partner name:

Dure Adan Ammara, Blekinge Tekniska Högskola

Wissam Aoudi, Clavister

Alberto Doval, Councilbox

Teemu Kemppainen, Mattersoft

Anders Lidén, Clavister

Karoly Makonyi, Savantic AB

Sara-Päivi Paukkeri, University of Jyväskylä

Pasi Tapanainen, University of Jyväskylä

Table of Contents

List of Figures

List of Tables

Abbreviations

AI – Artificial Intelligence

GAN – Generative Adversarial Network

GNSS - Global Navigation Satellite System

GPS – Global Positioning System

IoT – Internet of Things (IoT is used here fully inclusively, including Industrial Internet of Things (IIoT), Operational Technologies (OT), relevant networking technologies)

IP – Internet Protocol

MQTT – Message Queueing Telemetry Transport

RFID – Radio Frequency Identification

SCADA – Supervisory Control and Data Acquisition

TTL – Time to Live

WMA – Weighted Moving Average

1 Introduction

Artificial Intelligence (AI) lies at the heart of advancing cybersecurity in interconnected systems and critical infrastructures, offering powerful tools to address the growing complexity of IoT and OT networks, such as smart grids (CISSAN Use Case 2). In smart grids, AI-driven anomaly detection enables real-time identification of technical faults and cyberattacks, improving the resilience and operational efficiency of grid systems. The anomaly detection models presented in Section 2 run locally, in network nodes, reflecting the focus of CISSAN T5.1 on distributed detection and – more generally – the project’s focus on enabling Collective Intelligence for improved cybersecurity.

Large Language Models (LLMs) show potential as tools for detecting malicious activities in heterogeneous, high-volume data streams. Another T5.1 effort included in this document (Section 3) is studies in the application of a modern Large Language Model (Llama 3.1) to IoT network traffic analysis, with a goal of detecting and analysing cyberattacks with minimal preprocessing and preparations.

D5.1 also briefly presents work that fully or partially belongs to WP4: (i) the use of generative AI for producing synthetic data for training and validating intrusion detection systems, addressing such critical challenges as data scarcity and privacy (Section 4); (ii) analysis of GPS data in public transport systems (in the scope of CISSAN Use Case 1) to identify system faults and attacks and improve the safety and resilience of transport systems (Section 5). While these lines of work will be presented in future CISSAN deliverables, (i) clearly supports T5.1 and (ii) explores anomaly detection approaches, so we find it helpful to articulate the connections in D5.1.

The presented results and findings highlight AI’s transformative potential to manage emerging risks and enhance the security of modern digital ecosystems.

2 Network and operational anomalies in smart grid

2.1 Introduction

The integration of IoT and digital communications within electrical grids has paved the way for smart grids, transforming traditional power distribution systems into complex, interconnected networks. This evolution enables greater efficiency, real-time monitoring, and flexibility but also brings new security challenges. Detecting and managing anomalies—ranging from technical faults to sophisticated cyberattacks—is critical to ensuring the resilience and reliability of these systems. Anomalies within smart grids can signal irregular activities, such as data tampering, unauthorized access, or equipment malfunction, which may compromise the entire network's stability if left undetected.

This section describes the application of AI-driven anomaly detection techniques through experiments on network and operational data from several substations. Using data collected from IoT-enabled devices, these experiments simulate potential failure- and attack signatures on the network and operation data. This includes scenarios such as frozen-value and mean-shift failures and attacks, to evaluate the models' ability to detect subtle and complex anomalies. The results underscore the importance of adaptable and layered detection models in managing the evolving cyber risks associated with modern smart grids. This work introduces components of a distributed system that will be realized in the second half of the project.

By examining the advantages, challenges, and performance of different anomaly detection models, this document aims to provide insights into developing resilient, scalable solutions for anomaly detection in smart grids, with the goal of improving security, operational efficiency, and system reliability across electrical infrastructure networks.

2.2 Objective of this Section

The objective of this section is to present viable approaches to AI-based monitoring and highlight the hybrid approach as likely the most plausible one. Furthermore, this section presents the reader with experimental results and methodologies for AI-based monitoring of IoT-based power grids using real-world data.

2.3 Detection Approaches

There are various anomaly detection models available in the literature, ranging from fully centralized approaches to decentralized models.

A *centralized anomaly detection model* involves collecting data from all substations and analyzing it at a central location. This approach allows for a comprehensive analysis, as aggregating data from all stations facilitates the identification of patterns that might be missed if analyzed in isolation. Additionally, it offers scalability, with the ability to scale up processing power and storage at the central location to accommodate increased data loads as the number of stations grows. However, this model has its drawbacks, including potential delays due to data transfer time, especially in low-bandwidth or high-latency environments. The model's reliance on network connectivity means that disruptions can delay or even halt data collection and anomaly detection. Moreover, constantly transmitting large amounts of data from multiple stations can consume significant network bandwidth, and the central server or processing center becomes a single point of failure, making the entire system vulnerable if an issue arises.

In contrast, a *distributed anomaly detection model*, where each node (station) performs anomaly detection independently, addresses some of these challenges. By sending only anomaly reports or summaries to a central location, this model significantly reduces the amount of data transmitted over the network, leading to lighter bandwidth usage. It also eliminates the single point of failure, as each station operates independently, ensuring that a failure at one station or the central server does not impact others. Additionally, nodes can continue functioning autonomously even if they lose connectivity with the central server. Monitoring and maintaining the functionality and consistency of all nodes also requires more complex systems.

Hybrid approaches, such as hierarchical anomaly detection, seek to combine the strengths of both centralized and distributed models [1]. In hierarchical anomaly detection, each node performs initial anomaly detection on its data, focusing on clear, straightforward anomalies that require limited computational resources. The detected anomalies or summarized data are then sent to a central server for deeper, more comprehensive analysis, allowing for the identification of broader patterns or correlations between stations. This approach is efficient, reducing the amount of data sent to the central server and saving bandwidth while combining the immediacy of local detection with the broader perspective of centralized analysis. It is also scalable, as local processing lightens the load on the central server. However, it introduces complexity, requiring careful coordination between local and central algorithms, and nodes still need some level of processing power, though less than in a fully distributed model. The result of the process will be presented to the operators supporting his decision on the necessary action.

Each substation supplies electricity to a diverse range of end customers, which may include households, industrial zones, office buildings, or a mixture of these. These distinct areas often exhibit varying power demands and unique consumption patterns. Consequently, a single standardized model may not be suitable for all substations.

Different models were considered to use and finally a lightweight model was selected to use that is able to run on each substation, even with limited hardware resources. A primary advantage of this local monitoring is the access to data and measurements at a fine-grained level thanks to running analysis closer to the data source. However, monitoring only individual substations does not offer a comprehensive view of the entire electricity grid’s status. Therefore, each substation’s status should be sent to a central node in a higher layer, allowing for event aggregation and semantic correlation analysis, as well as presentation to operators. In this setup, a hybrid anomaly detection model serves as a suitable solution for our needs.

2.4 Anomaly Detection in IoT-based Smart Grids

A series of anomaly-detection experiments have been performed using AI-based algorithms on data received from Affärsverken. This section outlines the different steps in these experiments, describes the methods used, and presents experimental results and lessons learned.

2.4.1 Data preparation

The datasets from Affärsverken were provided in the form of network captures in pcap format and originated from three stations. Experiments were conducted on all three stations, both at the network level by analysing IP headers and at the operational level by monitoring signal data.

On the network level, Clavister’s AI algorithm PASAD [2, 3, 4, 5, 6] was applied on network data by processing selected IP headers of both MQTT and SCADA packets. Filtering IP traffic was based on source address, and selected headers included Header Length, Total Length, TTL, Protocol, Destination Address, etc., in addition to a crafted feature called Inter-Arrival Time, which measures the time gap between two consecutive packets to incorporate the time aspect in the analysis.

On the operational level, the algorithm was applied on sensor and signal values contained in the payloads of MQTT and SCADA messages. The filtering in this case was based on Information Object Address (IOA), and spreadsheets provided by project partners were used to map the IOA to signals of interest.

2.4.2 Networks-Level Experiment

In this experiment, we have trained and tested a PASAD model on selected IP headers of packets containing SCADA traffic.

Training of PASAD models on IP data based on the selected headers has been done on enough packets (1 million in this experiment) to ensure that the dynamics of the device or network node under analysis have been captured. To simulate an anomaly during the inference phase, we have manipulated the Total Length feature by changing a sequence of consecutive values to a constant 1000. Figure 2.1 shows the records of the original Total Length, highlighted in blue, as well as the manipulated records, highlighted in red.

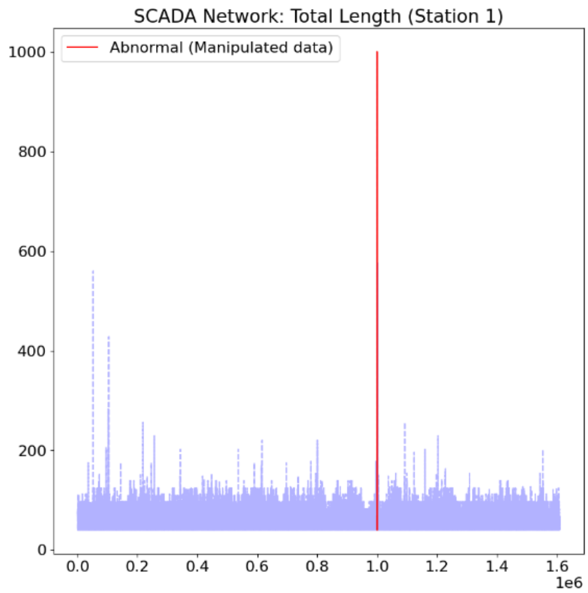


Figure 2.1: Total Length feature including manipulated values.

Figure 2.2 shows a 3D representation of how a PASAD model works. The figure shows the model's decision boundary (represented by the ellipsoid) and the classification of data records (green points) under normal conditions. If the communication behaviour of the monitored node conforms to the trained model and learnt patterns, the corresponding data instances are supposed to lie inside the decision boundary. In the case of anomalous behaviour, on the other hand, the datapoints will depart away from the datapoint cluster and cross the boundary. This is demonstrated in Figure 2.3 where PASAD could successfully detect the manipulation of the Total Length feature. The line graph in Figure 2.4 displays the so-called departure scores produced by the model, where each score represents, in a rough sense, the location of the corresponding datapoint with respect to the decision boundary. The model generates an event whenever the departure score crosses a predefined threshold.

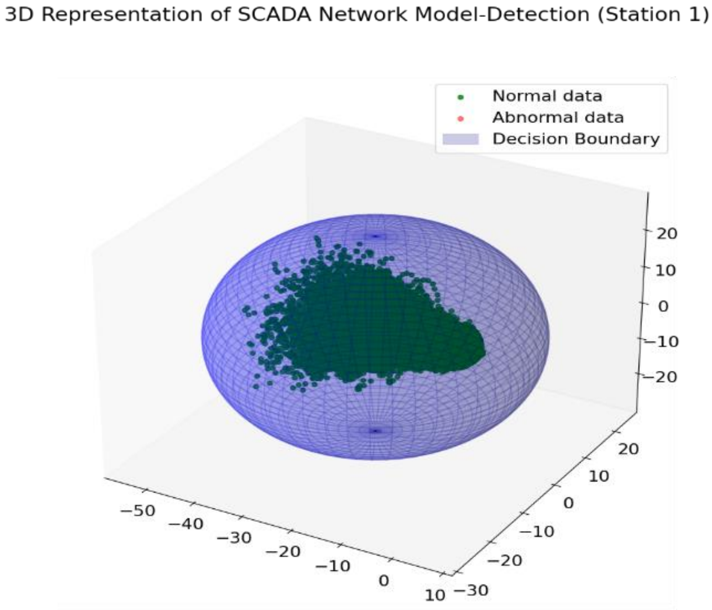


Figure 2.2. 3D representation of SCADA network in station 1.

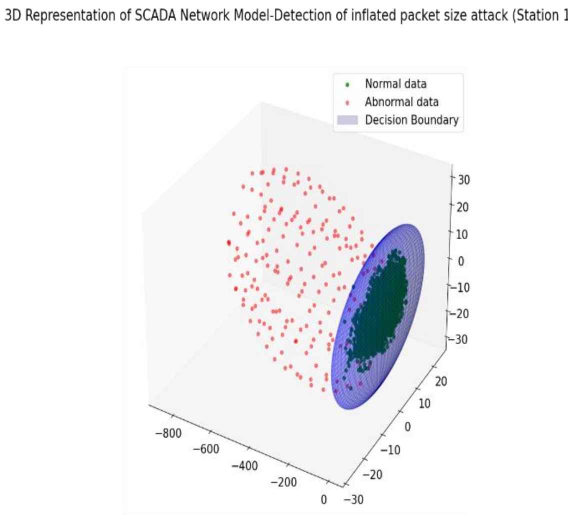


Figure 2.3. 3D visualization of detecting the Total Length attack on station 1.

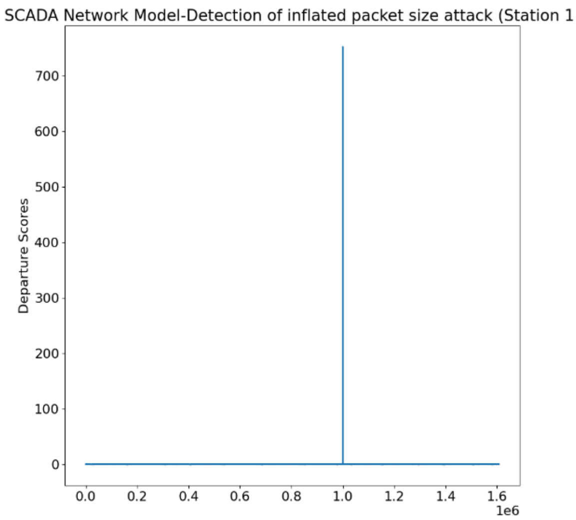


Figure 2.4. PASAD's departure scores.

2.4.3 Operation-Level Experiments

The purpose of this type of experiment is to explore the possibility of monitoring the process health by analysing power-related signals at substation level to detect cyberattacks that may evade detection at the communication level.

After identifying and extracting values for the different signals and training AI models on some selected signals, we have conducted two different attacks on signal data: the “Frozen-value” attack and the “Mean-shift” attack. For the Frozen-value attack, we have frozen the values of a specific range of consecutive packets. The Mean-shift attack was applied by taking the weighted moving average (WMA) plus one standard deviation of a specific range of consecutive packets based on the length of the signals. The Mean-Shift attack is stealthier than the Frozen-Value attack because it follows the distribution of the signal.

2.4.3.1 Frozen-Value Attack

In station 1, we have extracted time series corresponding to four different signals, namely, GRP4 Current B Level 1, Temperature T1, Power Factor T1-400, and Power Reactive T1-400. In addition, by manipulating the sensor values corresponding to the four different signals to a constant value, chosen to be the mean of the signals, we have ended up with four modified test time series. Figure 2.5 shows the four different signals and their corresponding manipulated signals, where the manipulated parts are highlighted in orange.

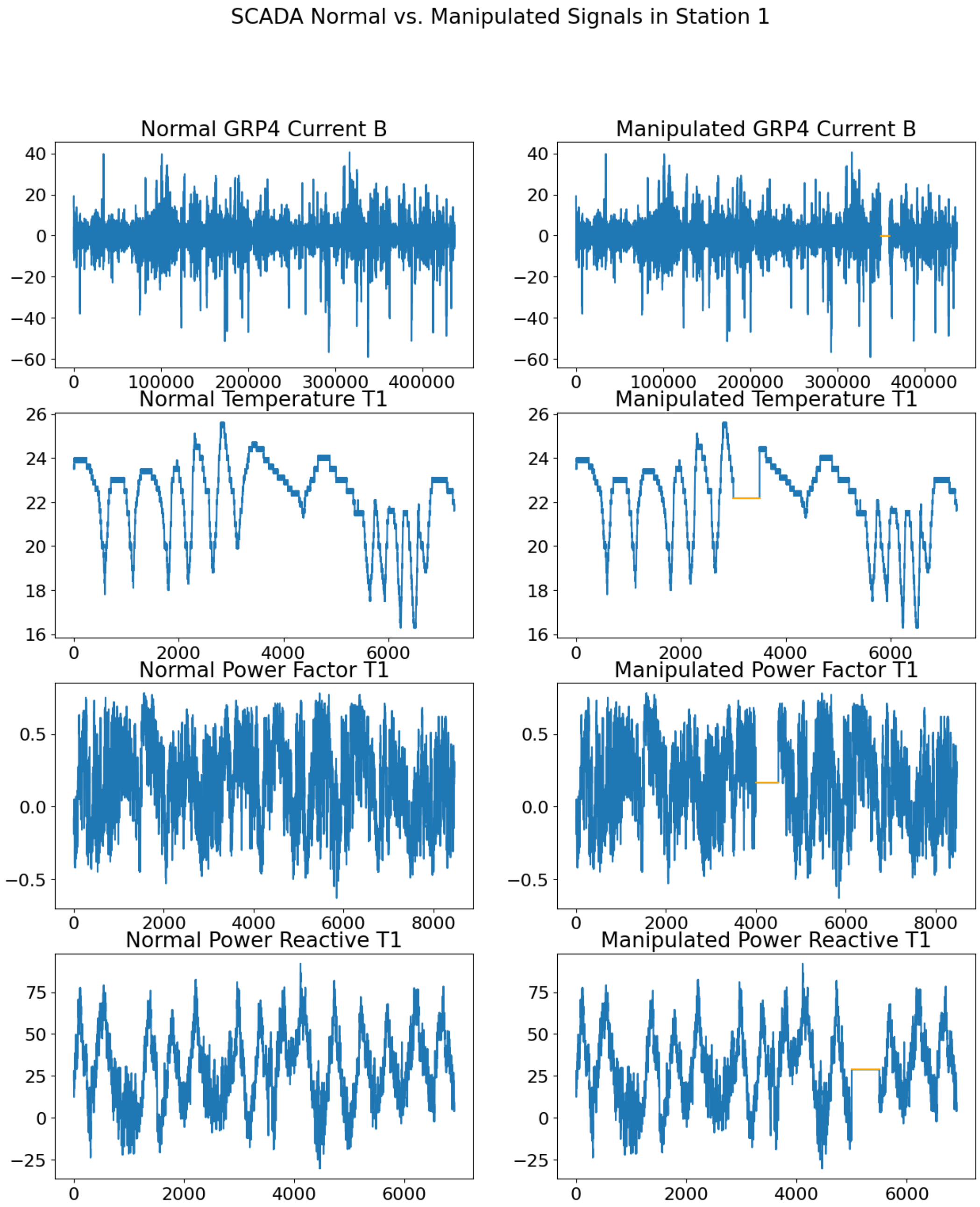


Figure 2.5. SCADA normal versus manipulated signals of station 1 (Frozen-value attack).

To train and test PASAD on these signals, we have extracted 15 rolling-window statistical features from these signals such as mean, standard deviation, median, min, and max. PASAD has trained on normal rolling-window statistical features, which represent the normal signal, and tested on manipulated rolling-window statistical features, which represent the manipulated signal.

The results show that PASAD could detect the manipulation of these signals. Figure 2.6 — Figure 2.13 show departure scores and detection in 3D representation of the PASAD models trained on GRP4 Current B Level 1, Temperature T1, Power Factor T1-400, and Power Reactive T1-400, respectively. As shown in these graphs, the departure scores for the different signals surpass the threshold during the manipulation period and then return to the normal level.

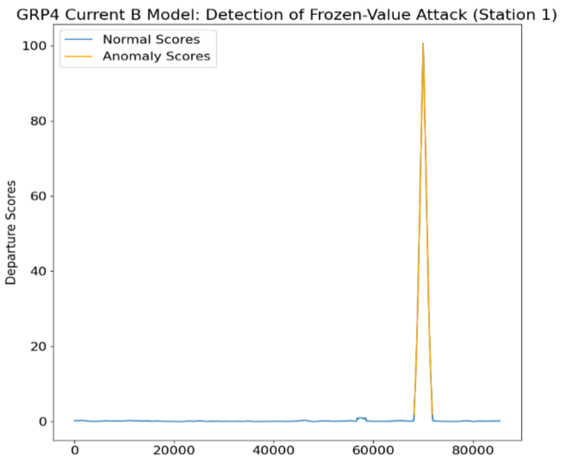


Figure 2.6: Departure scores of GRP4 Current B model in station 1.

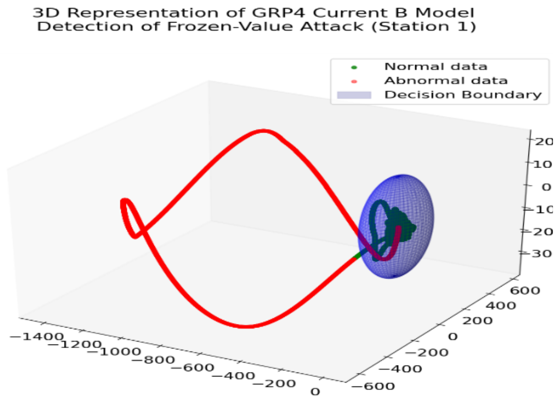


Figure 2.7: GRP4 Current B model detection of the Frozen-value attack in station 1 in 3D.

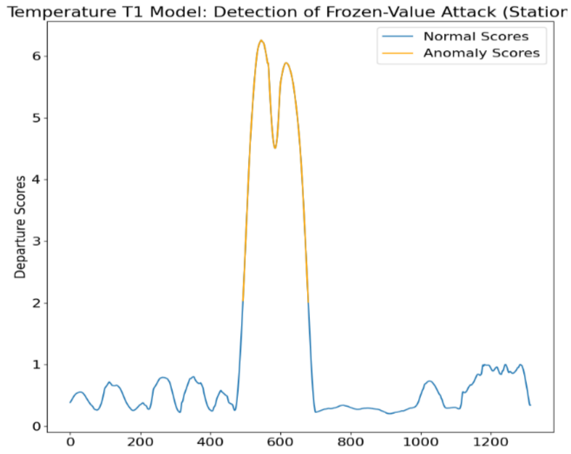


Figure 2.8: Departure scores of Temperature T1 model in station 1.

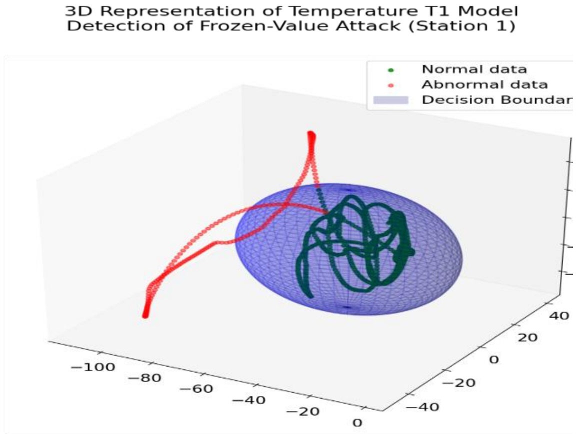


Figure 2.9: Temperature T1 model detection of the Frozen-value attack in station 1 in 3D.

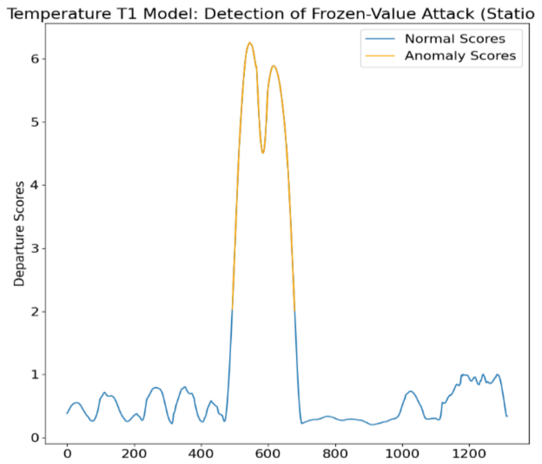


Figure 2.10: Departure scores of the Power Factor T1-400 model in station 1.

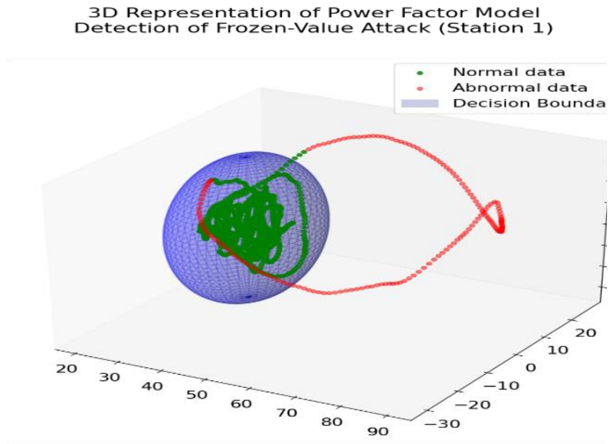


Figure 2.11: Power Factor T1-400 model detection of the Frozen-value attack in station 1 in 3D.

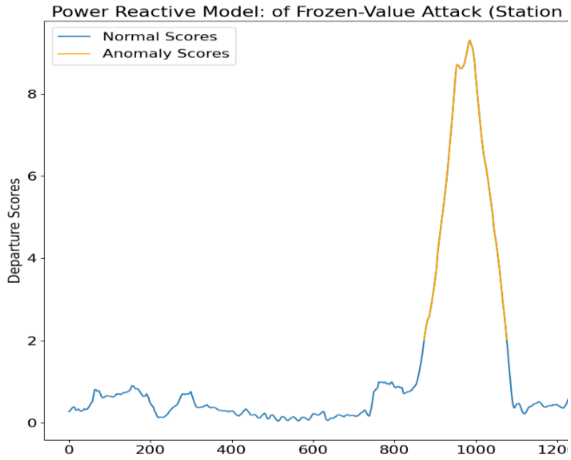


Figure 2.12: Depa[illegible]s of the Power Reactive T1-400 model in station 1.

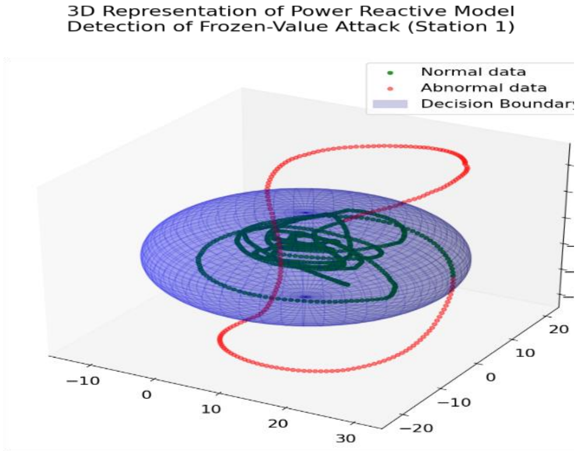


Figure 2.13: Power Reactive T1-400 detection of the Frozen-value attack in station 1 in 3D.

2.4.3.2 Mean-Shift Attack

For the mean-shift attack, we have manipulated the consecutive values of the signals GRP4 Current B Level 1, Temperature T1, Power Factor T1-400, and Power Reactive T1-400 in station 1 by changing the values to the WMA plus one standard deviation of each signal. Figure 2.14 shows the normal signals of GRP4 Current B and Power Factor (on the left), and the manipulated signals of GRP4 Current B and Power Factor (on the right). The manipulation of the consecutive values is highlighted in orange for each signal.

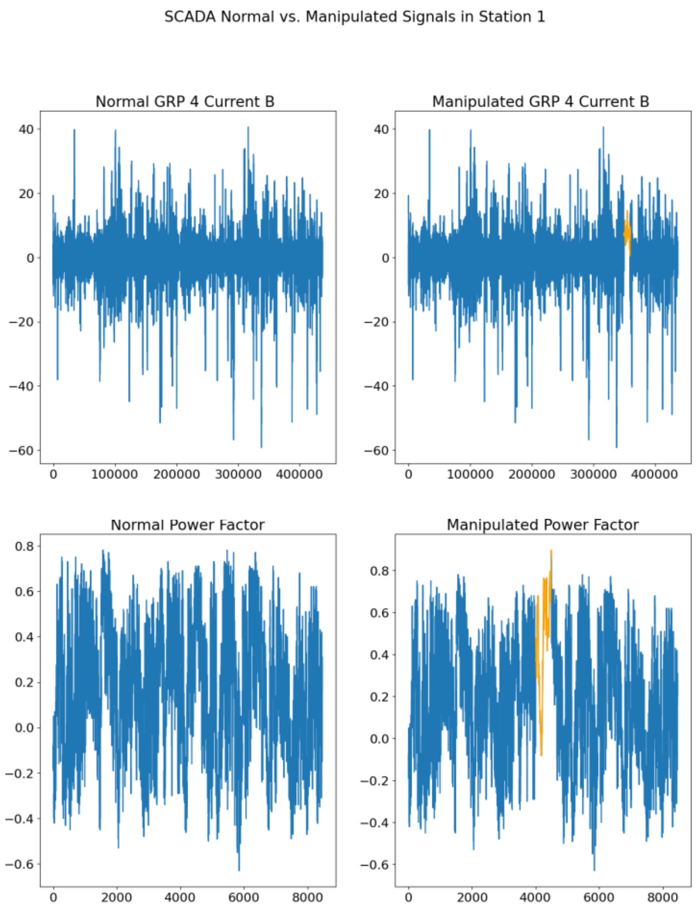


Figure 2.14. Normal GRP4 and Power Factor versus manipulated GRP4 and Power Factor in station 1 (Mean-shift attack).

PASAD was trained on the normal signals and tested on the manipulated signals. PASAD managed to detect the Mean-shift attack in all signals. Figure 2.15 – Figure 2.18 show that PASAD models successfully detected the manipulation.

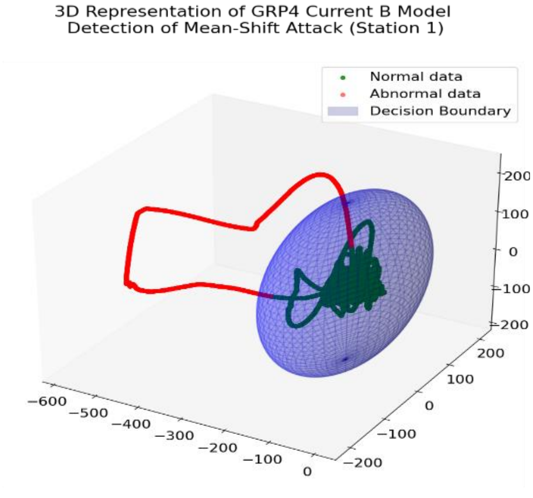


Figure 2.15. GRP4 Current B detection of Mean-shift attack in station 1 in 3D.

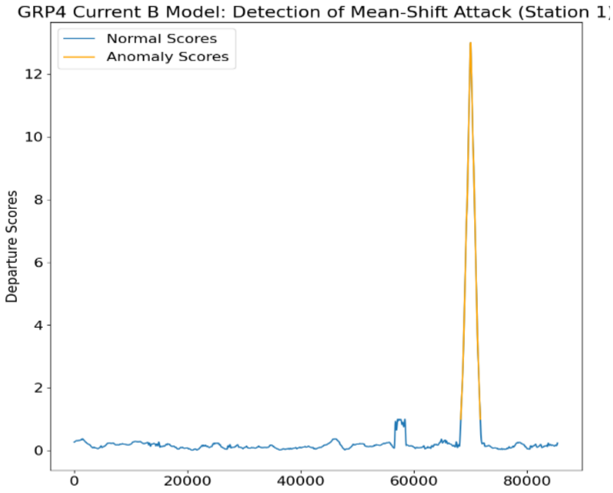


Figure 2.16. PASAD's departure scores of the GRP4 Current B model of Mean-shift attack in station 1

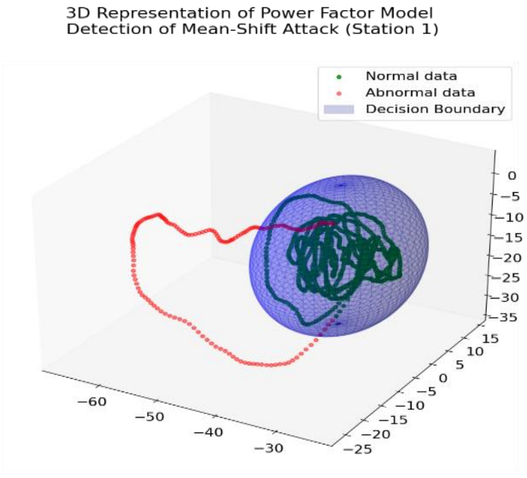


Figure 2.17. Power Factor detection of Mean-shift attack in station 1 in 3D.

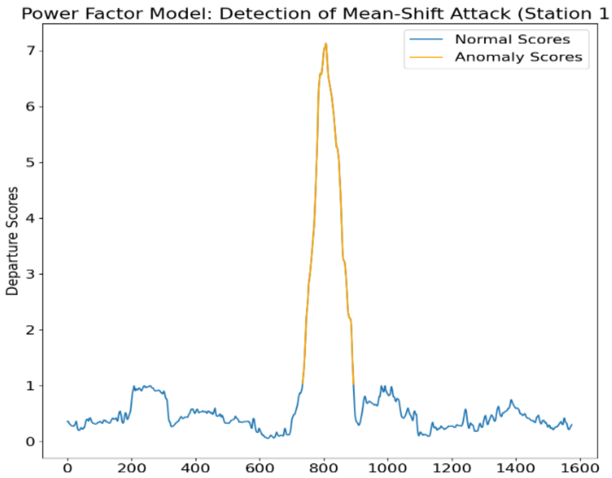


Figure 2.18. PASAD's departure scores of the Power Factor model of Mean-shift attack in station 1.

2.5 Future Perspectives

We have explored an anomaly detection approach for both internet-level and operation-level data streams generated by secondary substations in a smart electrical grid. However, our investigation thus far has focused on a localized anomaly detection scheme, where each substation identifies anomalies solely within its own data. These detected anomalies are not contextualized within a broader framework to enable global data interpretation.

Our next objective is to conduct a comprehensive evaluation of our method using synthetic data (see Section 4). Following this, we aim to design and implement a robust framework capable of gathering critical information from individual substations across the network. This framework will aggregate the collected data, enabling a holistic view of the electrical grid's performance and status. Additionally, we will integrate advanced explainability features into the framework to provide grid operators with actionable insights. These insights will help operators understand the underlying causes of network behaviours and facilitate informed decision-making to enhance grid reliability and efficiency.

2.6 References

[1] N. Q. T. a. J. R. V. Jeffrey, “A hybrid methodology for anomaly detection in Cyber–Physical Systems,” *Neurocomputing,* vol. 568, no. 127068, 2024.

[2] W. M. I. a. M. A. Aoudi, “Truth will out: Departure-based process-level detection of stealthy attacks on control systems,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[3] W. a. M. A. Aoudi, “A scalable specification-agnostic multi-sensor anomaly detection system for IIoT environments,” *International journal of critical infrastructure protection,* vol. 30, no. 100377, 2020.

[4] M. W. A. R. G. R. K. a. A. L. Almgren, “The nuts and bolts of deploying process-level ids in industrial control systems,” in *In Proceedings of the 4th Annual Industrial Control System Security Workshop*, 2018.

[5] W. a. M. A. Aoudi, “A framework for determining robust context-aware attack-detection thresholds for cyber-physical systems,” in *Proceedings of the 2021 Australasian Computer Science Week Multiconference*, 2021.

[6] M. S. W. A. R. L. O. M. A. a. H.-P. S. Kemal, “Model-free detection of cyberattacks on voltage control in distribution grids,” in *European Dependable Computing Conference (EDCC)*, 2019.

[7] N. Q. T. a. J. R. V. Jeffrey, “A hybrid methodology for anomaly detection in Cyber–Physical Systems,” *Neurocomputing,* vol. 568, no. 127068, 2024.

3 Llama analysis in detecting IoT-targeting cyberattacks

3.1 Introduction

The number of IoT devices has rapidly increased over the years, and this growth is not stopping. In 2023, the number of IoT devices worldwide was around 15 billion, and the number is forecasted to be more than 29 billion in the year 2030 [1]. The amount of data going through a network can be massive. However, one of the most significant issues regarding IoT network data is its heterogeneity. An IoT network may have thousands of devices that utilize various protocols. Building a monitoring system to detect malicious activities for IoT networks may require a lot of professional effort. Generally, in the literature, intrusion detection systems are tested by training them on only one or a few datasets. Even if a trained ML model works well with one dataset, it may not be suitable for another or real life. We wanted to avoid such an approach in this report.

Many works have utilized large language models (LLMs) in cybersecurity-related tasks with various success, such as writing malicious code [2, 3, 4], being sparring partners for pen-testers [5], identifying complex vulnerabilities [6] and imitating to be a honeypot [7, 8]. These examples illustrate LLMs' ability to understand cyber-attacks. Considering the IoT network data, one of the most interesting abilities of LLMs is its ability to understand text data and the context of data. These findings led us to experiment with how LLMs can enhance cybersecurity in IoT networks in the future.

Ideally, a well-performed LLM could be installed on an edge router to monitor traffic data. If the LLM detects an attack early enough, it can warn IoT devices about what kind of malicious traffic is coming. Then IoT devices can prepare for the attack. It may also be possible to have a conversation between IoT devices and an LLM if the IoT devices have sufficient capability, such as an internet connection. The IoT device could send suspicious packets to the LLM and ask it to analyze them and deliver possible IoAs to neighbouring IoT devices. The essential first step is to explore the strengths and weaknesses of recent LLMs in detecting cyberattacks from IoT traffic data. This information is needed for building real-life solutions and planning future research steps.

In this study, we analysed how capable Llama 3.1 is of detecting IoT-targeting cyberattacks of several frequently occurring types in IoT network traffic data. Our goal was to find out what kind of signals of malicious activities LLMs can detect, how well they can process the meaning of several simultaneous signals, and what the current limitations of LLMs are. Because one of the motivators for this study was the complexity of IoT networks, we experimented with LLMs as a generalized model. Thus, we did not preprocess our data or fine-tune our model.

This report is organized as follows. Section 3.2 (Method) presents our research process, prompt engineering, used data, and model selection. Section 3.3 (Results) provides our results and findings including both strengths and weaknesses of the used model. In Section 3.4 (Discussion), we discuss our results and outline future research directions. Finally, Section 3.6 (Conclusion) concludes this report.

3.2 Method

In our study, we experimented with how well LLMs can detect cyberattacks of popular types from IoT network traffic data. The heterogeneity of IoT traffic data is one of the most significant issues related to IoT network data. Thus, we wanted to test whether LLMs are capable of handling heterogeneous data without fine-tuning. Fine-tuning typically increases the performance of a model for a specific dataset, but it also requires work from professionals. We decided to leave the study of the effect of fine-tuning to later research and experiment with a model whose capabilities we have not affected.

We also decided not to preprocess our data. The main reason behind this decision was again the heterogeneity of IoT data. We wanted to experiment with how well LLMs understand heterogeneous data and what kind of preprocessing may be essential to achieve good results. Usually, data professionals select the features, clean the data, and otherwise preprocess it before training ML models. This is an essential process when using traditional ML models or DL models. However, this process causes a significant amount of work for professionals. If LLMs were able to understand raw data and detect cyber-attacks from it, it would be a waste of time to spend time for preprocessing the data. In this work, we studied if this kind of approach is possible.

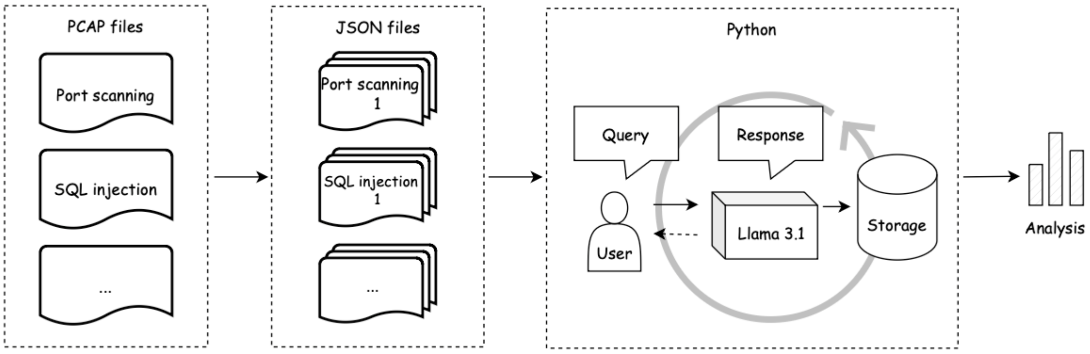


Figure 3.1: Research process

Figure 3.1 outlines the steps of our research process. Our dataset contains data from 12 different attack types in PCAP format. First, we selected three single packets from each attack type, as described in Section 3.2.3. Then, we converted the selected packets into JSON format with all the values Wireshark adds to the data by default. Next, we prompted LLM with each of the packets 12 times. Each of these 12 prompts asked LLM if the packets were part of a different cyber-attack. In total, we prompted the LLM 468 times, each time with a different prompt. Finally, we analyzed the results.

Our analysis included two phases. The first part of the analysis was to find out how often LLM detects a packet as an attack. The second part contains a closer analysis where we had five statements with which we tried to find out what kind of things LLM recognizes and what its limitations are. Our statements were the following:

TF: True finding – the findings of LLM were from the provided data, e.g., correct IP addresses and port numbers. In Section 3.2.4, the findings consist of the seven items mentioned by the LLM from the data.

FF: False findings – the LLM hallucinated the findings so much that it affected the quality of the response, e.g., additional transport layers or wrong MAC addresses.

AV: Attack vector – LLM identified signals of potential malicious activities (or a significant part of them) from the data and mentioned that the findings were suspicious. LLMs do not need to specify the exact attack type but should provide enough suspicious findings from the data for a user to deduce the attack vector.

CR: Correct reasoning – LLM drew correct conclusions based on its findings. It did not matter if the findings were hallucinated or not, nor were the answers correct. Here, we considered whether the conclusions drawn from the findings were correct.

FR: False reasoning – LLM's conclusions based on the findings were wrong.

During the analysis, we focused on the findings and reasoning that were important for the conclusions, disregarding minor mistakes. When identifying malicious activities, we decided that the answer should be sufficiently useful for a user. This means that one minor suspicious finding (e.g., only an unusual port number) is not enough. On the other hand, LLMs do not need to find all the suspicious things that humans can find. Additionally, one important finding (e.g., an SQL query) can also be enough. The purpose behind evaluating LLMs' ability to identify malicious activities is to ensure that the answer is useful for a user and mentions that the findings are potentially malicious. An example of a prompt and LLM's response is in Section 3.2.4.

3.2.1 Prompt engineering

> **{dataset}**
> **Here is an IIoT network capture file, specifically a JSON-formatted Wireshark capture. The file contains a single packet, and your job is to analyze this data.**
>
> **Can you identify this data as {attack type}? Include in your answer the probability of occurrence for attack and typical traffic.**
>
> **Explain this data to me in at least 100 words. Then solve for the answer. Go through the packet carefully step-by-step.**

Above is a prompt structure we developed to utilize in every prompt. We tried multiple prompts and chose one that seemed to be the best. In the beginning, there was a data capture to be analyzed, and the data always contained a single network packet. A more detailed description of the data can be found in Section 3.2.3. After the dataset, we briefly described the context of the data to the LLM. Next, we prompted the actual task for the LLM. Each packet had been prompted 12 times with different attack types, namely ARP spoofing, backdoor attack, cross-site scripting (XSS), DDoS, DNS spoofing, OS Fingerprinting, password cracking, port scanning, ransomware attack, SQL injection, uploading attack, and vulnerability scanning. Each prompt was individual, and the model did not see previous prompts or responses.

At the end of the prompt, we added two prompt engineering methods to improve the performance of LLMs. Yugeswardeenoo et al. introduced the Question-Answer prompting (QAP) method in their paper and its original form is "Explain this problem to me in at least n words. Then solve for the answer". Yugeswardeenoo et al. utilized their method in arithmetic and commonsense problems where the problem is described at the beginning of the prompt. [9] Their problem description corresponds to the data in our case, so we changed the word problem to the word data in our prompt. We came to this solution because LLMs are sensitive to word choices, as also mentioned by Yugeswardeenoo et al. [9]. For us, this also provides the possibility to check whether the LLM has understood the data correctly. Wei et al. prompted LLMs to think step-by-step in their Chain-of-Thought method [10]. We also adopted this method for our study.

3.2.2 Lama 3.1

This study considered IoT networks whose data privacy can be an important aspect. Additionally, IoT devices may have only a very narrowband internet connection, but an attack detection tool should still work. Thus, we decided to utilize local large language models (LLMs) in this study. LLMs are a fast-evolving technology that guided us to choose the most recent models for the study. We selected Llama as a model to be tested because Llama models can be run locally, they have succeeded in various benchmark tests, and the most recent models have a relatively long context length, 128K [11]. At the starting point of this study, the most recent version of Llama was Llama 3.1. We selected that version with 70 billion parameters, Llama 3.1:70b, in the hope of a balance between better results and computational requirements.

We wrote our code using Python programming language and utilized the Ollama framework [12] to run LLMs locally on our computer. The used computer has an Apple M3 Max chip with 128 GB of memory.

3.2.3 Data

To achieve our goals, our requirements for a dataset were that it should be cyber-related IoT network traffic data, labelled in terms of attack type, and heterogeneous. Edge-IIoTset contains labelled network data generated from an IIoT network including more than ten devices and using various protocols. The network has been attacked by 14 different attack types, namely port scanning, vulnerability scanning attack, four different types of DDoS attacks, OS Fingerprinting, DNS spoofing attack, ARP spoofing attack, Cross-site scripting (XSS) attack, SQL injection, Uploading attack, Backdoor attack, Password cracking attack, and ransomware attack. [13] The data is shared in CSV and PCAP formats.

We utilized Edge-IIoTset's PCAP files and decided to analyse individual packets because they are the basic unit of network traffic. We are aware that some attacks are challenging or even impossible to detect from a single packet. On the other hand, we need to know how well LLMs can analyse single packets in order to compare performance with multiple packets. This decision also allowed us to make a deeper analysis of Llama 3.1's behavior because individual packets are easier to analyse manually than larger datasets. We selected three packets from each attack type for our usage, except in the case of DDoS attacks, we selected three packets in total. The selected ones were one of each TCP SYN Flood attacks, UDP flood DDoS attacks, and HTTP flood DDoS attacks. We selected the packets randomly and then saw if the packets contained clues about the attack. If not, then we tried again. ARP spoofing attacks were challenging to detect from only one packet, so these packets did not contain clear clues about the attack. Between the packets, the difficulty of detecting an attack varies a lot. In addition, we selected three packets of normal traffic, one from each of following PCAP files: Distance, Modbus, and Temperature and Humidity. After selection, we converted the raw PCAP captures to JSON format and did not preprocess the data otherwise. In total, we used 39 individual packets in our analysis.

3.2.4 Example prompt and response

In this section, we provide an example of one prompt (input) and Llama 3.1's response (output). The attack type in the data was password cracking. The probability of attack was usually provided as a numerical value in Llama 3.1's responses, but in this example, it was in low-medium-high scale. However, in this example, it can be seen well how Llama 3.1 listed its findings from data and then made brief conclusions based on them. Finally, Llama 3.1 justified its estimation of the likelihood of an attack and offered overall conclusions. This kind of structure were usual for Llama 3.1's responses.

Input:

[{'_index': 'packets-2021-12-04', '_type': 'doc', '_score': None, '_source': {'layers': {'frame': {'frame.encap_type': '1', 'frame.time': 'Dec 4, 2021 19:30:22.830794000 EET', 'frame.time_utc': 'Dec 4, 2021 17:30:22.830794000 UTC', 'frame.time_epoch': '1638639022.830794000', 'frame.offset_shift': '0.000000000', 'frame.time_delta': '0.000133000', 'frame.time_delta_displayed': '0.007166000', 'frame.time_relative': '60286.869222000', 'frame.number': '449208', 'frame.len': '327', 'frame.cap_len': '327', 'frame.marked': '0', 'frame.ignored': '0', 'frame.protocols': 'eth:ethertype:ip:tcp:http:urlencoded-form', 'frame.coloring_rule.name': 'HTTP', 'frame.coloring_rule.string': 'http || tcp.port == 80 || http2'}, 'eth': {'eth.dst': 'dc:a6:32:fb:69:b5', 'eth.dst_tree': {'eth.dst_resolved': 'RaspberryPiT_fb:69:b5', 'eth.dst.oui': '14460466', 'eth.dst.oui_resolved': 'Raspberry Pi Trading Ltd', 'eth.dst.lg': '0', 'eth.dst.ig': '0', 'eth.addr': 'dc:a6:32:fb:69:b5', 'eth.addr_resolved': 'RaspberryPiT_fb:69:b5', 'eth.addr.oui': '14460466', 'eth.addr.oui_resolved': 'Raspberry Pi Trading Ltd', 'eth.lg': '0', 'eth.ig': '0'}, 'eth.src': 'd8:f2:ca:8e:17:69', 'eth.src_tree': {'eth.src_resolved': 'Intel_8e:17:69', 'eth.src.oui': '14217930', 'eth.src.oui_resolved': 'Intel Corporate', 'eth.src.lg': '0', 'eth.src.ig': '0', 'eth.addr': 'd8:f2:ca:8e:17:69', 'eth.addr_resolved': 'Intel_8e:17:69', 'eth.addr.oui': '14217930', 'eth.addr.oui_resolved': 'Intel Corporate', 'eth.lg': '0', 'eth.ig': '0'}, 'eth.type': '0x0800', 'eth.stream': '0'}, 'ip': {'ip.version': '4', 'ip.hdr_len': '20', 'ip.dsfield': '0x00', 'ip.dsfield_tree': {'ip.dsfield.dscp': '0', 'ip.dsfield.ecn': '0'}, 'ip.len': '313', 'ip.id': '0xd713', 'ip.flags': '0x02', 'ip.flags_tree': {'ip.flags.rb': '0', 'ip.flags.df': '1', 'ip.flags.mf': '0'}, 'ip.frag_offset': '0', 'ip.ttl': '64', 'ip.proto': '6', 'ip.checksum': '0xe030', 'ip.checksum.status': '2', 'ip.src': '192.168.0.170', 'ip.addr': '192.168.0.128', 'ip.src_host': '192.168.0.170', 'ip.host': '192.168.0.128', 'ip.dst': '192.168.0.128', 'ip.dst_host': '192.168.0.128', 'ip.stream': '0'}, 'tcp': {'tcp.srcport': '54974', 'tcp.dstport': '80', 'tcp.port': '80', 'tcp.stream': '40226', 'tcp.stream.pnum': '3', 'tcp.completeness': '31', 'tcp.completeness_tree': {'tcp.completeness.rst': '0', 'tcp.completeness.fin': '1', 'tcp.completeness.data': '1', 'tcp.completeness.ack': '1', 'tcp.completeness.syn-ack': '1', 'tcp.completeness.syn': '1', 'tcp.completeness.str': '·FDASS'}, 'tcp.len': '261', 'tcp.seq': '1', 'tcp.seq_raw': '3520572880', 'tcp.nxtseq': '262', 'tcp.ack': '1', 'tcp.ack_raw': '1550123349', 'tcp.hdr_len': '32', 'tcp.flags': '0x0018', 'tcp.flags_tree': {'tcp.flags.res': '0', 'tcp.flags.ae': '0', 'tcp.flags.cwr': '0', 'tcp.flags.ece': '0', 'tcp.flags.urg': '0', 'tcp.flags.ack': '1', 'tcp.flags.push': '1', 'tcp.flags.reset': '0', 'tcp.flags.syn': '0', 'tcp.flags.fin': '0', 'tcp.flags.str': '·······AP···'}, 'tcp.window_size_value': '502', 'tcp.window_size': '64256', 'tcp.window_size_scalefactor': '128', 'tcp.checksum': '0x360d', 'tcp.checksum.status': '2', 'tcp.urgent_pointer': '0', 'tcp.options': '01:01:08:0a:e7:b3:5a:00:7b:28:4a:6d', 'tcp.options_tree': {'tcp.options.nop': '01', 'tcp.options.nop_tree': {'tcp.option_kind': '1'}, 'tcp.options.timestamp': '08:0a:e7:b3:5a:00:7b:28:4a:6d', 'tcp.options.timestamp_tree': {'tcp.option_kind': '8', 'tcp.option_len': '10', 'tcp.options.timestamp.tsval': '3887290880', 'tcp.options.timestamp.tsecr': '2066238061'}}, 'Timestamps': {'tcp.time_relative': '0.007223000', 'tcp.time_delta': '0.007166000'}, 'tcp.analysis': {'tcp.analysis.acks_frame': '449203', 'tcp.analysis.ack_rtt': '0.007166000', 'tcp.analysis.initial_rtt':

'0.007223000', 'tcp.analysis.bytes_in_flight': '261', 'tcp.analysis.push_bytes_sent': '261'}, 'tcp.payload':
'50:4f:53:54:20:2f:44:56:57:41:2f:6c:6f:67:69:6e:2e:70:68:70:20:48:54:54:50:2f:31:2e:30:0d:0a:48:6f:73:74:3a:20:31:39:32:2e:31:36:38:2e:30:2e:31:32:38:0d:0a:55:73:65:72:2d:41:67:65:6e:74:3a:20:4d:6f:7a:69:6c:6c:61:2f:35:2e:30:20:28:48:79:64:72:61:29:0d:0a:43:6f:6e:74:65:6e:74:2d:4c:65:6e:67:74:68:3a:20:33:38:0d:0a:43:6f:6e:74:65:6e:74:2d:54:79:70:65:3a:20:61:70:70:6c:69:63:61:74:69:6f:6e:2f:78:2d:77:77:77:2d:66:6f:72:6d:2d:75:72:6c:65:6e:63:6f:64:65:64:0d:0a:43:6f:6f:6b:69:65:3a:20:50:48:50:53:45:53:53:49:44:3d:33:71:35:6c:64:6f:73:6c:36:35:66:61:74:6f:31:33:30:67:32:61:37:71:66:6a:6c:70:3b:20:73:65:63:75:72:69:74:79:3d:69:6d:70:6f:73:73:69:62:6c:65:0d:0a:0d:0a:75:73:65:72:6e:61:6d:65:3d:61:64:6d:69:6e:26:70:61:73:73:77:6f:72:64:3d:30:30:26:4c:6f:67:69:6e:3d:4c:6f:67:69:6e'}, 'http': {'POST /DVWA/login.php HTTP/1.0\\r\\n': {'http.request.method': 'POST', 'http.request.uri': '/DVWA/login.php', 'http.request.version': 'HTTP/1.0'}, 'http.host': '192.168.0.128', 'http.request.line': 'Cookie: PHPSESSID=3q5ldosl65fato130g2a7qfjlp; security=impossible\r\n', 'http.user_agent': 'Mozilla/5.0 (Hydra)', 'http.content_length_header': '38', 'http.content_length_header_tree': {'http.content_length': '38'}, 'http.content_type': 'application/x-www-form-urlencoded', 'http.cookie': 'PHPSESSID=3q5ldosl65fato130g2a7qfjlp; security=impossible', 'http.cookie_tree': {'http.cookie_pair': 'security=impossible'}, '\\r\\n': '', 'http.request': '1', 'http.response_in': '449210', 'http.request.full_uri': 'http://192.168.0.128/DVWA/login.php', 'http.file_data': '75:73:65:72:6e:61:6d:65:3d:61:64:6d:69:6e:26:70:61:73:73:77:6f:72:64:3d:30:30:26:4c:6f:67:69:6e:3d:4c:6f:67:69:6e'}, 'urlencoded-form': {'Form item: "username" = "admin"': {'urlencoded-form.key': 'username', 'urlencoded-form.value': 'admin'}, 'Form item: "password" = "00"': {'urlencoded-form.key': 'password', 'urlencoded-form.value': '00'}, 'Form item: "Login" = "Login"': {'urlencoded-form.key': 'Login', 'urlencoded-form.value': 'Login'}}}}}]

Here is an IIoT network capture file, specifically a JSON-formatted Wireshark capture. The file contains a single packet, and your job is to analyze this data.

Can you identify this data as XSS? Include in your answer the probability of occurrence for attack and typical traffic.

Explain this data to me in at least 100 words. Then solve for the answer. Go through the packet carefully step-by-step.

Output:

The provided JSON-formatted Wireshark capture contains a single packet with various layers of information. Let's break it down step by step:

1. The first section describes the frame, including the protocol (HTTP), source and destination IP addresses, and other low-level details.

2. The next section is the HTTP request layer:

- The `http.request.method` is set to `POST`, indicating that the client is sending data to the server.

- The `http.request.uri` is `/DVWA/login.php`, which suggests that this packet is part of a login process on a Damn Vulnerable Web Application (DVWA) instance. DVWA is a deliberately vulnerable web application used for training and testing purposes.

3. The `http.host` field contains the domain name or IP address of the server being accessed, which in this case is `192.168.0.128`.

4. The `http.user_agent` field indicates that the client claims to be running Mozilla/5.0 (Hydra). However, Hydra is a known vulnerability scanner and exploit tool, not a typical web browser.

5. The `http.content_type` field shows that the request body contains data in the format of application/x-www-form-urlencoded.

6. Looking at the request body, we find the following form data:

- `username=admin`

- `password=00`

- `Login=Login`

7. Lastly, let's examine the HTTP cookie header: `PHPSESSID=3q5ldosl65fato130g2a7qfjlp; security=impossible`. The presence of a PHP session ID suggests that this packet is part of an ongoing interaction with the server.

Now, considering these details as a whole:

*- **Probability of occurrence for attack:** High*

The combination of a POST request to a login page on a vulnerable web application (DVWA), along with form data containing a username and password, raises suspicions. The fact that Hydra is being used instead of a typical browser further increases the likelihood of this packet being part of an attack.

*- **Typical traffic:** Low*

Given the specific details of this packet, such as the use of DVWA, Hydra, and the explicit mention of "security=impossible" in the cookie header, it's unlikely that this represents normal traffic. This suggests a deliberate attempt to exploit vulnerabilities rather than routine web activity.

Is this data indicative of XSS (Cross-Site Scripting)?

While this packet does contain signs of malicious activity, there isn't explicit evidence of Cross-Site Scripting (XSS). The presence of form data and the context suggest an attempt to login or interact with the application in a potentially unauthorized manner. However, without seeing any attempts to inject scripts into the request or manipulate the server's response for client-side script execution, we cannot conclusively say this packet is indicative of XSS.

Conclusion:

This packet shows signs of malicious activity aimed at exploiting vulnerabilities in a web application (DVWA). The high probability of occurrence for an attack and the low likelihood of representing typical traffic are clear indicators that this data warrants closer inspection. However, it does not explicitly indicate Cross-Site Scripting (XSS) without further evidence or context.

3.3 Results

3.3.1 Quantitative results

Figure 3.2 outlines the results of our Llama analysis. Every prompt was analysed based on the answer (whether the packet was identified as an attack or not) and five statements which are described in the previous section. The prompts where the requested attack type matches the data attack type are grayed out. Also, the corresponding calculated values are marked with a light gray background. The ARP spoofing and normal traffic packets have a '-'mark in the attack vector (AV) column because there were no clear signals about attacks.

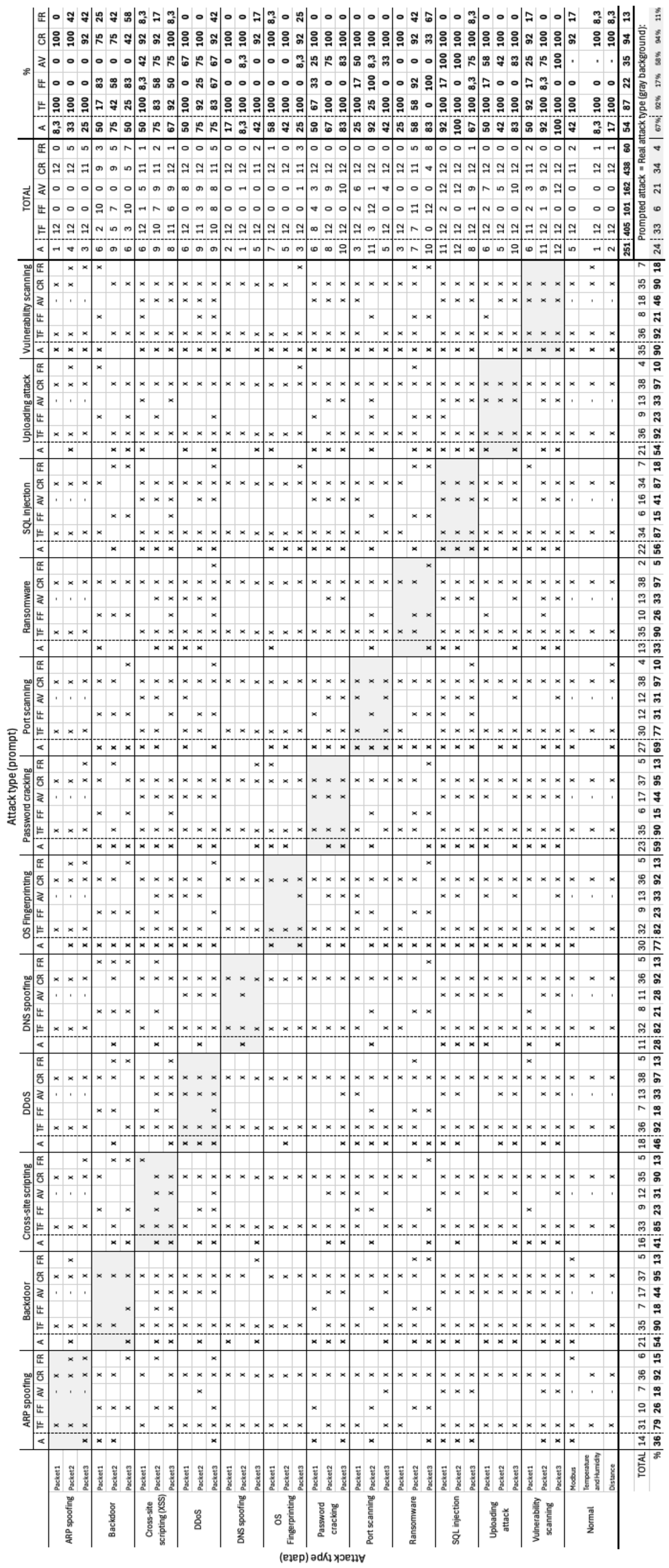
Llama 3.1 labelled our packet captures as a cyber-attack (A in Figure 3.2) in 54% of the cases within all the packets and all the prompts, but it found the signals of malicious activities (AV in Figure 3.2) in only 35% of the cases. This means that Llama 3.1 relatively often said that the packet was an attack but did not give clear evidence of that. Here, it should be remembered that we required the model to identify malicious signals enough that the answer would be useful to professionals. In addition, it had to be mentioned that these signals were suspicious (see Section Model). This choice made an impact on the success rate in identifying signals of malicious activities.

Llama performed best with SQL injection. Within the three SQL packets and all the experiments, it concluded that the packet was an attack (A in Figure 3.2) in 86% of the cases. It also identified malicious signals (AV in Figure 3.2) almost every time, in 92% of the cases. This means that Llama successfully mentioned suspicious activities even if it was not sure about the attack. It detected SQL queries well from the data and could specify that the attack type was an SQL injection, even though it had been requested about other attack types. The detection rate for vulnerability scanning attacks (A in Figure 3.2) was also high, at 81%. Unfortunately, it could identify the suspicious signals (AV in Figure 3.2) worse than in the case of SQL injection, achieving a rate of 67%. Moreover, Llama 3.1 claimed the packets as attacks in 90% of the cases when the prompted attack type was a vulnerability scanning attack, even whenever the packet was normal traffic. Thus, the detection of vulnerability scanning cannot be seen as successful as in the case of SQL injection.

The most challenging attack types for Llama3.1 were ransomware attacks, OS fingerprinting, DNS spoofing, backdoor attacks, and ARP spoofing. The poor success in OS fingerprinting, DNS spoofing, and ARP spoofing is explained by the fact that they were challenging to detect from one packet even by humans. An interesting aspect of ransomware and backdoor attacks was that Llama hallucinated lot in these cases. We assume that the reason behind this was encrypted data in the payloads. Llama 3.1 can read ASCII-formatted data and decode it into human-readable format, but in the case of encrypted data, it answered badly.

In general, Llama 3.1's hallucination significantly reduces the quality of the response (FF in Figure 3.2) in 22% of the cases within all the packets and all the experiments. Mostly, this happened when there was an encrypted payload in the packet (ransomware packets 2-3 and backdoor packets 1-3 in Figure 3.2), as mentioned earlier. Llama 3.1 was able to read ASCII-formatted payloads (e.g., DDoS packet 2 in Figure 3.2). In these cases, the hallucination rate (FF in Figure 3.2) was relatively low, even if the text was not strictly human-readable. However, Llama 3.1 hallucinated if the packet contained other unreasonable strings, even if it seemed to humans that the data was only machinery-generated filler. These kinds of issues occurred with a large amount of repeated hexadecimal numbers in payload, such as 58s (DDoS packet 3 in Figure 3.2) or 00s (Port scanning packet 2 in Figure 3.2), and with hexadecimal formatted TCP options (XSS packets 2 and 3 in Figure 3.2). These observations strongly indicate that Llama 3.1 is not able to answer correctly if the prompt contains data it cannot read.

Llama 3.1's reasoning was sensible (CR in Figure 3.2) 94% of the time within all the prompts. In other words, Llama 3.1 can generate a believable answer regardless of whether it is true or not. Thus, the hallucination is challenging to detect if you do not have the original data as a reference point. In any case, LLMs' habit of hallucinating causes more work for professionals because they must double-check all the answers. In addition, it reduces the reliability of the model's answers. Interestingly, we observed that if Llama3.1 hallucinated in its answer, it usually hallucinated also values provided in the data, such as IP addresses and port numbers. Fortunately, these values are easy to check, but unfortunately, the model did not mention these values in every answer.



Attack type (data)	Packet	TOTAL A	TOTAL TF	TOTAL FF	TOTAL AV	TOTAL CR	TOTAL FR	% A	% TF	% FF	% AV	% CR	% FR
ARP spoofing	Packet1	1	12	0	0	12	0	8,3	100	0	0	100	0
	Packet2	4	12	0	0	12	5	33	100	0	0	100	42
	Packet3	3	12	0	0	11	5	25	100	0	0	92	42
Backdoor	Packet1	6	2	10	0	9	3	50	17	83	0	75	25
	Packet2	9	5	7	0	9	5	75	42	58	0	75	42
	Packet3	6	3	10	0	5	7	50	25	83	0	42	58
Cross-site scripting (XSS)	Packet1	6	12	1	5	11	1	50	100	8,3	42	92	8,3
	Packet2	9	10	7	9	11	2	75	83	58	75	92	17
	Packet3	8	11	6	9	12	1	67	92	50	75	100	8,3
DDoS	Packet1	6	12	0	8	12	0	50	100	0	67	100	0
	Packet2	9	11	3	9	12	0	75	92	25	75	100	0
	Packet3	9	10	8	8	11	5	75	83	67	67	92	42
DNS spoofing	Packet1	2	12	0	0	12	0	17	100	0	0	100	0
	Packet2	1	12	0	1	12	0	8,3	100	0	8,3	100	0
	Packet3	5	12	0	0	11	2	42	100	0	0	92	17
OS Fingerprinting	Packet1	7	12	0	0	12	1	58	100	0	0	100	8,3
	Packet2	5	12	0	0	12	0	42	100	0	0	100	0
	Packet3	3	12	0	1	11	3	25	100	0	8,3	92	25
Password cracking	Packet1	6	8	4	3	12	0	50	67	33	25	100	0
	Packet2	8	12	0	9	12	0	67	100	0	75	100	0
	Packet3	10	12	0	10	12	0	83	100	0	83	100	0
Port scanning	Packet1	3	12	2	6	12	0	25	100	17	50	100	0
	Packet2	11	3	12	1	12	0	92	25	100	8,3	100	0
	Packet3	5	12	0	4	12	0	42	100	0	33	100	0
Ransomware	Packet1	3	12	0	0	12	0	25	100	0	0	100	0
	Packet2	7	7	11	0	11	5	58	58	92	0	92	42
	Packet3	10	0	12	0	4	8	83	0	100	0	33	67
SQL injection	Packet1	11	12	2	12	12	0	92	100	17	100	100	0
	Packet2	12	12	0	12	12	0	100	100	0	100	100	0
	Packet3	8	12	1	9	12	1	67	100	8,3	75	100	8,3
Uploading attack	Packet1	6	12	2	7	12	0	50	100	17	58	100	0
	Packet2	5	12	0	5	12	0	42	100	0	42	100	0
	Packet3	10	12	0	10	12	0	83	100	0	83	100	0
Vulnerability scanning	Packet1	6	11	2	3	11	2	50	92	17	25	92	17
	Packet2	11	12	1	9	12	0	92	100	8,3	75	100	0
	Packet3	12	12	0	12	12	0	100	100	0	100	100	0
Normal	Modbus	5	12	0	0	11	2	42	100	0	-	92	17
	Temperature and Humidity	1	12	0	0	12	1	8,3	100	0	-	100	8,3
	Distance	2	12	0	0	12	1	17	100	0	-	100	8,3
		251	405	101	162	438	60	54	87	22	35	94	13
Prompted attack = Real attack type (gray background):		24	33	6	21	34	4	67%	92%	17%	58%	94%	11%

Attack type (prompt)	TOTAL A	TOTAL TF	TOTAL FF	TOTAL AV	TOTAL CR	TOTAL FR	% A	% TF	% FF	% AV	% CR	% FR
ARP spoofing	14	31	10	7	36	6	36	79	26	18	92	15
Backdoor	21	35	7	17	37	5	54	90	18	44	95	13
Cross-site scripting	16	33	9	12	35	5	41	85	23	31	90	13
DDoS	18	36	7	13	38	5	46	92	18	33	97	13
DNS spoofing	11	32	8	11	36	5	28	82	21	28	92	13
OS Fingerprinting	30	32	9	13	36	5	77	82	23	33	92	13
Password cracking	23	35	6	17	37	5	59	90	15	44	95	13
Port scanning	27	30	12	12	38	4	69	77	31	31	97	10
Ransomware	13	35	10	13	38	2	33	90	26	33	97	5
SQL injection	22	34	6	16	34	7	56	87	15	41	87	18
Uploading attack	21	36	9	13	38	4	54	92	23	33	97	10
Vulnerability scanning	35	36	8	18	35	7	90	92	21	46	90	18

Figure 3.2: Each prompt is rated based on attack identification (A) and five assertions: True Findings (TF), False Findings (FF), Attack Vector Identification (AV), Correct Reasoning (CR) and False Reasoning (FR). True statements are indicated with a cross.

The prompts where the requested attack type matched the data attack type had better results than the average of all prompts. In these cases, Llama 3.1 identified the packets as attacks (A in Figure 3.2) 67% of the time. Among all attack types and all prompts, it successfully identified them in only 54% of the cases (56,3% without normal traffic). Respectively, the identification rate of malicious activities (AV in Figure 3.2) rose to 58% from 35% (37,5% without normal traffic). The latter one is interesting because the increase is so large. Thus, it seems that the type of attack given in the prompt affects the analysis of LLM, and Llama 3.1 can detect malicious activities better if it has a prompt with a similar attack type to the actual attack type.

3.3.2 Qualitative results

In this section, we go through examples that show LLM's ability to understand network data and its context. Examples are collected in Table 3.1. We selected examples that described well how Llama 3.1 worked and examples that existed multiple times. The goal of this section is to improve understanding of Llama's strengths and weaknesses as a cyber-attack detector.

Table 3.1: Llama 3.1's Analysis for single packets

Example number	Attack type	Example of Llama's response
1	XSS	The packet contains an HTTP request with the method "TRACE" (line 234). This is not a typical method used in standard web browsing and is often used by attackers to identify vulnerabilities.
2	XSS	Typical traffic for an IIoT network would likely involve machine-to-machine communication, such as sensor data transmission or control commands. This packet appears to be part of a human-initiated HTTP request, which is unusual in an IIoT context
3	DDoS	**Source and Destination**: The source IP address is `78.151.46.12`, and the destination IP address is `192.168.0.128`. This suggests that the traffic is coming from outside the local network.
4	Port scanning	The destination port is `65535`, which is an unusual and high-numbered port.
5	Port scanning	tcp.srcport`: Source port number (`2922`) is not a standard port for any well-known service.
6	OS Fingerprinting	The source port number is 12924, which could indicate that it's a dynamically assigned port, often seen in botnet traffic or other malicious activities.
7	Backdoor	The packet is a TCP segment with a destination port of 80, indicating it's targeting a web server.
8	DDoS	The payload of the packet contains a sequence of `X` (ASCII value 58) characters, which appears to be padding or filler data. This could be an indication of anomalous traffic, as legitimate HTTP requests typically contain meaningful data in the payload.
9	OS Fingerprinting	This contains a large amount of data (48 bytes) with a repeating pattern of incrementing hexadecimal values (`08:09:0a:0b:...`). This is likely a padding or probing packet used to gather information about the target device.
10	Password	The presence of a `<form>` element submitting data to a PHP script suggests that there may be an opportunity for SQL injection.
11	SQL injection	Specifically, the query tries to select 5666 from a subquery that sleeps for 5 seconds (`SLEEP(5)`). This is likely an attempt to exploit a SQL injection vulnerability in the web application. (`/DVWA/vulnerabilities/sqli_blind/?id=7%20AND%20%28SELECT%205666%20FROM%20%28SELECT%28SLEEP%285%29%29%29WkGO%29&Submit=Submitid=57)
12	Vulnerability scanning	The query parameter value contains suspicious characters and a Windows file path (`c:\\boot.ini`), which could indicate an attempt to inject malicious code.
13	Vulnerability scanning	`urlencoded-form`: The URL-encoded form data contains an interesting item: + Key: `<!--#include virtual` + Value: `"/index.jsp"-->"` This appears to be an attempt to inject malicious code using the `#include` directive, which is commonly used in Server-Side Includes (SSI). This could be a signature of a Cross-Site Scripting (XSS) attack.

14	Vulnerability scanning	The `http.user_agent` field contains a user agent string that identifies the client as "Mozilla/5.00 (Nikto/2.1.6)". Nikto is a well-known web application vulnerability scanner, which further supports the conclusion that this traffic is related to vulnerability scanning.
15	Sql injection	that the request is coming from an instance of sqlmap, a popular open-source tool for identifying and exploiting SQL injection vulnerabilities.
16	Sql injection	DVWA (Damn Vulnerable Web Application) is a known vulnerable web application, which suggests that this packet might be part of a vulnerability scan.
17	Password	The `User-Agent` header is set to `Mozilla/5.0 (Hydra)`, which is a known HTTP client that can be used for malicious activities
18	DDoS	This User-Agent string appears to be spoofed or manipulated, as it contains a mix of different browsers and versions: * Opera 9.80 is an outdated version. * Macintosh; Intel Mac OS X 10.7.0 suggests a Mac running Lion (2011). * Edition MacAppStore is not a standard User-Agent component. * AppleWebKit/534.34 is an old WebKit version. * PhantomJS/1.9.0 is a headless browser, which is unusual for regular web traffic. * Safari/534.34 is another outdated Safari version.
19	DDoS	['tcp.payload': '47:45:54:20:2f:20:48:54:54:50:2f:31:2e:31:0d:0a:48:6f:73:74:3a:20:31:39:32:2e:31:36:38:2e:30:2e:31:32:38:0d:0a:55:73:65:72:2d:41:67:65:6e:74:3a:20:4f:70:65:72:61:2f:39:2e:38:30:20:28:4d:61:63:69:6e:74:6f:73:68:3b:20:49:6e:74:65:6c:20:4d:61:63:20:4f:53:20:58:20:31:30:2e:37:2e:30:3b:20:55:3b:20:45:64:69:74:69:6f:6e:20:4d:61:63:41:70:70:53:74:6f:72:65:3b:20:65:6e:29:20:4d:6f:7a:69:6c:6c:61:2f:35:2e:30:20:28:4d:61:63:69:6e:74:6f:73:68:3b:20:49:6e:74:65:6c:20:4d:61:63:20:4f:53:20:58:29:20:41:70:70:6c:65:57:65:62:4b:69:74:2f:35:33:34:2e:33:34:20:28:4b:48:54:4d:4c:2c:6c:69:6b:65:20:47:65:63:6b:6f:29:20:50:68:61:6e:74:6f:6d:4a:53:2f:31:2e:39:2e:30:20:28:64:65:76:65:6c:6f:70:6d:65:6e:74:29:20:53:61:66:61:72:69:2f:35:33:34:2e:33:34:0d:0a:52:65:66:65:72:65:72:3a:20:54:45:53:54:49:4e:47:5f:50:55:52:50:4f:53:45:53:5f:4f:4e:4c:59:0d:0a'] The HTTP request is interesting: * The `Host` header specifies the destination IP address (`192.168.0.118`). * The `User-Agent` header identifies the client software as Opera 9.80 (Macintosh; Intel Mac OS X 10.7; U; Edition MacAppStore; en) with a AppleWebKit-based rendering engine. * The `Referer` header contains a URL that appears to be a test or debugging purpose (`TESTING_PURPOSES_ONLY`).

Based on our results, Llama 3.1 understood how different protocols work, what the function of a protocol is, and what kind of data it typically transfers. The meaning of parameters was usually correctly understood, and the JSON formatted data did not seem to cause difficulties. Llama was even able to consider the context of an IIoT network as we can see from example 2 in Table 3.1. In Table 3.1, example 1 shows how Llama used information about the TRACE in its reasoning and thus, we can see that it understands the difference between different HTTP methods.

The most typical observations from a packet capture were MAC addresses, IP addresses, and port numbers. Llama3.1 has a good, but not perfect, idea about the default ports of different protocols, such as HTTP using port 80 and HTTPS port 443. It also saw high-numbered ports as suspicious ports. Examples 4-7 in Table 3.1 outline this capability of LLMs. In addition, Llama understood the structure of IP addresses, which can be seen in its responses where it deduces whether the network is local or not based on the IP address. It was even able to use this information to estimate the likelihood of a cyber-attack (see example 3 in Table 3.1).

Llama 3.1 can identify human-readable suspicious pieces of data and make conclusions based on them. In addition, Llama3.1 understands ASCII-formatted data if it is not encrypted (example 19). The model was reliable in detecting SQL queries, usually from URLs (see example 11 in Table 3.1), and it multiple times detected suspicious payloads where data seemed maliciously generated. For instance, Llama 3.1 identified if the payload repeats the same numbers (00:00:00:00:00… or 58:58:58:58:58…) or the data contained increasing values (08:09:0a:0b…) as shown in examples 8 and 9 in Table 3.1. Interestingly, Llama 3.1 also concluded that an event is suspicious even if it cannot directly see the malicious data. In Table 3.1, examples 10 and 13 show how Llama identified files as potentially malicious activities. By combining the other evidence of malicious traffic, it concluded that the uploaded file was probably malicious.

Examples 14-17 in Table 3.1 illustrate Llama 3.1's ability to identify used tools as evidence of malicious activities. Not only did Llama recognize tools, but it could also tell the suspicious purpose of the tool or vulnerabilities related to the tool (example 18 in Table 3.1).

As shown above, Llama 3.1 was able to detect different attacks using various hints of malicious activities. However, according to our experiments, we cannot yet trust Llama 3.1 as a cyberattack detector because it does not notice the attack every time and it also has other weaknesses. Next, we go through our findings regarding the issues that need to be solved before using them in real-life solutions.

Some of our experiments were quite challenging. Sometimes, it was difficult even for the researchers to show properly that a single packet contains malicious activities. There might have been only some weak signs about anomalies in the data, such as ransomware should be detected mainly based on large amounts of encrypted data. These experiments provided us an interesting view of LLM's behavior. If LLM was not able to answer the question, it hallucinated additional information to the prompted data. The most significant thing was that LLM did not only add hallucinated values, such as additional transport layers, but it usually also hallucinated existing parameters, such as IP addresses and port numbers. For example, it said that "The packet is an Ethernet II frame with a source MAC address of `a5:54:27:39:4d:8d:59:51:c2:19:95:ee:44:45:31:f4:69:2a:15:60:45:d3:05:c2:35:c8:09:9b:b2:2f:2f:21:4d:7c:82:0d:de:8a:c9:20:3b:c3:e3:49:0b:7b:20:fe:24:96:42:d9:24:27:09:0d:95:78:45:80:59:1e:f1:a3:a6:bf:81:48:65:af:49:99:f4:0f:fa:fb:ef:a2:71:99:fe:d5:35:40:4c:96:9d:d7:14:aa:60:9f:9c:bb:c8:7f:23:79:93:65:23:d9:be:61:63:a4:0b:98:cd:5b:ed:9b:e7:0e:53:32:7b:f2:70:dc:dc:10:3a:ab:5a:53:a4:05:be:9d:89:c3:9e:86:c0:61:5a`." In this case, the string was a part of the payload. IP and MAC addresses were also often strings other than those appearing in the data. In future research, it would be interesting to experiment with whether it is possible to detect hallucinated answers of LLM by checking the correctness of parameters existing in the provided data.

Llama 3.1 had challenges with data that Wireshark had added to the traffic data. Often, it did not consider the relative sequence and acknowledge numbers that could have given clues about the attack, instead, it only mentioned raw numbers. Sometimes, it also confuses TCP flags and conversation completeness flags. In the case of DNS spoofing, Llama 3.1 did not find short TTL values (TTL=1) or the lack of DNS response as suspicious things even if the data contained a note or warning by Wireshark. More context of the network data or context-learning would be valuable for LLM to understand better the data added by Wireshark. Overall, Llama 3.1 had a high variation in identifying attacks across attack types and packets.

3.4 Discussion

Based on our analysis, Llama 3.1 understands what different attack types mean and what traffic should be transferred with various protocols. It also has many advantages compared to other models, such as traditional ML and DL models. Llama 3.1 was able to understand the context of data, has knowledge about cyber-attacks, protocols, and tools, understood human-readable language and even ASCII characters, and made logical conclusions. In addition, Llama could explain what happened in the data and what could be signs of malicious activity. With such a response, a cyber specialist can quickly verify whether the findings and conclusions are true. Llama 3.1 may also help professionals to focus quickly on suspicious things though it is not yet reliable enough to operate alone. Thus, LLMs are a promising technology in IIoT network traffic analysis in the future. In particular, LLMs may be a valuable tool in explaining the content of network traffic packets.

In this study, we also could use our LLM without fine-tuning or context-learning and analyze data without preprocessing. Our method allowed us to observe how Llama 3.1 works with unfamiliar datasets and various protocols. Our study showed that Llama 3.1 did not require information about normal traffic to analyze network traffic data. Thus, LLMs may be valuable tools when handling heterogeneous data, because it does not need to define all the network's characteristics. However, providing a smaller amount of information can still be useful. More research is needed to determine the necessary information, but we see promising signals here. LLMs must be made to recognize cyber-attacks with sufficient credibility.

Llama 3.1 still has multiple weaknesses that need solutions. The general weaknesses are the computational requirements and its habit of hallucinating. The analysis of packets took so long that it is not meaningful to analyse network traffic packets one by one using LLMs. Additionally, the calculation requires a lot of memory and power, which can be limited in IoT networks. In our analysis, we noticed that Llama 3.1 got confused when it had encrypted data in the prompt, and then only in rare cases, was it able to answer sensibly. In addition, even if Llama 3.1 could identify various clues about attacks from data, this ability was unreliable. Often, it did not identify clear signs which, however, it detected in another run. All in all, Llama 3.1 has the ability to detect malicious activities. There is still a relatively long way to go to detect malicious activities every time and based on weak signals. Data preprocessing, and model fine-tuning or context learning are still needed. The future research question is to determine how much LLMs need preprocessing and fine-tuning.

Some of our experiments were difficult to identify as an attack based on only one packet, especially ARP spoofing. Thus, we tested how Llama 3.1 can detect ARP spoofing if we prompted the model using two packets so that the detection is possible. We used the same prompt as earlier, except the text “The file contains a single packet” was replaced with the real number of packets. A part of Llama 3.1’s analysis is added to Table 3.2 and it shows that Llama 3.1 can utilize multiple packets while analyzing network traffic data. The usage of larger amounts of data would enable a wider analysis of network traffic.

Table 3.2: Llama 3.1's analysis using multiple packets

Attack type	Example of Llama’s response
ARP spoofing	After analyzing the packets, I can see that: * Both packets are ARP responses with opcode 2. * The source MAC address in both packets is the same ('d8:f2:ca:8e:17:69'). * The destination MAC address in both packets is the same ('dc:a6:32:fb:69:b5'). * However, the source IP addresses in the two packets are different ('192.168.0.152' and '192.168.0.101'). This behavior is indicative of an ARP spoofing attack, where an attacker sends fake ARP responses to associate their MAC address with multiple IP addresses on the network.

Based on the results of this study, we decided to test whether we could reduce the hallucination of Llama 3.1. We removed the encrypted payloads from the data and replaced this information with information about what kind of data is removed and how much. Early results show that Llama 3.1 hallucinated significantly less when the data did not contain non-human-readable data.

3.5 Limitations and Future Research

Our study examines only one dataset and one LLM, albeit in depth. In the future, we are going to study more datasets and models. More experiments with normal network traffic are also needed. Large language models are rapidly evolving technology and may require research such as this on a regular basis to detect when models are advanced enough for reliable analysis.

Based on our results we also identified other future research questions. In our study, we did not provide any information about the IIoT network for the model. It would be important to research what kind of information would be most valuable for LLMs and still retain the model’s ability to generalize. Llama 3.1 considers ports, IP, and MAC addresses carefully, so we are going to do more experiments by providing information about used ports and trusted IP and MAC addresses for the model.

Data preprocessing also appears to be required. Handling encrypted strings seems to be the most significant challenge. Is it necessary to delete encrypted data or can the matter be handled with prompt engineering? The usage of a larger amount of data in one prompt is also one future research direction as well as finding out the effect of model fine-tuning or other ways to help LLM with analysis.

First, we intend to prioritize the research on handling encrypted data and the information provided to the model. Based on the results of these tests, we plan the order of the next steps.

3.6 Conclusions

In this work, we analysed Llama 3.1’s ability to detect IoT-targeting cyber-attacks from IIoT network traffic data. We prompted Llama 3.1 with combinations of single network packet captures and possible attack types. Based on our results, Llama 3.1 has multiple advantages compared to other machine learning models, such as its ability to understand the context of data and knowledge about network traffic and cyber-attacks. In addition, our study showed the possibility of using LLM with an unusually low amount of data preprocessing. However, before LLMs can be used in real-life solutions, there are multiple research questions to be solved, such as handling encrypted data, hallucinations, resource intensiveness, and utilizing more input data in prompts.

3.7 References

[1] Statista. 2023. Number of IoT connected devices worldwide 2019-2023, with forecasts to 2030. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[2] Gupta, M., Akiri, C., Aryal, K., Parker, E., & Praharaj, L. 2023. From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy. IEEE Access, 11, 80218–80245. https://doi.org/10.1109/ACCESS.2023.3300381

[3] Al-Hawawreh, M., Aljuhani, A., & Jararweh, Y. 2023. Chatgpt for cybersecurity: practical applications, challenges, and future directions. Cluster Computing, 26(6), 3421–3436. https://doi.org/10.1007/s10586-023-04124-5

[4] Alawida, M., Abu Shawar, B., Abiodun, O. I., Mehmood, A., Omolara, A. E., & al Hwaitat, A. K. 2024. Unveiling the Dark Side of ChatGPT: Exploring Cyberattacks and Enhancing User Awareness. Information (Basel), 15(1), 27. https://doi.org/10.3390/info15010027

[5] Happe, A., & Cito, J. 2023. Getting pwn'd by AI: Penetration Testing with Large Language Models. ArXiv.Org. https://doi.org/10.48550/arxiv.2308.00121

[6] Ferrag, M. A., Battah, A., Tihanyi, N., Debbah, M., Lestable, T., & Cordeiro, L. C. 2023. SecureFalcon: The Next Cyber Reasoning System for Cyber Security. ArXiv.Org. https://doi.org/10.48550/arxiv.2307.06616

[7] McKee, F. & Noever, D. 2022. Chatbots in a Botnet World. arXiv.org. doi:10.48550/arxiv.2212.11126

[8] Sladić, M., Valeros, V., Catania, C. & Garcia, S. 2024. LLM in the Shell: Generative Honeypots. arXiv.org. doi:10.48550/arxiv.2309.00155

[9] Yugeswardeenoo, D., Zhu, K. & O'Brien, S. 2024. Question-Analysis Prompting Improves LLM Performance in Reasoning Tasks. arXiv.org. doi:10.48550/arxiv.2407.03624

[10] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. & Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv.org. doi:10.48550/arxiv.2201.11903

[11] Meta. 2024. Introducing Llama 3.1: Our most capable models to date. Available: https://ai.meta.com/blog/meta-llama-3-1/

[12] Ollama. Available: https://ollama.com

[13] Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L. & Janicke, H. 2022. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. IEEE access, 10, pp. 40281-40306. doi:10.1109/ACCESS.2022.3165809

4 Generative AI for Generating Cybersecurity Data

4.1 Introduction

The generation and application of synthetic data through advanced generative artificial intelligence (AI) models have emerged as transformative solutions in the domain of cybersecurity. Real-world cybersecurity data collection faces significant challenges, including stringent privacy regulations, logistical constraints, and the inherent difficulty of obtaining representative datasets encompassing both normal and attack scenarios. Synthetic data provides a promising alternative by replicating real-world data characteristics without violating privacy boundaries. However, creating synthetic data that achieves the necessary balance among utility, fidelity, and integrity remains a formidable challenge. Poorly generated synthetic data may lack realism or introduce vulnerabilities, compromising its effectiveness for training and testing cybersecurity tools.

4.1.1 The Role and Importance of Synthetic Data in Cybersecurity

Synthetic data serves as a cornerstone for advancing cybersecurity research and application. Its ability to preserve privacy while enabling comprehensive data analysis makes it a critical resource. The scarcity of cybersecurity data, particularly for rare and evolving malicious events, underscores the importance of synthetic data in addressing class imbalances and enhancing IDS capabilities. Synthetic datasets can be tailored to simulate diverse and uncommon attack scenarios, allowing developers to rigorously test and optimize IDS tools. Moreover, high-quality synthetic datasets provide a broader and more diverse training pool for AI models, significantly improving IDS performance and adaptability.

4.1.2 Challenges in Synthetic Data Generation for IDS

Despite its advantages, synthetic data generation faces several challenges. The foremost is ensuring realism: synthetic data must closely emulate real-world conditions, including network behaviour and malicious activity patterns. Insufficient variability limits the capacity to capture the full spectrum of potential threats, making models trained on such data less effective. Addressing class imbalances is another critical challenge, as malicious traffic is often underrepresented in real-world datasets. Privacy concerns remain paramount, as synthetic data might inadvertently expose sensitive details through replication of underlying data structures. Furthermore, standardizing the processes for generating, validating, and integrating synthetic data is complex, creating inconsistencies and reliability issues. Finally, evaluating the performance of synthetic datasets is challenging; high similarity to real data does not always translate to real-world applicability, necessitating advanced and nuanced validation metrics.

4.2 Objectives of the Study

The primary aim of this study is to evaluate synthetic data generation methods based on their utility, fidelity, integrity, and capacity to address class imbalances. Key research questions include:

- Which methods preserve utility and effectively train IDS models?
- How well do these methods maintain the statistical properties of original datasets?
- Can these methods balance class distributions while preserving data integrity?
- Which generative adversarial network (GAN) models are most effective for synthesizing cybersecurity network traffic data?

4.3 Techniques for Synthetic Data Generation

Synthetic data generation methods can be broadly categorized into non-AI-based and AI-based approaches. Each approach offers unique strengths and limitations, as outlined below:

4.3.1 Non-AI-Based Methods

Random Oversampling (ROS): This method duplicates minority class samples to balance class distributions, making it suitable for addressing simple class imbalances in smaller datasets. However, its reliance on duplication can lead to overfitting and a lack of data diversity.

SMOTE (Synthetic Minority Oversampling Technique): SMOTE generates new minority class samples by interpolating between existing samples, thereby enhancing the dataset's diversity. While effective for structured datasets, SMOTE struggles with complex data interdependencies.

ADASYN (Adaptive Synthetic Sampling): Building upon SMOTE, ADASYN focuses on hard-to-learn instances within the minority class, creating synthetic samples in areas with sparse data. This method prioritizes adaptability but may introduce noise in high-dimensional datasets.

Cluster Centroids: This technique reduces majority class samples by clustering and averaging them, preserving overall data structure while addressing imbalances. However, it is less effective for datasets with highly complex or overlapping class distributions.

Gaussian Mixture Models (GMM): GMM represents data as a mixture of Gaussian distributions, effectively modeling sub-populations within datasets. While it excels in capturing statistical properties, its utility for IDS training is limited by computational complexity and scalability issues.

4.3.2 AI-Based Methods

Bayesian Networks (BNs): BNs employ probabilistic graphical models to capture dependencies among variables. Their ability to model complex relationships makes them valuable for feature-rich datasets, but they face challenges in scalability and computational efficiency.

Tabular Variational Autoencoder (TVAE): TVAE extends traditional autoencoders to accommodate tabular data, ensuring that synthetic data preserves original dataset complexity and accuracy. It is particularly effective in maintaining fidelity and utility across diverse datasets.

Diffusion Models (e.g., TabDDPM): Adapted for tabular data, diffusion models offer high-quality synthesis of mixed data types. However, their computational demands limit their scalability for large datasets.

Generative Adversarial Networks (GANs):

- **CTGAN**: Designed specifically for tabular data, CTGAN effectively handles class imbalances and preserves inter-variable relationships. It uses conditional generation techniques to produce high-quality synthetic data.
- **CopulaGAN**: By integrating copula-based statistical modelling with GANs, CopulaGAN captures intricate dependencies among variables, making it highly effective for complex datasets.
- **GANBLR++**: This enhanced version of GANBLR handles mixed data types and captures intricate dependencies, achieving a balance between accuracy and computational efficiency.
- **CasTGAN**: Leveraging a cascaded architecture, CasTGAN generates semantically valid data by accurately modelling interdependencies. However, it struggles with binary constraints and high-dimensional data.

4.4 Evaluating Feature Dependencies with Mutual Information (MI)

Mutual information (MI) provides a robust framework for evaluating dependencies between variables. Unlike correlation measures, MI captures both linear and nonlinear relationships, making it ideal for cybersecurity datasets with complex inter-variable interactions. MI is model-agnostic and unbiased, enabling reliable feature selection by considering relationships among features and their relevance to the target variable.

In this study, MI was used to rank features based on their relevance to the target variable. The top 25% of features were selected for modelling, ensuring computational efficiency and high information retention. This approach enhanced the interpretability and efficacy of IDS models, as the selected features accurately represented critical patterns in the data.

4.4.1 Experimental Setup and Results

The experiments were conducted on advanced hardware, including a 13th Gen Intel® i9 processor and NVIDIA GeForce RTX 4090 GPU. A comprehensive software environment incorporating Python 3.12.4, PyTorch, SDV, and scikit-learn facilitated the evaluation. Preprocessing of datasets such as NSL-KDD and CICIDS-17 involved encoding, feature selection using MI, and dimensionality reduction, ensuring optimized inputs for synthetic data generation.

Statistical methods like SMOTE and Cluster Centroid demonstrated strong performance in preserving class balance and statistical fidelity, achieving up to 99% accuracy in classification tasks. However, these methods lacked the novelty required for generating realistic synthetic data. In contrast, GAN-based approaches, particularly CTGAN and CopulaGAN, excelled in utility and fidelity metrics, achieving utility accuracy rates of up to 98%. TVAE's use of latent probabilistic spaces further enhanced its ability to generate high-quality synthetic data. Although CastGAN showed promise in modelling complex dependencies, it struggled with binary constraints. Bayesian Networks and diffusion models exhibited potential but faced scalability and fidelity challenges.

4.4.2 Future Directions and Implications

Within the CISSAN framework, future research will focus on leveraging GANs to simulate evolving cyber threats and generate realistic attack scenarios. This includes dynamically defining thresholds for anomaly detection and balancing fidelity with privacy requirements. Real-time feature adaptation and dynamic synthetic data generation techniques will further enhance IDS robustness in IoT environments.

The findings of this study highlight the trade-offs between different synthetic data generation methods. While statistical methods like SMOTE and Cluster Centroid effectively address class imbalances, advanced GANs such as CTGAN and CopulaGAN provide superior realism and utility. These results have significant implications for cybersecurity applications, guiding the selection of synthetic data generation techniques based on specific requirements.

By integrating GAN-generated data into frameworks like CISSAN, cybersecurity systems can achieve greater resilience, privacy-conscious operation, and adaptability. This adaptability is particularly crucial for securing critical infrastructures, such as energy systems and IoT networks, against evolving threats. As synthetic data generation methods continue to advance, their role in shaping the future of cybersecurity becomes increasingly pivotal.

5 Analysis of GPS data in the public transport domain

5.1 Introduction

GPS interference, spoofing and jamming have become an increasing cybersecurity concern, affecting especially aviation but to some extent also public transport. Real-time passenger information and computer-aided vehicle location systems within public transport require reliable, frequently updated or close to real-time positioning information from vehicles.

In most contemporary systems, positioning is based on GNSS (global navigation satellite system) technologies, especially on GPS (global positioning system). Alternative sources are positioning satellites (such as Galileo or GLONASS), Inertia measuring units in vehicles, or RFID tags along the pathways or tracks. These and some other solutions, which we plan to elaborate further in deliverable D6.1, require additional equipment and interfaces to be established in vehicles or in the traffic infrastructure.

Within CISSAN, we wanted to explore software solutions in detecting possible anomalies in readily available GPS positioning data. Real-time position data of the buses in a Finnish city (Tampere) were made available to CISSAN via a data acquisition interface. In this brief note for D5.1, the initial discoveries of our technical experiments with the collected real-life data are presented.

5.2 Technical Experiments with GPS Coordinates of buses

The primary goal was to understand the properties of and anomalies in GPS data of buses via data analysis methods.

5.3 Methodological Approach

Given the large volume of GPS data collected daily (approximately 5-7 million coordinates), detecting anomalies requires robust and efficient approaches. Various algorithms, including statistical methods, machine learning, and artificial intelligence, can be applied to identify different types of anomalies. However, there is no gold standard delivering perfect results, necessitating manual assessment for quality assurance during the project.

The data is collected from an MQTT (Message Queuing Telemetry Transport) broker and stored daily in SQLite before being loaded into PostgreSQL for spatial queries and analysis. Preprocessing steps include data cleaning, coordinate transformation, noise reduction, error detection, and spatial indexing.

By applying multiple algorithms, we aim to capture a wide range of anomalies. Each algorithm may highlight different issues, from minor deviations to significant anomalies. The manual review process during the project is essential for ensuring the quality and accuracy of the detected anomalies.

5.4 Initial Experiments and Findings (Work in progress)

- Buses frequently outside the defined area were identified.
- Instances where buses were more than 10 km from their path were counted.
- Maximum distances and frequencies of buses being off path were analysed.
- Devices frequently deviating by at least 1 km were studied.
- Off-path percentages at different times on November 8th and 18th 2024 were compared.

5.5 Additional Analysis

PostGIS (a spatial database extender for PostgreSQL) functions were used to determine the proximity of buses to stops, updating fields to indicate whether a bus is near a stop and the current stop in the sequence. Distances between bus stops and routes were analysed to identify potential issues in stop placement or route mapping.

5.6 Discussion

Frequent small errors are likely due to environmental interference, device sensitivity, or route characteristics. Infrequent large errors may be caused by temporary signal loss, device malfunction, or data transmission glitches. Mixed patterns suggest variable environmental conditions or intermittent device issues.

5.7 Future Work

The plan for future work focuses on developing and testing anomaly detection methods in public transportation data. It includes:

- **Preliminary Analysis and Data Preparation**:
  (1) Collect and preprocess data, generate additional variables, and clean the data.
  (2) Create new variables using the existing ones.
  (3) Create proper aggregation to handle long-term datasets.
- **Algorithm Selection and Initial Testing**: Select and test such algorithms as statistical methods, Local Outlier Factor (LOF), and Isolation Forest to find anomalies. These algorithms should gain from a large variable set.
- **Detailed Analysis and Optimization**: Fine-tune parameters and optimize algorithms for better performance.
- **Validation and Final Evaluation**: Validate algorithms on a separate dataset and compare performance.
- **Implementation and Deployment** (Optional): Integrate the best-performing algorithm for real-time anomaly detection and provide training and documentation for end-users.