Detection and Mitigation of Cyber Attacks in Smart Grid Networks

MS (Research) Thesis

By

Nisha Kumari Barsha

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2024

Detection and Mitigation of Cyber Attacks in Smart Grid Networks

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

in fulfillment of the requirements for

the award of the degree

of

Master of Science (Research)

by

Nisha Kumari Barsha

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2024

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled "Detection and Mitigation of Cyber Attacks in Smart Grid Networks" in the fulfillment of the requirements for the award of the degree of MASTER OF SCIENCE (RESEARCH) and submitted in the DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING, Indian Institute of Technology Indore, is an authentic record of my own work carried out during the time period from August 2022 to May 2024 under the supervision of Prof. Neminath Hubballi, Professor, Indian Institute of Technology Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Nisha Kumari Barsha 22/05/2024

Signature of the Student with Date

(Nisha Kumari Barsha)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of Thesis Supervisor 1

with Date 22-May-2024

(Prof. Neminath Hubballi)

Nisha Kumari Barsha has successfully given her MS (Research) Oral Examination held on _____

Signature of Chairperson, OEB	Signature of External Examiner	Signature of Thesis Supervisor #1
Date:	Date:	Date:
Signature of PSPC Member #1	Signature of PSPC Member #2	
Date:	Date:	
Signature of Convener, DPGC	Signature of Head of Department	
Date:	Date:	

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to people who in one or the other way contributed by making this time as learnable, enjoyable, and bearable as possible. At first, I would like to thank my supervisor **Prof. Neminath Hubballi**, who was a constant source of inspiration during my work. With his constant guidance and research directions, this research work has been completed. His continuous support and encouragement has motivated me to remain streamlined in my research work. I am also grateful to **Dr. Ranveer Singh**, HOD of Computer Science for all his help and support.

I am thankful to **Dr. Bodhisatwa Mazumdar** and **Prof. Trapti Jain**, my research progress committee members for taking out some valuable time to evaluate my progress all these years. Their valuable comments and suggestions helped me to improve my work at various stages.

My sincere acknowledgement and respect to **Prof. Suhas S. Joshi**, Director, Indian Institute of Technology Indore for providing me the opportunity to explore my research capabilities at Indian Institute of Technology Indore.

I would like to express my heartfelt respect to my parents, my friends, and my elder brother for the love, care, and support they have provided to me throughout my life.

Nisha Kumari Barsha

To you as a reader

ABSTRACT

Smart grid networks use Supervisory Control and Data Acquisition (SCADA) systems for managing the grid network. These are critical infrastructure meeting energy demands of consumers. SCADA systems collect measurement data from different places of the grid network to make safety-critical control decisions. However, these networks use TCP/IP networks for transmitting such data and this has exposed them to cyber attacks, necessitating effective detection mechanisms. This thesis presents a three-fold contribution to enhancing the security of smart grids.

In the first contribution, a comprehensive approach is taken to identify and address cyber threats in smart grid networks. Three broad classes of anomalies, namely single message anomaly, message sequencing anomaly, and time based anomaly, are introduced. We show that several cyber attacks in smart grid networks can be detected by identifying these three types of anomalies. A novel state transition machine model, `Deterministic Counting Timed Automata (DCTA)` is proposed to identify these anomalies. `DCTA` formalizes constraints on message attributes, event timing, and counter values associated with states, showcasing its efficacy in detecting cyber attacks. Experimental validation with a publicly available dataset establishes the capability of `DCTA` and its benchmarking against a recent method from the literature.

The second contribution addresses specific security threats to smart grid infrastructure, focusing on flooding and bruteforce attacks. We model the Modbus communication messages used in SCADA systems as a probability distribution representing normal communication. Subsequently, we compare the probability distribution of a test interval with the base distribution using Hellinger distance metric to detect anomalies. Experimental evaluations using two publicly available datasets demonstrate the method's effectiveness in detecting flooding and bruteforce attacks, highlighting its potential for enhancing the security of smart grid networks.

In the third contribution, we study two types of attacks known as malformed and message sequencing attacks. In response, a method using first-order logic statements is introduced for the detection of variants of malformed messages. Furthermore, a

filtering mechanism using the Extended Berkeley Packet Filter (eBPF) is proposed to identify and mitigate the impact of malformed and sequencing attacks. Experimental results demonstrate the effectiveness and robustness of this filtering approach against diverse attack variants and intensities.

Collectively, these contributions provide a holistic approach for strengthening the security of smart grid networks by identifying, formalizing, and mitigating various cyber threats. The methodologies introduced in these works contribute to the ongoing efforts to develop advanced and adaptive security mechanisms for critical infrastructure.

List of Publications

Journals

1. **N.K. Barsha** and N. Hubballi, "Anomaly Detection in SCADA Systems: A State Transition Model", *IEEE Transactions on Network and Service Management,* 2024 (Accepted).

Conferences

1. N. Hubballi and **N.K. Barsha**, "Mitigating Resource Depletion and Message Sequencing Attacks in SCADA Systems", *38th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 37-47, 2024.

2. **N.K. Barsha** and N. Hubballi, "Detecting Cyber Attacks in Smart-Grid Networks with Probability Distribution Comparison", *24th IEEE Consumer Communications & Networking Conference (CCNC)*, pp. 648-649, 2024.

3. **N.K. Barsha** and N. Hubballi, "Network Flow based Cyber Attack Detection in Smart-Grid Networks", *PhD Forum 18th International Conference on Information Systems Security (ICISS)* 2022.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Industrial Control Systems (ICS) play a crucial role in overseeing and managing critical infrastructure such as electric grids, water distribution networks, and chemical plants. Typically, Supervisory Control and Data Acquisition (SCADA) systems are used as part of ICS to monitor and control these infrastructures. For e.g., in water distribution networks, SCADA has completely revolutionized the management of water resources by providing real-time monitoring and control capabilities. By collecting data from sensors installed across pipes, treatment plants, and reservoirs, SCADA helps operators in optimizing pumping schedules and flow rates, thereby enhancing resource utilization. Additionally, this information also helps the operator to detect pipe leakage and helps in regulating the water pressure to meet the demand. Thus, SCADA helps in enhancing the capability and reliability of water distribution networks. On the other hand, SCADA serves as a control hub for the smart grid networks. It allows operators to keep a close eye on how electricity flows in real time. By collecting measurement information from sensors deployed in the grid network, it gives insights into parameters like voltage, power, and status of equipment/device. Operators can make better decisions with this data like changing the route of power supply to balance the load. SCADA can also remotely manage devices, thus enabling a steady supply of power. In crisp, SCADA systems facilitate the transfer of measurement data to a central control center where control decisions are made.

Communication protocols are essential to facilitate these data transfers and com-

mand exchanges. These protocols serve as a bridge through which information flows and commands are executed within the smart grid network. There are both proprietary and open design protocols available to facilitate these communications. Proprietary protocols are vendor specific and offer optimized performance but lack the feature of interoperability, whereas open source protocols have interoperability features and are preferred more because of the transparency. Application protocols like Modbus [2] and IEC 60870-5-104 [3] were earlier proprietary but now their design is made public and standardized. These protocols have gained popularity for communication within these systems. However, these protocols lack sufficient security measures, especially as SCADA networks (traditionally isolated) are now being integrated with the internet and these messages are being sent over TCP/IP making them vulnerable to cyber attacks. Incidents like the STUXNET attack [4] on the Iranian Nuclear Plant and the Ukrainian Grid network [5] underscore the severity of such vulnerabilities.

While traditional protection methods involve the installation of firewalls and intrusion detection systems, they may not be specifically tailored for safeguarding SCADA system communications. Moreover, challenges exist in monitoring internal communications due to the typical placement of these security measures at the network perimeter. The risk of cyber attacks initiated through compromised devices within the SCADA infrastructure calls for a focus on monitoring application layer communications for anomaly detection.

The rest of this chapter is organized as follows. The motivation behind our work is described in Section 1.1 and also highlights the objectives of our work. In Section 1.2 we summarize the thesis contributions. Outline of the rest of the thesis is described in Section 1.3.

1.1 Motivation

Smart grid security is crucial for ensuring the reliability as well as stability of modern power systems. However, smart grids are susceptible to cyber attacks due to the use of TCP/IP networks over which SCADA communications happen. Attackers may

modify or steal sensitive information such as energy consumption patterns, customer bills and details, etc. They can even hamper the critical components of the smart grid devices. Hence, it is really crucial to detect these attacks in order to ensure the safety of smart grid.

Our main motivation is to enhance the security of smart grids by detecting and mitigating cyber attacks. We acknowledge the inherent threats that are introduced by the use of TCP/IP networks in smart grids. The existing techniques proposed for attack detection and mitigation mainly include machine learning models, state transition models, and other techniques like behaviour based models, signature based models, etc. Machine learning models entirely depend on the quality and quantity of data that is being used for training. Thus, it makes them inefficient due to the limited availability of real-time data. Available state transition models are not able to categorize attacks into different categories based on their nature. In addition to this, they fail to detect DoS attacks. Thus, we propose a mechanism to overcome the limitations of these available techniques. We also note that in the case of DoS attacks, the volume of data received is very high, and detection models built using state transition models may not be efficient in tracking them. Owing to this there is a need for a detection method which is computationally efficient to handle volumes of packets received. This helps in speeding up the detection process. Further, we notice that all messages are being sent to the control center where processing is done. A motivated adversary can impact this control center by sending random packets (illegal/malformed packets) with an objective of depleting its resources. Attackers craft malicious packets in order to waste system resources, which we term as resource depletion attack. We set mitigating such attacks as another objective for our work.

In this thesis, we mainly deal with techniques to detect and mitigate cyber attacks at the application layer in smart grid networks. For this, we propose various models. Previous works [6, 7] have shown that analyzing flow level details of SCADA communications is a reliable method to detect attacks. Several works [7, 8, 9, 10] proposed attack detection methods based on state transition models owing to the periodic nature of communication in SCADA systems. Such models can flag unusual patterns

by identifying sequences and patterns of messages. However, these models have their own drawback which includes the inability to detect DoS attacks and categorization of attacks. There are a number of ways for detecting DoS attacks in conventional networks but it has not been explored much in smart grid setup. The traditional methods to safeguard include the use of firewalls. However, these firewalls themselves are susceptible to attacks. Also, the packet filtering mechanism is not reliable unless the packet content is thoroughly examined. Various detection mechanisms [9, 11] have been introduced to address this issue. However, this mechanism works as an application, its monitoring code executes in user space and utilizes the resources for context switch in between user and kernel space. To overcome these limitations present in the existing techniques, we made three major contributions which are highlighted in Section 1.2. Drawing motivation from this, we set the following as our objectives for the work.

1. To detect and categorize different cyber attacks at the application layer in smart grid networks.

2. To design a lightweight method for detecting flooding based DoS attacks in smart grid networks.

3. To mitigate resource depletion attacks in smart grid networks.

To achieve these objectives, we make three contributions as outlined in Section 1.2.

1.2 Thesis Contributions

A brief overview of our research contributions is provided below, and more details are available in the later chapters.

I State Transition Modelling for Cyber Attack Detection: While acknowledging the limitations of existing state transition based models, we design a new state transition machine based model to detect and categorize different cyber attacks. We categorized attacks into three anomaly types, namely, single message anomaly,

message sequencing anomaly, and time based anomaly. Identifying these anomalies helps in detecting different cyber attacks. Our proposed state transition machine model can identify these anomaly types in SCADA (Supervisory Control and Data Acquisition) communications. This state transition model has been constructed using the extended network flow records originating from SCADA communications in smart grid networks. To evaluate the performance of our proposed model, we used a publicly available dataset and benchmarked its performance against the prior works.

II Attack Detection Using Probability Distribution Comparison: In our second contribution, we propose a lightweight method for detecting flooding attacks and bruteforce attacks. This detection method is motivated by the fact that SCADA communications are highly periodic and synchronized in nature. Hence, we utilized this property to detect attacks. We describe a probability distribution based comparison method for detecting the attacks. Here we generate a baseline probability distribution using different messages of Modbus communications. Subsequently, we compare a distribution generated from a known test interval with the distribution generated earlier to detect anomalies in communication patterns. We evaluated the performance of our lightweight model via experiments conducted on publicly available datasets for flooding and bruteforce attacks targeting smart grid networks.

III Mitigating Resource Depletion Attack in Smart grid: In our last contribution, we focused on mitigating the resource depletion attack generated using malformed packets. Traditional methods of protecting SCADA systems, such as deploying firewalls, have limitations as they rely on software solutions which are vulnerable to attacks and may not thoroughly inspect packet content. Here we want to filter malformed packets from utilizing any sort of resources. To address this, we designed a method for offloading the filtering task to network interface cards. Our study identifies various malformed attacks on protocols like IEC-104 and Modbus and proposes a method to detect them using proposed First Order Logic Predicate statements which are generated using the protocol specifications. By shifting the screening and filtering of these malformed packets to programmable network interface cards, we aim to reduce system resource overhead while ensuring robust security measures against attacks.

1.3 Organization of the Thesis

This thesis is organized into six chapters. A summary of each chapter is provided below:

Chapter 1 (Introduction)

In this chapter, we provide a concise summary of our research, including the background necessary for understanding our work, the motivation for our work, the unique contributions of our thesis, and the overall organization of the subsequent sections.

Chapter 2 (Literature Survey)

In this chapter, we discuss the related work on cyber attack detection and mitigation in smart grid networks. All the major techniques used for this purpose are explained in detail.

Chapter 3 (SCADA Anomaly Detection: A State Transition Modeling)

In this chapter, we present our first contribution: State Transition Modelling for cyber attack detection. Preceding this, we introduce three generic anomaly types and demonstrate how all existing cyber attacks can be categorized within this framework.

Chapter 4 (Detecting Flooding and Bruteforce Attacks in Smart Grid Networks with Probability Distribution Comparison)

In this chapter, we introduce a lightweight probabilistic distribution-based technique for detecting flooding and bruteforce attacks.

Chapter 5 (Mitigating Resource Depletion and Message Sequencing Attacks in SCADA Systems)

In this chapter, we propose an eBPF-based filtering technique to mitigate resource depletion attacks mounted against smart grid control center.

Chapter 6 (Conclusion and Future work)

In this chapter, we summarize and conclude the work in our thesis and discuss the future directions in the area.

Chapter 2

Literature Survey

As mentioned in Chapter 1, Supervisory Control and Data Acquisition (SCADA) systems play a crucial role in smart grid networks by facilitating the transmission of measurement data from various sensory nodes to control centers. This data is then used to make important decisions regarding the operation and management of the grid, including safety-critical control decisions. Traditionally, SCADA systems relied on proprietary communication protocols and networks, which provided a certain level of security as they were isolated. However, modern grids exchange messages over TCP/IP networks for improved connectivity and interoperability. Hence, they have become vulnerable to cyber attacks. Given the critical nature of smart grid infrastructure, the security of SCADA systems is of utmost importance. To enhance the security of the smart grid, it is important to create systems capable of detecting and mitigating potential cyber attacks that could target the grid infrastructure. Designing appropriate detection and mitigation techniques requires an understanding of various threat vectors. In the literature, there is a good coverage of different types of cyber attacks that pose a threat to SCADA systems in smart grid networks. Attacks like connection-loss, which disrupts the communication between devices and control centers; DoS attack, which overwhelms systems with traffic to make them unresponsive; scanning attack, which looks for vulnerabilities in networked devices; injection attacks like OS injection and SQL injection, exploiting vulnerabilities in system inputs to execute unauthorized commands or access data, etc. have been covered. However, this is

not an exhaustive list of possible attacks. For a complete taxonomy of various attacks, the reader is referred to [12].

Grids are critical infrastructure and millions of users depend on them for their energy security. Thus, it is important to safeguard these networks from cyber attacks. There are both detection and mitigation techniques proposed for the safety of these networks. We review the existing literature on detecting and mitigating cyber attacks in smart grid networks. Existing works related to the security of these systems fall into the following two categories.

`(i) Studying Attack Vectors:` Several prior works [13, 14, 15] cover cyber attacks against grid and SCADA systems and associated protocols. Most studied attacks include false data injections [15] and denial of service attacks [16]. False data injection attacks generate false measurement data and have it sent to the control center to mislead the state estimation or assessment. On the other hand, DoS attacks prevent the normal operation of either the control center or measurement units/clients from performing its operations. We study a different class of attacks that target the computational resources of the control center. Such attacks are studied in other domains like web servers [17] using which the server resources are depleted.

`(ii) Detecting the Attacks:` These methods develop techniques to detect different types of attacks. There are several machine learning algorithms [18, 19], state transition modeling approaches [9, 20, 7] and commercial IDS solutions [21] available for this purpose. Many of these techniques rely on Deep Packet Inspection (DPI) [22] for detecting these attacks. However, these methods detect conventional attacks like scanning, injection, connection-loss, etc. Lin et al. [23] adopted Bro [24] (now called zeek) intrusion detection system (IDS) for SCADA networks. Bro being a signature based IDS; detects attacks by matching signatures. The authors also propose to filter malformed messages in SCADA systems. However, their study is limited to a limited number of attribute types and that to of only DNP3 protocol. Further, Bro rules were manually written for detection. As our work mainly focuses on detecting and mitigating different attacks in the SCADA systems, we provide an elaborate discussion on the various detection methods subsequently.

We organize the prior works for attack detection into three groups, i.e. Machine Learning-based Models, State Transition Models, and Other Methods like Signature-based intrusion detection systems, Statistical Methods, etc. as shown in Figure 2.1. These three categories of work are elaborated in the next three sections.

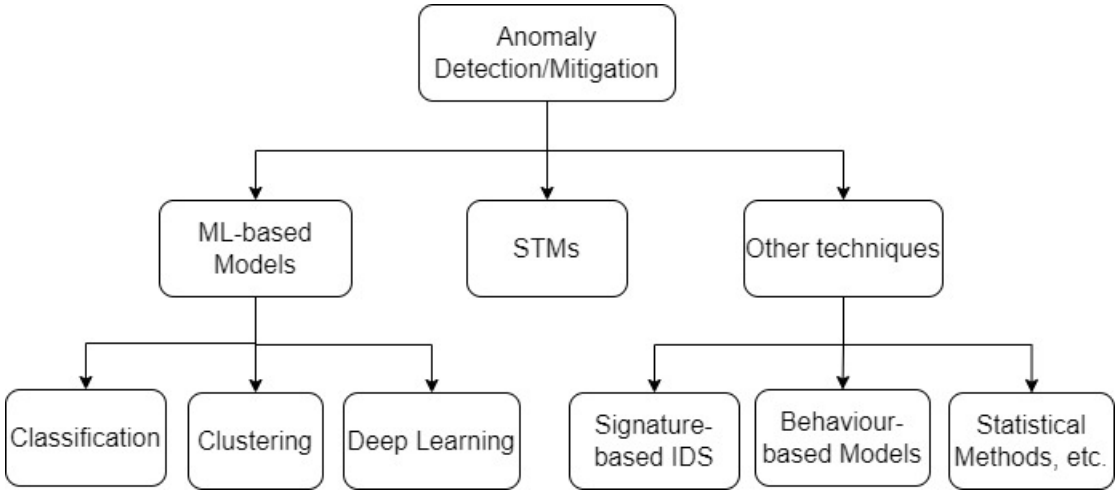


Figure 2.1: Attack Detection in SCADA Systems

2.1 Machine Learning-based Models

Extensive research has been done on the use of machine learning (ML) based models for cyber attack detection in SCADA networks. These models are trained using good quality data that comprises of readings from Remote Terminal Units (RTUs) and data collected from networks, which includes detailed information at the flow and packet levels. For feature selection for training purposes, a complete range of parameters is explored, ranging from electrical attributes like voltage, current, angle, and phase values to network-centric features like traffic volume, source and destination IP addresses, and even packet payload content. To detect attacks, different classification algorithms such as decision trees, random forests, and support vector machines are used. The major drawback for these machine learning models lies in the limited availability of real world data that is needed for the training of these models. A major concern with these detection methods is of false positives. Machine learning models tend to gener-

ate a large number of false positives. Also, the interoperability of machine learning models is not straightforward, and mostly its operations are opaque and may not give a structured representation of the system's behavior and attack patterns. Major ML techniques include Classification, Clustering, and Deep Learning techniques. In the following subsections, we elaborate these machine learning techniques describing how they are used for attack detection in smart grid networks.

2.1.1 Classification

Classification methods in machine learning are one of the crucial techniques for attack detection in SCADA systems. It uses both normal scenario as well as attack scenario data for training purposes. These methods include a wide range of techniques such as Decision Tree [25], Random Forest [25, 18], and Support Vector Machines [25, 18, 26], which are trained on features extracted from sensor data, control commands, and network traffic logs. Decision tree and random forest algorithms construct a tree with attributes of training data with leaf nodes indicating the labels. Support vector machine decide a decision boundary around the samples separating both normal and attack cases. Leandros et al. [27] propose a method based on One-Class Support Vector Machine (OCSVM) for intrusion detection in SCADA. Their work uses a central OCSVM and multiple OCSVMs are automatically generated for each significant traffic source in the system. Also, it uses social metric analysis, aggregation techniques, a voting mechanism, and K-means clustering to categorize final alerts. In a separate work, Simon et al. [28] analyze industrial operation network data using machine learning and time series-based anomaly detection algorithms to detect attacks. SVM and Random Forests are used for attack detection, with Random Forest showing slightly better performance than SVM. Xiangwu et al. [29] propose a fault detection algorithm, which is developed using a multi-layer neural network (MNN) and random forest (RF) based on SCADA data. The MNN detects faults early by analyzing reconstruction errors, while RF accurately identifies fault types. Vivek et al. [30] introduce an Intelligent Remedial Action Scheme (IRAS) designed to detect cyber attacks from physical disturbances in smart grids. They propose a decision tree-based

anomaly detection method using voltage and current phasor values to differentiate between normal power line faults and malicious tripping attacks on physical relays to enhance systems efficiency and reliability.

These models are trained on diverse feature sets derived from Remote Terminal Unit (RTU) and Phasor Measurement Unit (PMU) data, having measurements such as current, voltage, as well as attributes extracted from network packets. Such features provide an overview of grid behavior. After training and evaluation, these classification models are used for real-time monitoring. A trained model looks for anomalous behavior to detect attacks. With regular upgradation and refinement, these methods help to maintain the security and reliability of the smart grid despite of evolving cyber security challenges.

2.1.2 Clustering

Clustering in machine learning is a technique where a set of data points are grouped into clusters based on their similarities. Clusters are formed by maximizing intra-cluster similarity and minimizing inter-cluster similarity. Clustering techniques use a variety of algorithms to differentiate between data samples with similarities and differences. Islam et al. [31] proposed a cluster-based parallel-ensemble method for attack detection, which aims for both high robustness against evasion attacks and accurate theft detection. This method operates on two levels of defense, i.e. clustering and ensemble. Clustering helps reduce regularization and enhance robustness by grouping similar patterns together, whereas an ensemble of diverse decision models helps to improve robustness against transferability issues. The combination of both these techniques helps in identifying instances of theft with minimized false positives.
To detect cyber attacks in SCADA, various clustering algorithms like K-means clustering [32, 26] and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [33] have been used. The K-means clustering algorithm works by dividing the dataset into several groups via repetitive phase-wise partitioning. Each data point is assigned to the cluster with the nearest centroid and this process continues until convergence is achieved. On the other hand, DBSCAN operates by expanding the size of

clusters by merging nearby and similar data points into cohesive groups. Here, attacks are often identified as outliers or as small clusters within the data distribution. These may represent a deviation from expected patterns, indicating a potential malicious activity. However, this approach is still prone to generating false alarms, as certain legitimate communications or behaviors can also be classified as outliers or anomalies by the algorithm. This draws attention to the need for careful interpretation of results to differentiate between actual threats and misclassified threats.

2.1.3 Deep Learning

Recent studies [34, 35] have utilized deep learning, specifically convolutional neural networks (CNNs), to identify cyber threats in SCADA systems. These models automate feature extraction from data but operate as "black boxes", lacking transparency in how they choose features and assess performance. This lack of transparency poses challenges in understanding how the models make decisions, making them less reliable. Without insight into which features the model uses and how they impact predictions, it's hard to assess reliability and applicability. Additionally, the absence of detailed information on feature selection and performance evaluation hampers reproducibility and comparison across studies.

In a recent work of Sayawu et al. [36], authors proposed a specialized Genetically Seeded Flora Transformer Neural Network (GSFTNN) algorithm for intrusion detection by analyzing operational pattern anomalies. They used the Washington University St. Louis Industrial IoT 2018 (WUSTL-IIOT-2018) dataset [37] for the experiment and evaluation purpose. This dataset has data pertaining to various attacks, including port scanning, address scanning, device identification, aggressive model device, and exploit device attacks. David et al. [38] proposed a novel deep learning based framework for attack detection in smart grid, using unsupervised feature learning. This approach automatically identifies crucial patterns for attack detection in transmission SCADA systems, reducing dependence on explicit system models and human expertise. On the similar lines, Ahmad et al. [39] proposed a deep learning approach for constructing a robust and adaptable Intrusion Detection System (IDS),

utilizing a multi-layer perceptron and binary-based architecture. This implementation demonstrates efficiency in detecting intrusion as well as offers flexibility to detect variety of threats. Additionally, Sasanka et al. [40] developed a dataset and utilized deep learning techniques like Stacked Auto-Encoders (SAE) and Deep Belief Networks (DBN) for feature extraction, followed by classification using Support Vector Machine (SVM) and Softmax Regression (SMR).

While there are a significant number of Machine Learning and Deep Learning models for attack detection in SCADA systems, these methods call for a comprehensive and representative dataset during the training phase [18]. However, these methods are prone to generating false alarms, which can hamper the overall effectiveness of the security system [25]. To address these challenges, Suaboot et al. [41] suggest leveraging ensemble-based techniques like Voting and Boosting to enhance accuracy. Additionally, they also propose integrating automated detection with rule-based solutions, offering flexibility in identifying unknown malicious incidents. On the other hand, Hadir et al. [1] demonstrate that machine learning algorithms are vulnerable to adversarial attacks, resulting in a significant drop in accuracy. In response, they introduce a hierarchical multi-layer perceptron (HMLP) and use defensive distillation to enhance the model's resilience against such attacks. In summary, machine learning methods require good quality data and can sometimes trigger false alarms. Thus, researchers are exploring techniques like rule-based solutions to improve accuracy and flexibility. However, adversaries can exploit vulnerabilities in machine learning models, leading to reduced accuracy. To counter this, techniques such as hierarchical multi-layer perceptrons and defensive distillation are being used to make the models more robust against attacks.

2.2 State Transition Models

State transition modeling involves representing the behaviors of a smart grid system as a series of states and transitions between them. These states can represent different operational conditions or specific error situations that may occur in SCADA

systems. By modeling the system behavior, it is possible to detect anomalies or unusual patterns in the system's behavior. One approach to anomaly detection using state transition model involves identifying sequences of actions that have not been observed before or that are rare based on their probability. For example, if a sequence of operations are seen that deviates significantly from a known typical behavior, it may indicate a potential security threat or malfunction in the system. Steven et al. [8] were among the first to apply this method to Modbus communications, creating a State Transition Model (STM) to detect anomalies. Authors have utilized the improved uniformity and stability of control systems concerning their structure, communication, and setup to implement a lightweight model based intrusion detection system. Similarly, Goldenberg and Wool [9] introduced a model-based intrusion detection system tailored for Modbus/TCP networks commonly used in SCADA systems. Their method identifies that Modbus traffic between human-machine interfaces (HMIs) and programmable logic controllers (PLCs) follows a highly predictable pattern. The system utilizes deterministic finite automata (DFA) to model each HMI-PLC channel uniquely, automatically constructing DFAs from around 100 captured messages. This approach enables deep inspection of Modbus/TCP packets, offering a detailed traffic model for effective intrusion detection. However, Caselli et al. [10] found that not all Modbus communications follow predictable cyclic patterns, as assumed by previous works. Hence, they instead modeled Modbus communication as a Discrete Time Markov Chain (DTMC), which allows for more flexibility in capturing diverse communication behaviors. To address challenges such as burst traffic patterns observed in SACDA systems [42], researchers have developed specialized models like burst DFA (Deterministic Finite Automaton) to capture these behaviors accurately. Fabio et al. [43] present a novel technique aimed at identifying and mitigating attacks targeted at SCADA systems. The method utilizes a model checking approach, which involves converting time-series logs retrieved from SCADA systems into a structured network of timed automata. Through the application of timed temporal logic, the behavior exhibited by a SCADA system during an attack scenario is precisely described and analyzed. This method offers a systematic and formalized means of understanding

the dynamics of SCADA systems under attack, facilitating more effective detection and response strategies. Moreover, Matoušek et al. [7] introduced a method to model Industrial Control Systems (ICS) communications using their probability or frequency values within automata structures. They employed two types of automata: Deterministic Probabilistic Automata (DPA) and Prefix Tree automata. The traffic between two devices is also analyzed as conversations. In their approach, each conversation is represented as a string along with its corresponding frequency or probability of occurrence. Prefix Tree automata assign frequency values to transitions, indicating how frequently a particular conversation appears in training sequences. The Alergia learning algorithm is utilized to generate Deterministic Probabilistic Automata (DPA), which assigns probability values to transitions. These probabilities are calculated by dividing individual transition frequency values by the overall frequency of all outgoing transitions and the frequency of state acceptance. Thus, each transition probability contributes to the probability of a string. Moreover, Alergia optimizes the DPA by merging states to reduce its size. This method provides a formal yet accessible framework for modeling ICS communications, facilitating more effective analysis and understanding of system behavior.

The majority of the state transition models discussed primarily concentrate on capturing the sequential pattern of messages exchanged between components. Consequently, these models may overlook various anomalies, including timing irregularities and disruptions in connections, as they do not account for temporal aspects or broader contextual meanings beyond message order.

2.3 Other Methods

There are also other approaches for securing smart grid networks against cyber threats. One commonly used method is signature-based intrusion detection systems [44]. These systems rely on predefined rules or signatures to identify known attacks. Essentially, they match observed network activity against a database of known attack patterns. On the other hand, behavior-based models [45] take a different approach.

Instead of relying on predefined signatures, these models focus on detecting anomalies by comparing current behavior to established patterns of normal behavior. Any deviations from these patterns are flagged as potential threats. Statistical methods play a crucial role in detecting anomalies as well. For instance, control charts [46] monitor system behavior for deviations from expected norms. These charts establish control limits based on historical data and raise alerts when observed behavior falls outside these limits. Ivana et al. [47] demonstrate the effectiveness of statistical modeling in identifying anomalies that arise from irregular transmissions, device/link failures, and cyber attacks such as packet injection, scanning, or denial of service (DoS). It outlines the automatic creation of a statistical model from a training dataset, presenting two distinct profiles: a master-oriented profile for one-to-many communication and a peer-to-peer profile describing traffic between two ICS devices. The proposed method can easily be integrated into an intrusion detection system (IDS) or anomaly detection (AD) module. Regression analysis examines relationships between different variables in the system. Regression analysis can uncover potential security threats by identifying patterns that do not comply with expected communication patterns. Bhattacharjee et al. [48] present two methods to undermine anomaly-based attack detectors in smart metering infrastructure and compare their effectiveness. Using the L1 norm instead of the L2 norm for threshold learning provides some defense against fast gradient value inspired adversarial data poisoning(FGAV) due to induced gradient shattering. Experimental results demonstrate a lesser impact of poisoning attacks with the L1 norm. Additionally, the paper introduces threshold learning for anomaly detection using robust loss functions under quantile and unweighted regression. The evaluation shows that the Cauchy loss function performs better for impact robustness than Huber loss. Moreover, quantile weighted regression outperforms regular regression when considering the expected time between false alarms. Another approach involves time-series analysis [49, 50]. This method analyzes data patterns over time to identify anomalies. Sudden changes or irregularities in these patterns may indicate malicious activity. Numerous anomaly detection systems rely on the cyclic nature of polling mechanisms within SCADA systems. However, the usability of anomaly

detection systems centered around non-polling traffic or spontaneous events remains largely unexplored. To address this gap, Yin et al. [50] introduce a new method for modeling the timing characteristics of spontaneous events within an IEC-60870-5-104 network and utilize this model for anomaly detection. The system's effectiveness is evaluated using a real-time dataset obtained from a power utility, which includes injected timing effects from two attack scenarios. One scenario involves persistent malfunctioning in field devices, leading to consistent timing anomalies, while the other scenario involves intermittent anomalies caused by malware on field devices, which is considered stealthy. The results indicate promising detection accuracy and timing performance for scenarios with persistent anomalies. However, for scenarios with intermittent anomalies, the effectiveness of the approach is demonstrated primarily in instances of low-volume traffic or attacks lasting over one hour.

Akashdeep et al. [51] address the vulnerability of Industrial Control Systems (ICS) to cyber attacks due to increased networking and automation. They introduce a novel approach using process analytics to detect attacks in ICS infrastructure. The study compares this method with traditional signature-based detection techniques. A pattern recognition algorithm named "Capturing-the-Invisible (CTI)" is proposed to uncover hidden processes in ICS device logs, enabling real-time detection of Behavior-based attacks. A recent work [52] proposes to model the sensor/actuator driven communication which is not initiated by the control center to identify Advanced Persistent Threats (APTs) and malware infections. By employing Multivariate Correlation, this approach identifies anomalies in communication patterns, helping to safeguard smart grid networks against sophisticated cyber attacks.

In summary, the pool of techniques for securing smart grid networks against cyber threats offers a variety of approaches, each with its own strengths and limitations. Signature-based intrusion detection systems offer a reliable means of identifying known attacks by matching observed network activity against a database of predefined rules or signatures. On the other hand, behavior-based models focus on detecting anomalies by comparing current behavior to established patterns of normal behavior, offering adaptability to emerging threats but requiring robust baseline data. Statistical

methods, such as control charts and regression analysis, provide valuable insights into system behavior deviations, yet they may have issues with the complexity of real-time anomalies and evolving attack strategies. Time-series analysis offers promise in identifying irregularities over time, but its effectiveness varies depending on the nature of the anomalies and the volume of traffic.

Chapter 3

SCADA Anomaly Detection: A State Transition Modeling

3.1 Introduction

Industrial Control Systems (ICS) are used for managing smart grid SCADA infrastructure. These systems collect measurement data and export it for decision making at a central control center. Various proprietary protocols [53, 54] have been developed for transmitting these measurements to the control unit/master. Modbus [2] and IEC 60870-5-104 protocol [3] are among the most popular choices for communication protocols due to their standardized designs. However, these protocols lack security measures, especially considering that SCADA networks were traditionally isolated from the internet. Off late SCADA systems are being integrated into the internet via TCP/IP communication. Due to this, they have become vulnerable to cyber attacks [55, 56]. Incidents like the cyber attack on the Ukrainian Grid network [5] highlight the potential consequences of such vulnerabilities. Traditional methods of protecting critical infrastructure often involve the installation of firewalls and intrusion detection systems. However, these approaches are not specifically designed to safeguard SCADA system communications. Additionally, monitoring internal communications can be challenging as these systems are typically deployed at the network perimeter. Further, cyber attacks can originate from compromised devices within the

SCADA infrastructure through malware or other means. Therefore, it is crucial to analyze SCADA communications at the application layer to detect anomalies or cyber attacks effectively. Previous research [6, 7] has demonstrated the effectiveness of monitoring network flow level information of SCADA communications for detecting attacks. Owing to the periodic nature of communication in SCADA systems, several attack detection models [7, 8, 9, 10] described state transition machine based detection systems. While these models can identify unusual message sequences, they may not detect all variants of attacks, such as Denial of Service (DoS) attacks initiated by sending excessive legitimate messages. Additionally, these methods often utilize complementary approaches, such as generating probability distributions from individual state transition machines, to detect specific attack types. Motivated by the success of state transition models and flow level information monitoring, we design a new state transition model for detecting different types of anomalies and hence cyber attacks in smart grid networks. This model not only identifies unusual event sequences but also incorporates different types of constraints, including timing, the number of events of a particular type, and application message values within a single state transition machine. This enhanced approach improves the chances of detecting attacks, reduces false alarms, and minimizes overhead significantly. In specific our contributions in this chapter are the following.

- We categorize three types of anomalies and demonstrate that various cyber attacks can be identified by detecting these anomaly types.

- We introduce a novel state transition machine model, Deterministic Counting Timed Automata (`DCTA`), that can effectively detect the three identified anomaly types present in SCADA communications.

- Utilizing extended network flow records, we develop these `DCTAs` to detect a range of anomalies.

- We assess the performance of DCTA using publicly available datasets and conduct a comparative analysis with previous studies.

We organize the rest of the chapters as follows. In Section 3.2, we discuss previous research regarding the detection of cyber attacks in smart grid networks. In Section 3.3, we propose our method for detecting anomalies using state transition machines. We provide the evaluation findings in Section 3.4. Finally, we conclude this chapter in Section 3.5.

3.2 Related Work

The existing literature for detecting these attacks deals with three main approaches. Machine learning based models utilize algorithms such as decision trees [25], random forests [25], and support vector machines [18] for anomaly detection, but suffer from the need for extensive training data and false alarms. State Transition Models [9] represent smart grid operations as states and transitions, detecting anomalies based on uncommon sequences of moves, although they may overlook timing anomalies. Other Methods include signature-based intrusion detection systems [44], behavior-based models [45], statistical techniques [46, 48] like control charts and regression analysis, and time-series analysis, each with its strengths in identifying deviations from normal behavior. Recent advancements focus on modeling sensor driven communication to detect APTs and malware infections. Overall, while these approaches offer diverse strategies for cyber attack detection in SCADA networks, each has its own set of advantages and limitations.

3.3 Proposed Anomaly Detector

In this section, we describe the proposed state transition model `Deterministic Counting Timed Automata (DCTA)` which detects cyber attacks using network flow records of IEC 60870-5-104 communication. Hence we begin by providing a brief overview of the IEC 60870-5-104 communication protocol, and details of the different attack vectors considered. Subsequently, we formally define the `DCTA` and also show the state transition models designed for detecting these attacks.

3.3.1 IEC 60870-5-104 Communication Background

IEC 60870-5-104 protocol stack is a standard developed by the International Electrotechnical Commission (IEC) for telecontrol of equipment/system over TCP/IP[1]. The standard defines a message format in the form of an Application Protocol Data Unit (APDU) and these messages are used for communication between the master station (SCADA master) and the remote devices (slaves). APDU is made up of a mandatory Application Protocol Control Information (APCI) or an APCI with Application Service Data Unit (ASDU) as shown in Figure 3.1. The control information that is

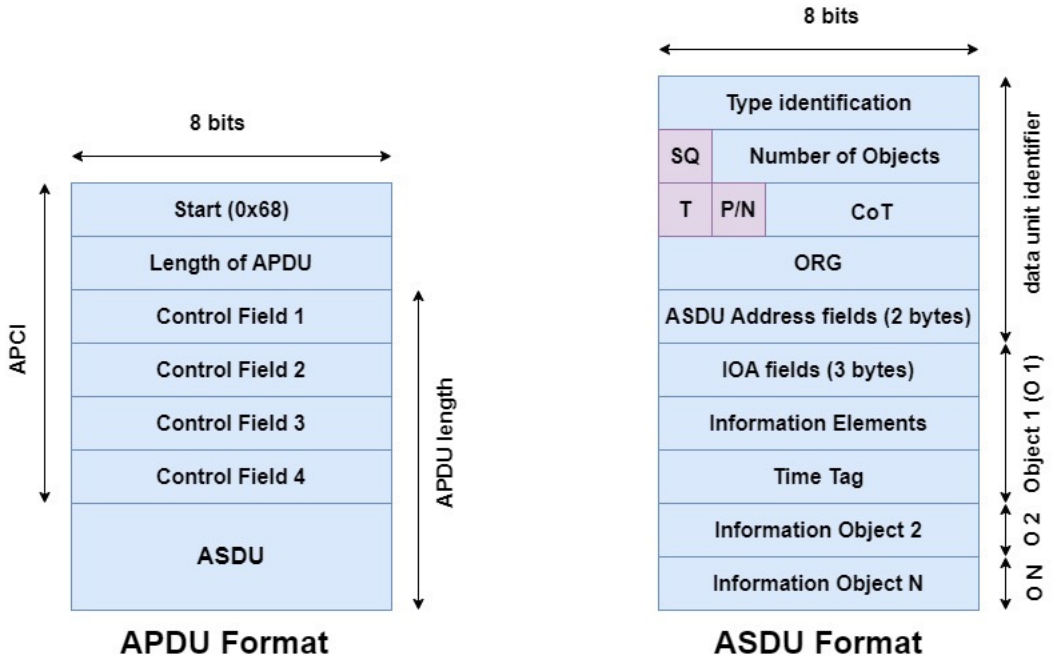


Figure 3.1: APDU Message Format

added to the payload data is referred to as APCI. It consists of functions like start, stop, test, reset, and data transfer which enables session establishment, termination, reliable data transmission, and link integrity checks in the protocol. ASDU consists of data identifier field and data itself. Data identifier field includes $ASDUTYPE$, Number of Information Objects ($NUMIX$) within APDU, Cause of Transmission (CoT), Originator Address (ORG) i.e. sender address, and ASDU address field ($ADDR$).

[1]More details in [3]

$ASDUTYPE$ represents the type of information being transmitted within an ASDU. CoT provides information about the purpose or event that triggered the transmission, allowing the recipient to understand the context or significance of the message. Data is made up of one or more information objects. IOA represents the address of information objects present inside the ASDU. It helps to identify the particular data within a defined station. The values set in the fields of APDU, number of APDU messages, and their order can help identify anomalies. For e.g., if there is a series of messages received with $ASDUTYPE$ 46 and CoT values 6, 7, and 10 by a device, it is possibly an indicator of a switching attack (details in the next subsection).

3.3.2 Attack Vectors

There are several cyber attacks that can be launched against the SCADA infrastructure. These attacks can manifest from different layers of SCADA systems like application interfaces, web interfaces, implementation bugs, etc [57]. However, these cyber attacks are applicable to conventional networks as well and are well studied. We consider a few major cyber attacks which are exclusive to smart grid networks and study their behavioral observations. We also identify three network anomaly types [2]. Identifying these anomalies help in detecting the attacks.

i) Connection-loss: This is an adverse situation in which a network's connection is lost due to external interruptions which include node capture, physical issues like damaged network cables, etc. As SCADA communications are periodic in nature, such interruptions result in missing communications or in general missing flows which is a pattern of failed data transmission from the controller/master to the Remote Terminal Units (RTUs) and the other way around.

ii) Denial of Service: In this attack, an attacker overwhelms the master by sending a large number of messages. This keeps the master busy and may even bring it down disturbing the legitimate communications with the master. Indications of a DoS attack in smart grids include the unresponsiveness of controller/master or RTUs,

[2]There are other types of anomalies that can be seen in SCADA systems. For e.g., measurement anomalies [58].

significant degradation in system performance, and a sudden increase in network traffic or requests.

iii) Injection Attack: In this attack an adversary sends spurious messages/communication to the master/controller possibly using a new device or compromising an existing device in the network. These messages can be of two types (a) Attributes of messages having illegitimate values: Here attributes take values out of their permitted ranges. For e.g., attribute CoT can take values 6, 7, 8, 9, 10, 44, 45, 46, and 47 when attribute $ASDUTYPE$ has values in the range 45-51. If the CoT attribute takes the value 5 when $ASDUTYPE$ is 46, it is illegitimate.

(b) Attributes having legitimate values; but their sequence and numbers are illegitimate: Here all the attributes take legitimate values but their sequencing in a limited span of time can create illegitimate actions.

Indications of an injection attack in smart grids include unexpected or unauthorized commands sent to RTUs, abnormal control actions or system behavior, and anomalies in data readings or sensor outputs.

iv) Rogue Devices: Rogue devices are unauthorized devices that send illegitimate messages to the master as in the case of an injection attack. However, the only difference is; these messages may be sent by known devices in the injection attack. Indications of rogue device communications include the execution of unauthorized commands, abnormal system behavior, and anomalies in data readings.

v) Scanning Attack: In this case, an attacker scans the SCADA network for potential machines/devices to communicate. Horizontal scanning and vertical scanning are two types of scanning techniques. Indications of horizontal scanning attack include an increased volume of network traffic, attempts to identify and connect to devices, and a surge in unauthorized connection requests. Indications of vertical scanning attack include attempts to connect to different Information Object Addresses (IOA) within a device. Every such object within a device is assigned an address and by sending a probe to that address, an attacker is finding if that address is active or in use. Both horizontal and vertical scanning techniques will be probing to either identify devices and addresses valid and also possibly vulnerabilities in communication protocols or

device configuration.

vi) Switching Attack: In this case, an attacker sends a series of messages with specified $ASDUTYPE$ and CoT values to the target that causes the device to turn on and off. Unauthorized changes in the control settings, abnormal switching operations or patterns, and discrepancies between commanded and actual switching states are indications of switching attack.

It is worth noting that, the above list of cyber attacks is not exhaustive. However, these and many other potential cyber attacks fall into one of the following three types of anomalies.

(**i) Single Message Anomaly:** These anomalies are a consequence of illegal/changed values of attributes in a single message. The APDU message can have attributes set to values that are invalid. Some of the rogue communications, and injection attacks fall under this category as they involve messages being sent from unknown/illegitimate sources.

(ii) Message Sequencing Anomaly: These anomalies are caused by message sequencing that are not seen before. For e.g., the master and RTU communications carry messages with specific values set. Every response message should be preceded by a query message. Moreover, there should be a correspondence between the query and response which are usually matched with attribute values. Switching attack falls under this category where messages are sent with toggling mode of operation with a sequence between them.

(iii) Time based Anomaly: These anomalies are caused by the repeated appearance of a large number of messages over a period of time. These messages may possibly have legitimate values set for their attributes. However, their sheer number which is otherwise not seen or absence of these messages indicates an anomaly. DoS attacks fall under this category when the messages sent have legitimate values.

3.3.3 State Transition Model based Detection

Our detection model uses extended network flows for detecting different attacks in the smart grid networks. Corresponding to every attack, there is a state transition machine as shown in Figure 3.2. Network flows are generated from packets (collected from switch/router through which these packets pass through) with additional details as required for detection. Such flows are given as input to all the machines and all of these machines process the flow if it is relevant and take appropriate transitions depending on their current state and transition constraints. An attack is detected if any of these machines move to a state representing an attack.

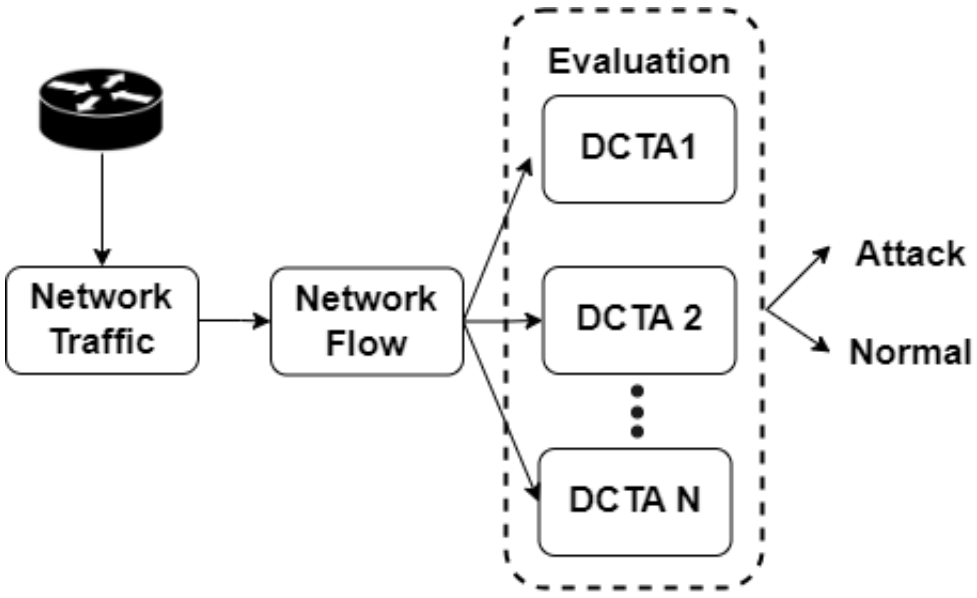


Figure 3.2: Working of Proposed Detection Method

In order to detect the above cyber attacks, we propose a new state transition model. This model uses network communication flows as input and performs state changes to detect the attacks. This new state transition machine, we name as `Deterministic Counting Timed Automata (DCTA)`.

The machine is formally denoted as seven tuple

M=(Q, Σ, C, $\mathcal{T}$, δ, q_0, F) where

Q	A finite set of states
$q_0 \in Q$	An initial state
Σ	A finite set of input symbols corresponding to different messages exchanged in the SCADA system with $m_i \in \Sigma = \{a_{i1}, a_{i2}, \cdots, a_{ik}\}$ are the attributes of m_i
C	A finite set of counter variables
$\mathcal{T}$	A finite set of clock variables
δ	Is a transition function
F	A subset of states ($F \subseteq Q$) which are final states

Each transition $\delta_i \in \delta$[3] is a nine tuple

$\langle q_i, q_j, m_i, \phi(m_i), \phi(t_i), \phi(c_i), Reset(t_j), Inc(c_j), Log(m_i)\rangle$ with the elements representing

$q_i \in Q$	Current State
$q_j \in Q$	Next State
m_i	Input Symbol
$c_i \in C$	Counter value at the state q_i
$\phi(m_i)$	Boolean conjunction of constraints on the attributes of message m_i
$\phi(t_i)$	Timing constraint on the timer at q_i
$\phi(c_i)$	Counter constraint, where c_i is the counter at q_i
$Reset(t_j)$	A function which resets the time variable $t_j \in \mathcal{T}$ at state q_j
$Inc(c_j)$	Is a function that maps the current state of the counter at q_j to a new value
$Log(m_i)$	Is a function that logs the necessary attributes from message m_i.

A value of '$-$' in a transition signifies that the corresponding value or constraint is not applicable to this transition. For example, a transition $(q_1, q_2, m_1, LEN \in [1, 10], t_1 < 3, c_1 > 3, t_2 \leftarrow 0, c_2 \leftarrow 1, -)$ indicates a transition from state q_1 to q_2 upon arrival of a message m_1 and following constraints are met

1) one of the attributes namely length is having a value between 1 and 10,

2) message is received in less than 3 units since the last transition to q_1 and clock reset

[3]Not a total function. However the transitions are deterministic.

timing,

3) counter value at q_1 is greater than 3 when the transition takes place,

4) counter at state q_2 i.e. c_2 is initialized to 1 and a '$-$' at the end indicates nothing from this message m_1 is logged for subsequent usage/reference.

`DCTA` has few states marked as anomaly indicators or final states. Progression to these states through a sequence of moves indicates identified anomalies in the system. These sequence of moves are of the form $(q_o, m_1, q_i), (q_i, m_2, q_j), \cdots, (q_{n-1}, m_n, q_n)$ with $q_n \in F$ (to the anomaly state) also define the language accepted by the respective `DCTA`.

3.3.4 State Transition Models for Different Attacks

Using the state transition model presented previously, we design specific machines for detecting different attacks.

Connection-loss Attack: As mentioned earlier, communication in SCADA systems is usually periodic in nature with master querying the RTU and other measurement units for readings at regular intervals. Thus, a series of such missing flows indicates connection-loss and this needs to be traced through the state transition machine for detecting the attack. In order to detect this attack, we design a STM as a time based anomaly detector. The STM is initialized afresh for every W units of time and it keeps track of number of missing communication flows with the help of a counter within that W time period. Suppose, on an average after every x seconds one flow is received, state transition machine keeps checking after every x seconds (taking into account random network delays) if there is a flow or not. The STM for detecting this attack is a three state machine as shown in Figure 3.3. State q_1 is the initial state and q_2 is the state where counting is done. The machine stays in state q_1 if there is a message m_i (through a network flow) observed as expected; otherwise it takes a transition to q_2, increments missing flow/communication counter at q_2 and returns back to q_1[4]. The transitions from q_1 to q_1 and q_1 to q_2 are having timing constraints to indicate respective time delays. It moves to state q_3 if the counter value at q_2 exceeds

[4]This is an instantaneous transition without any delay.

a threshold which is indicated as a constraint on the transition from q_2 to q_3.

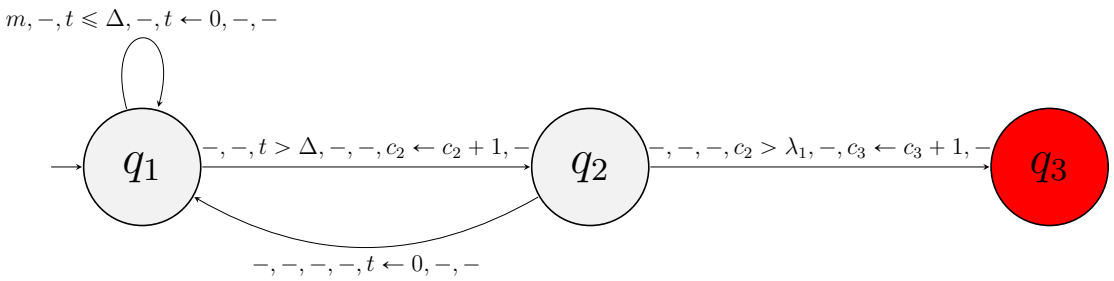


Figure 3.3: DCTA for Detecting Connection-loss Attack

DoS Attack: DoS attack results in a surge of messages/flows from an adversary. We design a time based anomaly detector with two states as shown in Figure 3.4 to detect this attack. The STM is initialized afresh for every W units of time and it keeps track of number of incoming flows within that W time period. If the number of such messages increase beyond a threshold value, the STM declares attack. The state q_1 keeps track of number of flows within W time period by incrementing a counter if there is a message m_i observed as expected. It moves to state q_2 if the counter value at q_1 exceeds a threshold which is indicated as constraint $c_1 \geqslant \lambda_2$ on the transition from q_1 to q_2.

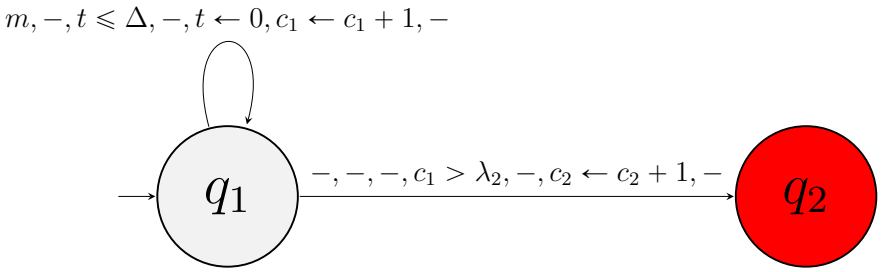


Figure 3.4: DCTA for Detecting DoS Attack

Injection Attack and Rogue Devices: We observe that the actions in injection and rogue device communications are similar where some unauthorized communication or unexpected messages are transferred to the master. The only difference is that

these messages may originate from a legitimate device in the injection attack and from an illegitimate device in the rouge communication. Hence, we have common STMs for detecting these two attacks.

(a) APDU messages having illegal values can be detected with appropriate constraints placed on the message attributes. The STM for detecting these illegal value combinations are shown in Figure 3.5 which is a two state STM with a single transition $\delta 1$ having constraints on message attributes as shown in Table 3.1.

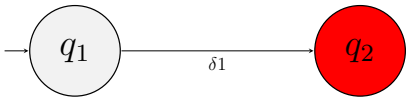


Figure 3.5: DCTA for Detecting Injection/Rogue Device Attack (Single Message Anomaly)

Table 3.1: DCTA Transitions of Figure 3.5 for Injection/Rogue Device Attack (Single Message Anomaly) Detection

Transition	Current State	Next State	Input Symbol	$\phi(m)$	$\phi(t)$	$\phi(c)$	Reset(t)	Inc(c)	Log(m)
$\delta 1$	q_1	q_2	m (APDU)	$srcPort, dstPort \notin [0, 65535]\|\|\|$ $LEN \notin [0, 255]\|\|\|FMT \notin \{0x0, 0x1, 0x3\}\|\|$ $ASDUTYPE \notin [1, 135]\|\|\|NUMIX \notin [0, 127]\|\|\|$ $CoT \notin [1, 47]\|\|\|ADDR \notin [1, 65535]\|\|\|$ $IOA \notin [0, 65535]\|\|\|$ $(ASDUTYPE = 35\&\& \sim (CoT = 3\|\|CoT = 5))...$	-	-	-	-	-

For e.g., DCTA has a constraint that the length of APDU (LEN) should be within the range 0-255 on this transition from q_1 to q_2. It is easy to see that this STM can detect single message anomaly type[5].

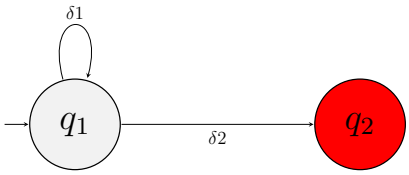


Figure 3.6: DCTA for Detecting Sequence based Injection Attack and Rogue Devices

[5]As there are many combinations of values, we show a few representative constraints here and omit the rest of the details for the sake of brevity.

(b) Second category of anomalies can be detected by looking at the sequence of messages over a very short period of time. Hence, we design a sequence based anomaly detector (STM) with appropriate constraints on transitions as shown in Figure 3.6.

The transitions in this STM have constraints on the message type, their values, timing, etc. as shown in Table 3.2. Here, STM keeps track of legitimate devices as well as values of different attributes of messages by logging these using function $Log(m)$.

Table 3.2: `DCTA` Transitions of Figure 3.6 for Sequence based Injection Attack and Rogue Devices Detection

Transition	**Current State**	**Next State**	**Input Symbol**	$\phi(m)$	$\phi(t)$	$\phi(c)$	**Reset(t)**	**Inc(c)**	**Log(m)**
$\delta 1$	q_1	q_1	m (APDU)	$srcIP, dstIP \in \text{Log}(m)$ && $\sim (\{ASDUTYPE = 45 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 47 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 48 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 49 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 50 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 51 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|)$	-	-	-	-	Log(m)
$\delta 2$	q_1	q_2	m (APDU)	$srcIP, dstIP \notin \text{Log}(m)\|\|$ $\{ASDUTYPE = 45 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 47 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 48 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 49 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 50 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$ $\{ASDUTYPE = 51 \text{ \&\& } CoT = \{6,7\}\} \in \text{Log}(m)\|\|$	-	-	-	-	-

Message constraint imposed on transitions $\delta 1$ and $\delta 2$ is a collection of constraints on various attributes of a message. A control command $ASDUTYPE$ takes values 45, 47, 48, 49, 50, and 51 except for the double point control command ($ASDUTYPE$=46) which is used to switch on/off a device. Moreover, these messages should come from a legitimate device (known IP address). Thus, the message constraint imposed on transition $\delta 1$ denotes that only legitimate device IP addresses should be used. In addition to this, no such message should appear in which the control command activation and confirmation are happening sequentially within a limited amount of time. Similarly, the constraint on transition $\delta 2$ denotes that if control command activation and confirmation are done sequentially within a limited amount of time then it represents an anomaly. Whenever devices apart from the legitimate ones try to communicate; then it gets classified as a rogue device.

Scanning Attack: In the case of a scanning attack, the attacker sends specific mes-

sages to scan the network to find active devices. If any device is active, then the scan yields a response in the form of test frame confirmation. In order to detect this attack, we designed a STM as a time based anomaly detector with two states as shown in Figure 3.7.

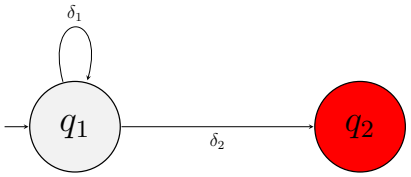


Figure 3.7: DCTA for Detecting for Scanning Attack

Table 3.3: DCTA Transitions of Figure 3.7 for Horizontal Scanning Detection

Transition	**Current State**	**Next State**	**Input Symbol**	$\phi(m)$	$\phi(t)$	$\phi(c)$	**Reset(t)**	**Inc(c)**	**Log(m)**
$\delta 1$	q_1	q_1	m (APDU)	$dstIP \notin \text{Log}(m)$	$t \leqslant \Delta$	-	$t \leftarrow 0$	$c_1 \leftarrow c_1 + 1$	Log(m)
$\delta 2$	q_1	q_2	m (APDU)	-	-	$c_1 > \lambda_3$	-	$c_2 \leftarrow c_2 + 1$	-

Table 3.4: DCTA Transitions of Figure 3.7 for Vertical Scanning Detection

Transition	**Current State**	**Next State**	**Input Symbol**	$\phi(m)$	$\phi(t)$	$\phi(c)$	**Reset(t)**	**Inc(c)**	**Log(m)**
$\delta 1$	q_1	q_1	m (APDU)	$IOA \notin \text{Log}(m)$	$t \leqslant \Delta$	-	$t \leftarrow 0$	$c_1 \leftarrow c_1 + 1$	Log(m)
$\delta 2$	q_1	q_2	m (APDU)	-	-	$c_1 > \lambda_3$	-	$c_2 \leftarrow c_2 + 1$	-

The STM is initialized afresh for every W units of time and it keeps track of the number of messages sent to different devices or different objects within a device. If the number of such messages increases beyond a threshold value, the STM detects an attack [6]. The state q_1 keeps track of the number of such messages sent for scanning within W time period by incrementing a counter if there is a message m_i observed such that it is addressed to a different device (different IP address in case of a horizontal scanning) or different objects (different Information Object Addresses) within a device (vertical scanning). IP addresses of devices that received messages and also the Information Object Addresses of objects within a device are tracked by logging the details using function $Log(m)$. It moves to state q_2 if the counter value at q_1 exceeds

[6]It will keep track of only either of these two

a threshold which is indicated as constraint $c_1 \geqslant \lambda_3$ on the transition from q_1 to q_2. The transitions corresponding to this STM are shown in Table 3.3 and 3.4.

Switching Attack: In switching attack, an attacker sends multiple messages with $ASDUTYPE$ value 46 and changes the CoT values to 6, 7, and 10. Thus, detecting this attack requires analyzing a sequence of messages for these changes. In order to capture these changes, we designed a STM as a time based anomaly detector with two states as shown in Figure 3.8. The transition from the state q_1 to q_1 keeps track of different attribute values (CoT and $ASDUTYPE$ extracted from messages) by logging them using function $Log(m)$. It moves to state q_2 if the earlier logged values (within a short window period) contain CoT value 6, 7, 10 and $ASDUTYPE$ value 46, as it indicates acceptance of switching command immediately which can bypass normal control system logic and can override established protection schemes. The transitions corresponding to this STM are shown in Table 3.5.

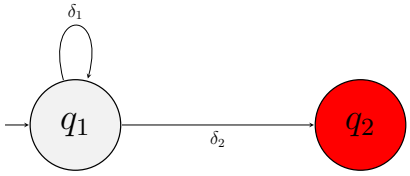


Figure 3.8: `DCTA` for Detecting Switching Attack

Table 3.5: `DCTA` Transitions of Figure 3.8 for Switching Attack Detection

Transition	Current State	Next State	Input Symbol	$\phi(m)$	$\phi(t)$	$\phi(c)$	Reset(t)	Inc(c)	Log(m)
δ_1	q_1	q_1	m (APDU)	-	$t \leqslant \Delta$	-	$t \leftarrow 0$	-	Log(m)
δ_2	q_1	q_2	m (APDU)	$\{ASDUTYPE = 46$ && $CoT = \{6, 7, 10\}\} \in \text{Log}(m)$	-	-	-	-	-

Attack Detection with `DCTA`s an Example: Consider that a fresh window starts at w_1 and lasts upto w_2. All the packets collected from a router/switch in this period are used to construct flows with necessary additions. Lets assume an attacker with a valid IP address is trying to perform scanning attack by sending legitimate packets but with different IOA values to check for an active victim. It sends IEC-104 packets featuring an Information Object Address (IOA) denoted as x_1. Flow generation will generate a flow corresponding to this which will eventually fed to all `DCTA`s. In response, all

DCTAs governing distinct security aspects react concurrently. The DCTA of connection-loss attack reverts to its initial state (q_1 to q_1), recognizing the presence of the incoming packet. Similarly, the DCTA monitoring rogue and injection activities maintain its initial state, as the packet is validated as legitimate (IP address is valid). The Denial of Service (DoS) DCTA increments its counter, acknowledging the presence of a fresh packet. The DCTA, designed for detecting scanning activities, logs the IOA value if the value is absent in the log file otherwise it increments its counter only. The Switching DCTA, checking for specific CoT and $ASDUTYPE$ values remains in the initial state as there are no values corresponding to this in the flow/packet. In the event of receiving a sequence of packets with diverse IOA values (prior to this packet/flow which are logged previously), the scanning DCTA may transition to a state indicating an attack if the counter surpasses a predefined threshold. Subsequently, at the end of each window period, a reset occurs across all DCTAs, ensuring a clean slate (including clearing logged values) for subsequent monitoring and analysis.

3.4 Experiments and Evaluation

In this section, we provide the details of experiments done to evaluate the detection performance of DCTA based anomaly detectors. We provide the details of evaluation results and sensitivity analysis in the following two subsections.

3.4.1 Evaluation

Here we present the details of the dataset used, STM implementation details, comparison with prior works, evaluation metrics and evaluation results respectively.

Dataset: For our experiments and evaluation, we used a publicly available dataset [59], namely but-iec104-i collected from a smart grid testbed at Brno University of Technology. This dataset is a processed collection of network flow records (in csv format) of IEC-104 communications. These flow records are extended IPFIX flows with a few custom fields added which indicate the type of IEC messages that appeared in the corresponding flow. These flows are collected from a network device that observes

the network traffic and generates IPFIX records and exported to a flow collector. Subsequently, these flows are processed and annotated after custom field additions. The records include fields like timestamps, IP addresses, port numbers of communicating devices, and also a few header details taken from IEC-104 messages, etc. The dataset consists of normal communication and also six types of cyber attacks generated at different time intervals. Table 3.6 shows the details of this dataset.

Table 3.6: Overview of Dataset

Type	**Duration**	**Flow Count**
Normal-traffic	67 hours 55 minutes	58,930
Connection-loss	67 hours 55 minutes	57,863
DoS-attack	67 hours 55 minutes	58,932
Injection-attack	67 hours 55 minutes	58,930
Rogue-device	67 hours 55 minutes	58,893
Scanning-attack	67 hours 55 minutes	58,927
Switching-attack	67 hours 55 minutes	59,002

The dataset consists of seven files each of a duration of 67 hours and 55 minutes. One of them is of normal communication and the other six correspond to an attack type. The table also shows the number of flow records in each of these files.

Implementation: We implemented `DCTA`s for detecting different anomalies/attacks in Python programming language. STMs have two main components namely states and transitions. In our implementation, we represented all the states with variables, and the current state is remembered by storing the state name in a separate variable. Transitions of a `DCTA` are implemented through a function named `transition()` which takes the current state and different constraints as input. We enlisted the different constraints (message attribute, timing, and counter values) that cause the transition from one state to another. For every state, we defined all of the possible transitions and the corresponding next state of the machine. In the function `transition()`, we implemented logic to evaluate the different constraints with *if-then-else* ladder to

make an appropriate state change in STM. For every state, all the transitions from that state are attempted to check if a move is possible. At the end of every transition, the current state value is updated. It is also possible that the current state is same as the previous state as some of the STMs have self loops.

Comparison with Prior Work: We compare the performance of our proposed `DCTA` based anomaly detector with two recent works namely of Matoušek et al. [7] which described two different types of STMs to detect anomalies and also with the work of Hadir et al. [1] which uses a machine learning algorithm for detecting different attacks. First one identifies two types of anomalies.

1) Single Conversation Reasoning- Using an STM constructed from the sequence of messages, it identifies transitions which were never seen before (i.e. the traces from which the STM is generated). Such transitions are detected as anomalies.

2) Comparing the Probability Distributions- All conversation sequences (message sequences in the trace) are used to generate `DPA/Prefix Tree` type STMs and subsequently these STMs are used to derive frequency values of individual transitions. These transition frequency values are represented as probability distributions indicating the likelihood of a particular transition. Subsequently, the probability distributions of five minute window periods from the training and testing intervals are compared with Euclidean distance. If the distance is high, then an attack is detected.

On the other hand, Hadir et al. [1] use hierarchical multilayered perceptron based ML algorithm to improve detection performance. The HMLP architecture has two layers: the first classifies the test data as benign or malicious, while the second identifies attack types. This approach enhances detection rates by isolating attack samples during training, reducing data imbalance effects. For training the HMLP model, all the data of dataset [59], including attacks and benign samples were merged together, with unified attack labels for binary classification. After training, robustness against different attacks is evaluated to measure model resilience.

Implementation for both the methods [7] and [1] are available and results reported here are based on their implementation.

Evaluation Metrics: We evaluate the performance of the proposed `DCTA`

based anomaly detector with five parameters namely `Detection Rate (DR)`, `False Positive Rate (FP)`, `Accuracy (Acc)`, `F1 score (F1)` and `Matthews Correlation Coefficient` (`MCC`). Equations 3.1 to 3.6 show the details of specific calculations for each metric.

$$\text{Detection Rate} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.1}$$

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.2}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.4}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP}) \times (\text{TP} + \text{FN}) \times (\text{TN} + \text{FP}) \times (\text{TN} + \text{FN})}} \tag{3.6}$$

In these equations, True Positive (TP) represents instances where attacks are correctly detected as attacks, False Negative (FN) denotes instances where attacks are incorrectly classified as benign, True Negative (TN) signifies instances where benign activities are correctly identified as such, and False Positive (FP) indicates instances where benign activities are incorrectly flagged as attacks.

Further, it is worth noting that we use few thresholds as constraints for transitions of `DCTA`. Few of these transitions lead to states representing attacks/anomalies. The parameters and corresponding threshold values are shown in Table 3.7 which we denote as golden thresholds.

Results: We present the results of the evaluation in two parts.

`(i) Grand Experiment:` Here we assess the performance on the combined dataset where all seven files (one normal and six attack types) are merged. For those attack types that are time based anomalies, we used a window time of five minutes. Entire dataset was given as input to both `DCTA` based STMs, also the two STMs of [7] and to the Hierarchical Multilayer Perceptron (HMLP) method to evaluate the performance. Table 3.8. shows the results of this experiment with different metrics. We

Table 3.7: Golden Threshold Values

Parameter	**Threshold**
λ_1	30
λ_2	45
λ_3	27
Δ	8

can notice that `DCTA` exhibits far better performance with 100% detection rate, nearly 100% accuracy, and F1 scores while both the methods of prior work have performed comparatively poorly. The low detection rate of `Prefix Tree` and `DPA` based learning methods is due to their inability in detecting certain attack types (details in the next part).

Table 3.8: Detection Performance Comparison of `DCTA` with Prior Works on Grand Dataset

Technique	**Testing Intervals**	**DR**	**FP**	**Acc**	**F1**
DCTA	5705	100%	0.017%	99.98%	99.29%
DPA	5705	47.14%	0%	99.35%	64.08%
Prefix Tree	5705	47.14%	0%	99.35%	64.08%
HMLP	5705	100%	0.6%	99.4%	94.92%

`(ii) Individual Experiment:` In the second part, we performed an evaluation with individual attack types combined with normal communication flows and compared it with only [7] as this is STM based model. Thus there are six files each having 1630 intervals of five minutes duration. The objective of this evaluation is to assess how good the detection is for different attacks. For this experiment also we used the same golden threshold values shown in Table 3.7. Table 3.9 shows the results obtained with `DCTA`, `DPA`, and `Prefix Tree` models. The table shows the confusion matrices for each experiment along with the detection rate, false positive rate, accuracy, and F1 score, and MCC values. It is worth noting from Table 3.9 that the `DPA` and `Prefix`

Table 3.9: Detection Performance Comparison of DCTA with Prior Works on Individual Attack Types

Technique	Testing Intervals	Confusion matrix			DR	FP	Acc	F1	MCC
DCTA	1630		**Normal**	**Connection-loss**	100%	0%	100%	100%	1
		Normal	1616	0					
		Connection-loss	0	14					
DCTA	1630		**Normal**	**DoS**	100%	0%	100%	100%	1
		Normal	1595	0					
		DoS	0	35					
DCTA	1630		**Normal**	**Scanning**	100%	0%	100%	100%	1
		Normal	1623	0					
		Scanning	0	7					
DCTA	1630		**Normal**	**Switching**	100%	0.061%	99.94%	80%	0.82
		Normal	1627	1					
		Switching	0	2					
DCTA	1630		**Normal**	**Injection**	100%	0%	100%	100%	1
		Normal	1624	0					
		Injection	0	6					
DCTA	1630		**Normal**	**Rogue**	100%	0%	100%	100%	1
		Normal	1624	0					
		Rogue	0	6					
Alergia	1630		**Normal**	**Connection-loss**	85.71%	0%	99.87%	92.30%	0.92
		Normal	1616	0					
		Connection-loss	2	12					
Alergia	1630		**Normal**	**DoS**	0%	0%	97.85%	0%	1
		Normal	1595	0					
		DoS	35	0					
Alergia	1630		**Normal**	**Scanning**	100%	0%	100%	100%	1
		Normal	1623	0					
		Scanning	0	7					
Alergia	1630		**Normal**	**Switching**	100%		100%	100%	1
		Normal	1628	0					
		Switching	0	2					
Alergia	1630		**Normal**	**Injection**	100%	0%	100%	100%	1
		Normal	1624	0					
		Injection	0	6					
Alergia	1630		**Normal**	**Rogue**	100%	0%	100%	100%	1
		Normal	1624	0					
		Rogue	0	6					
Prefix Tree	1630		**Normal**	**Connection-loss**	85.71%	0%	99.87%	92.30%	0.92
		Normal	1616	0					
		Connection-loss	2	12					
Prefix Tree	1630		**Normal**	**DoS**	0%	0%	97.85%	0%	0%
		Normal	1595	0					
		DoS	35	0					
Prefix Tree	1630		**Normal**	**Scanning**	100%	0%	100%	100%	1
		Normal	1623	0					
		Scanning	0	7					
Prefix Tree	1630		**Normal**	**Switching**	100%	0%	100%	100%	1
		Normal	1628	0					
		Switching	0	2					
Prefix Tree	1630		**Normal**	**Injection**	100%	0%	100%	100%	1
		Normal	1624	0					
		Injection	0	6					
Prefix Tree	1630		**Normal**	**Rogue**	100%	0%	100%	100%	1
		Normal	1624	0					
		Rogue	0	6					

`Tree` based models were unable to detect DoS attack as the messages sent during DoS attack are also legitimate ones. The probability/frequency of these transitions are well within the limit and hence are not detectable. However, `DCTA` is able to detect all those attacks by keeping track of number of continuous messages sent, their order, timing, and comparison with previous message attribute values as defined in their respective transitions. Further, `DCTA` based detection does not classify one attack type as other one. However, HMLP [1] based detection has this issue as shown in Table 3.10. We can notice from the table that several attacks of one type are being detected as other type. However, this does not alter the overall detection rate as long as it is detected as attack. Based on the considered cyber attacks, we also tabulate the detection ability of different STM based detection methods. Table 3.11 shows this comparison. In the table, Single refers to single conversation reasoning while $Distr_{Prefix\ Tree}$ and $Distr_{DPA}$ represent the probability distribution based anomaly detectors as defined in [7]. We can notice that `DCTA` is able to detect all the attack types while the STMs of [7] fail to detect a few attack types. In contrast, our proposed anomaly categorization aims to be comprehensive and can potentially detect different types of attacks.

Table 3.10: Detection Performance of HMLP [1]

	Benign	**Connection-loss**	**DoS**	**Switching**	**Scanning**	**Rogue**	**Injection**
Benign	199814	0	278	650		0	350
Connection-loss	0	3033	0	0	0	0	0
DoS	0	0	1243	0	118	0	79
Switching	0	0	0	1511	0	0	0
Scanning	0	0	7	0	3006	0	0
Rogue	0	2	0	0	0	1562	0
Injection	0	0	0	0	0	0	1517

Table 3.11: Detection Ability Comparison of DCTA

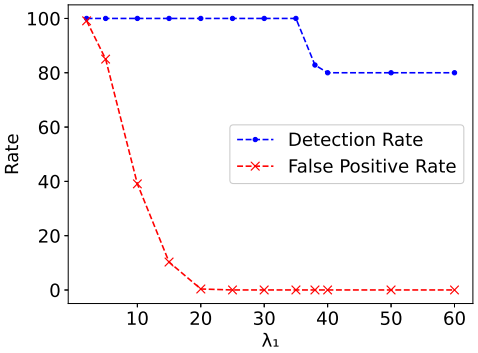
Anomaly	**DCTA**	**Single**	$Distr_{Prefix\ Tree}$	$Distr_{DPA}$
Connection-loss Attack	✓	✗	✓	✓
DoS Attack	✓	✗	✗	✗
Scanning Attack	✓	✓	✓	✓
Switching Attack	✓	✓	✓	✓
Rogue Device Attack	✓	✓	✓	✓
Injection Attack	✓	✓	✓	✓

3.4.2 Sensitivity Analysis

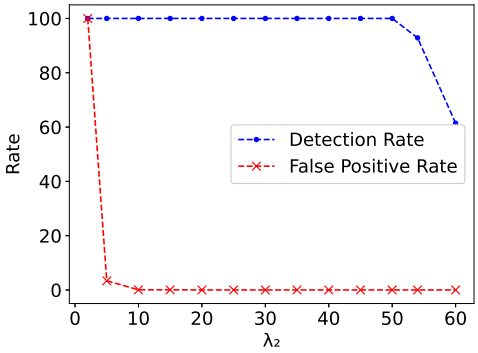
It is worth noting that, several transitions of the proposed DCTA models use thresholds for counter values at different states and also timing constraints. Thus, the detection performance will depend on these threshold values. In order to assess the performance variance with these threshold values, we perform the sensitivity analysis of two parameters namely counter values and timing constraints.

There are three attack types namely connection-loss, DoS, and scanning whose DCTAs have constraints on counter values namely $\lambda_1, \lambda_2, \lambda_3$ as shown earlier. Thus we assess the performance variation of respective DCTAs to variations in these thresholds using grand dataset. For this evaluation, we only varied one parameter value (λ_i) and kept all other parameters as in the grand experiment evaluation discussed above. The details of performance variation for different attacks are as below.

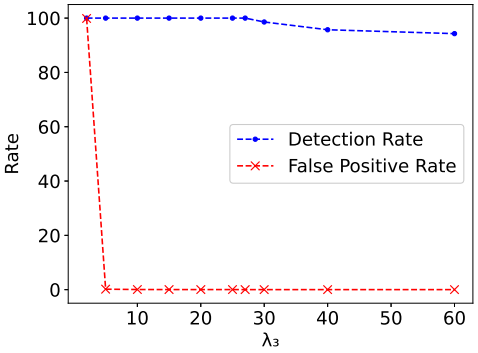
`Connection-loss:` The DCTA of connection-loss detection has λ_1 as a constraint on the transition from q_2 to q_3. The golden value of λ_1 is 30 which yielded 100% detection and 0% false positives. We varied the value of λ_1 in step size of 10 from 0 to 60. Figure 3.9a shows the performance variation in detection rate and false positive rates of this experiment. We can observe from the figure that for a threshold value around 20, false positives are completely reduced to 0% and the detection rate is maintained at 100% till the threshold value crosses 30. Hence a value in this range for λ_1 will yield better performance.



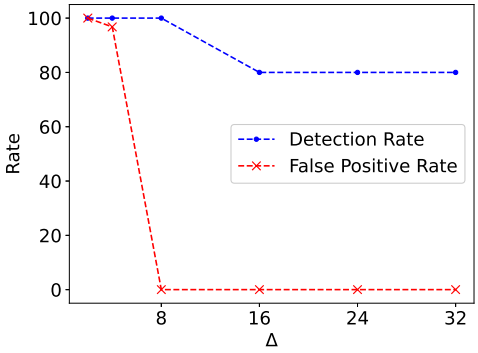
a Detection Performance Across λ_1 Values



b Detection Performance Across λ_2 Values



c Detection Performance Across λ_3 Values



d Detection Performance Across Δ Values

Figure 3.9: Detection Performance Variation with Different Thresholds

`Denial of Service:` The `DCTA` designed for detecting DoS attack has a counter threshold λ_2 on the transition from state q_1 to q_2 whose golden value is 45. Similar to the previous case, here also we varied the value of λ_2 in step size of 10 from 0 to 60. Figure 3.9b displays the change in detection rate and false positive rate with respect to change in the value of λ_2. We can notice from this figure that the detection rate is 100% until the threshold crosses 45 and beyond which it falls. Further, the false positive rates fall to 0% for a value beyond 10.

`Scanning:` The `DCTA` of scanning attack has a threshold λ_3 on the transition between q_1 and q_2 whose golden threshold is 27. Similar to the above two cases, we varied this value in step size of 10 and calculated the detection rate and false positive rates. Figure 3.9c shows the variation in these values. We can notice that the detection rate is 100% till this threshold crosses 27 and the false positive rate falls to 0% to a threshold value beyond 5.

(ii) Timing Values: Several transitions of different `DCTA`s have timing constraints Δ. Timing constraint control when a transition is enabled. Smaller values may trigger transitions prematurely; while too large values can delay both transition and anomaly detection. In order to assess the performance variation of detection, we performed an evaluation by varying the Δ values in step size of 8. Figure 3.9d shows the variation in detection rate and false positive rates for the connection-loss attack[7] on the grand dataset. It can be observed from the figure that too small values of Δ generate a large number of false positives (due to premature transitions), while large values reduce false positives but may decrease detection performance.

3.5 Conclusion

Smart grid is a critical infrastructure and it uses ICT for exchanging safety critical information and hence is vulnerable to different cyber attacks. In this chapter, we presented three types of generic anomaly types namely single message based, message sequence based, and time based anomalies that help in detecting these attacks. In

[7]Similar results were obtained for other `DCTA`'s and we do not show them here for space constraints.

order to detect these anomalies (attacks), we described a new state transition based model DCTA which takes extended network flow records as input and makes permitted transitions. The DCTA transitions have different types of constraints that help verify several attribute values, sequence, timing, number of messages, etc. to detect attacks. We evaluated the performance of our proposed approach on a publicly available dataset and benchmarked its performance against the prior works to show that it can detect different types of attacks with a very good detection rate of 100% in the best case. It can also detect attack types which are not detected by other STM based detectors. Further, the anomaly types presented provide a guideline or template for creating DCTAs for new attack types.

DCTA based detection shares a common limitation with other STM based methods which is the requirement of a specific STM for every attack type. In addition, our work and evaluation at present is limited to only application layer attacks and as such deals with only IEC-104 protocol. However, it will be useful to consider other protocols like Modbus, DNP, etc., and explore if similar models can be developed. Further, other attack types like layer4, layer3, and layer2 of TCP/IP stack can also be combined for evaluation as they will also have an impact on the smart grid operation.

Chapter 4

Detecting Flooding and Bruteforce Attacks in Smart Grid Networks with Probability Distribution Comparison

4.1 Introduction

In the previous chapter, we presented a state transition model to detect different cyber attacks in smart grid networks. However, probability distribution based methods offer scalability, adaptability, reduced memory needs, and faster processing, for detecting attacks compared to state transition machines particularly those involving sudden surges in packets. We observe that normal communication in a smart grid maintains a balance between query and response messages. In Modbus communication, every query has a response. We utilized this observation to detect flooding and bruteforce attacks by identifying deviations in this message sequence. In this chapter, we propose a lightweight method based on probability distribution comparison to detect flooding and bruteforce attacks. We make the following meaningful contributions in this chapter.

- We utilize a technique to create probability distributions by analyzing various messages within Modbus communications.
- We compare the probability distribution derived from normal communications with that of a testing interval to detect irregularities in communication patterns.
- We conduct experiments using two publicly accessible datasets to identify flooding and bruteforce attacks directed at smart grid networks.

The remaining chapter is organized as follows. In Section 4.2, we quickly review the prior works related to cyber attack detection in smart grid networks particularly using Modbus communication. In Section 4.3, we provide the details of the proposed anomaly detector for detecting anomalies in smart grid networks. We present the evaluation results in Section 4.4. Finally, the chapter is concluded in Section 4.5.

4.2 Related Work

Various methods have been employed for cyber attack detection in smart grid and SCADA systems. These methods can be categorized into three main types. Firstly, machine learning techniques [27, 26] utilize features derived from measurement values obtained from Remote Terminal Units (RTUs) or network packet/flow details. These features are fed into classification or clustering algorithms such as decision trees, random forests, support vector machines, and deep learning models to identify different types of attacks. Secondly, state transition modeling [42, 43] involves representing the normal operation and anomalies in communications happening in the grid using state transition machines. This approach detects anomalies by observing unobserved messages or differences in message sequences, utilizing models like Deterministic Finite Automata, Discrete Time Markov Chains, and probabilistic models. However, these models may fail to detect some stealthy attacks. Finally, other methods for anomaly detection include statistical analysis [47], time series analysis [50], and behavior-based models [45]. These approaches monitor various metrics, analyze data over time, and learn the normal behavior of the SCADA system to detect deviations indicative of

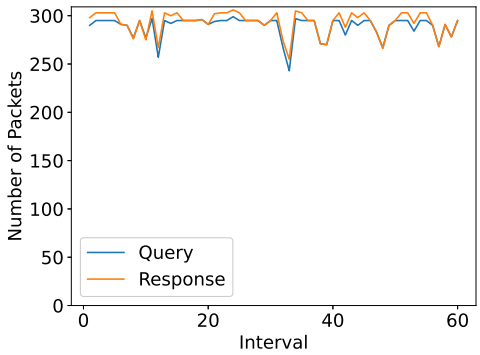
attacks. While effective for detecting novel attacks, these methods may require substantial training data. Overall, these diverse approaches contribute to enhancing the security of smart grid and SCADA systems against cyber threats.

4.3 Proposed Detection Method

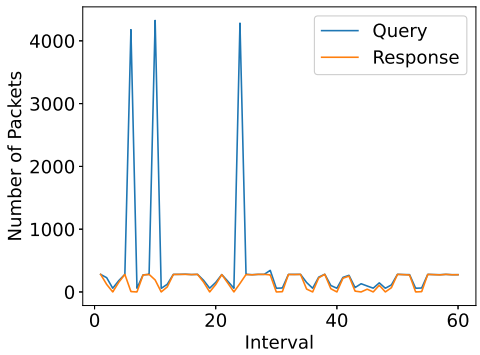
In this section, we present an anomaly detection technique that involves comparing two probability distributions to identify potential attacks. We start by providing a design rationale for this approach and then provide details of how the anomaly detector works.

4.3.1 Design Rationale

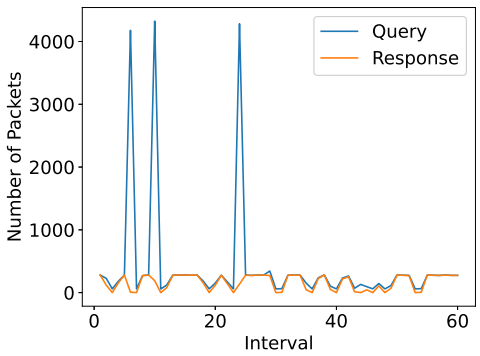
Our method for detecting anomalies is based on the predictable and synchronized nature of Modbus communications. In Modbus systems, which use a master-slave setup, each response from a remote terminal unit (RTU) is usually preceded by a query message from the master device. This means that, under typical conditions, the quantity of queries is approximately equivalent to the quantity of responses, and they occur in sync. In Figure 4.1a., we present a graph showing the number of queries and responses gathered from a dataset of smart grid communications (referred to as Dataset-1, explained in Section 4.4.1). This dataset covers a span of 5 hours, divided into 60 intervals of five minutes each, representing typical communication activity in the smart grid system. Upon examining the graph, we notice that although there are slightly more response messages than query messages, they follow a synchronized pattern. Whenever there's an increase in the number of queries, there's also a corresponding increase in responses, and vice versa. This synchronized behavior reflects the usual flow of communication in Modbus systems within smart grid setups. However, this balance is disrupted during attacks. In Figure 4.1b., we illustrate a period where the smart grid system faces a query flooding attack. During this attack, there is a sudden increase in the number of queries sent to the system. But, despite the surge in queries, there's a notable absence of corresponding response messages, indicating



a Normal (Dataset-1)



b Query Flooding Attack (Dataset-1)



c Bruteforce Attack (Dataset-1)

Figure 4.1: Modbus Query and Response Message Correlation

a disruption in the normal communication flow. Similarly, Figure 4.1c shows another scenario where the smart grid system encounters a bruteforce attack. Here, we observe a significant rise in the number of queries directed at the system, but no corresponding increase in response messages. This discrepancy suggests that the system is flooded by a lot of queries, possibly signaling malicious attempts to breach its security. These observations emphasize the importance of monitoring communication patterns in smart grid systems to detect and address potential cyber threats. By identifying deviations from normal behavior, such as those caused by query flooding or bruteforce attacks, numerous steps can be taken to protect the smart grid infrastructure from potential harm. After analyzing the information gathered in prior research [9], we utilize this observed behavior to develop an anomaly detector tailored to recognize flooding and bruteforce attacks within Modbus communications.

4.3.2 Anomaly Detector

We visualize the exchange of queries and responses in smart grid SCADA networks by creating a probability distribution that reflects the occurrence of each message type. This distribution, derived from typical Modbus communication patterns, serves as our reference point. When we examine communication during testing, we compare it to this baseline profile. We do this by analyzing how different the actual distribution is from the expected one. By calculating the distance or dissimilarity between the two distributions, we can detect any irregularities or deviations from the baseline profile, helping us identify potential anomalies such as flooding or bruteforce attacks.

Let's consider two probability distributions, represented as $X = X_1, X_2, \cdots, X_N$ and $Y = Y_1, Y_2, \cdots, Y_N$, each consisting of N components. Here, each X_i and Y_i represents the probability value associated with a particular event, making X and Y N-dimensional vectors. The vector X is constructed using probability values derived from normal Modbus communication, serving as our reference. On the other hand, the vector Y is built from a testing period, reflecting actual communication behavior during that interval. To measure the dissimilarity between X and Y, we employ a distance metric. Various metrics are available for this purpose, but we opt for the

Hellinger distance, denoted by Equation 4.1, due to its theoretical properties, which make it particularly suitable for our purposes.

$$d_H = \left(\frac{1}{2} \sum_{i=1}^{N} \left(\sqrt{X_i} - \sqrt{Y_i} \right)^2 \right)^{\frac{1}{2}} \tag{4.1}$$

Following are a few properties of Hellinger distance which makes it a better choice for comparing two distributions.

- Lightweight Computation: The Hellinger distance offers a lightweight computation advantage over metrics like the Mahalanobis distance. Unlike the Mahalanobis distance, which requires complex calculations such as matrix inverse or covariance, computing the Hellinger distance between two probability distributions is simpler and less computationally intensive. This makes it a more efficient option for models aiming to detect and address flooding and bruteforce attacks.

- Natural Lower and Upper Bounds: It is important to highlight that the Hellinger distance, denoted as d_H, always falls within the range of 0 to 1. In this scale, 0 indicates perfect similarity between the probability distributions X and Y, while 1 represents the maximum dissimilarity between them. This means that the Hellinger distance inherently possesses natural lower and upper bounds, unlike other distance measurement methods. This characteristic makes it unique and useful for assessing the degree of similarity or dissimilarity between two distributions.

- Yielding Finite Distance Value: Unlike the Kullback-Leibler Divergence, which requires specific dependencies between the probability distributions X and Y, the Hellinger distance does not have such restrictions. Specifically, the Hellinger distance remains defined regardless of whether certain probabilities in Y become zero. For instance, during a flooding attack, probabilities of certain events may drop to zero (Query event drops to zero while response flooding attack and Response event drops to zero while query flooding attack), leading to undefined

values in the Kullback-Leibler Divergence. However, the Hellinger distance provides a finite value within the range of 0 to 1, making it a more suitable measure in such scenarios.

The Hellinger distance metric provides distance values within the range of 0 to 1 and offers a straightforward computation process. In our scenario, we represent the vectors as $X = Prob_{Que}, Prob_{Res}$ and $Y = Prob_{Que}, Prob_{Res}$. Here, X is derived from a training period, and Y is obtained from a testing interval.

During the training phase, we collect samples over a duration of t intervals. From these samples, we generate a training profile (distribution X) using the formulas presented in Equation 4.2 and Equation 4.3. In these equations, RC_i and RS_i represent the counts of requests (queries) and responses in the i^{th} interval respectively. This process allows us to establish a baseline distribution based on the training data, which serves as a reference for comparison during subsequent testing intervals.

$$Prob_{Que} = X_1 = \frac{\sum_{i=1}^{t} RC_i}{\sum_{i=1}^{t} RC_i + \sum_{i=1}^{t} RS_i} \tag{4.2}$$

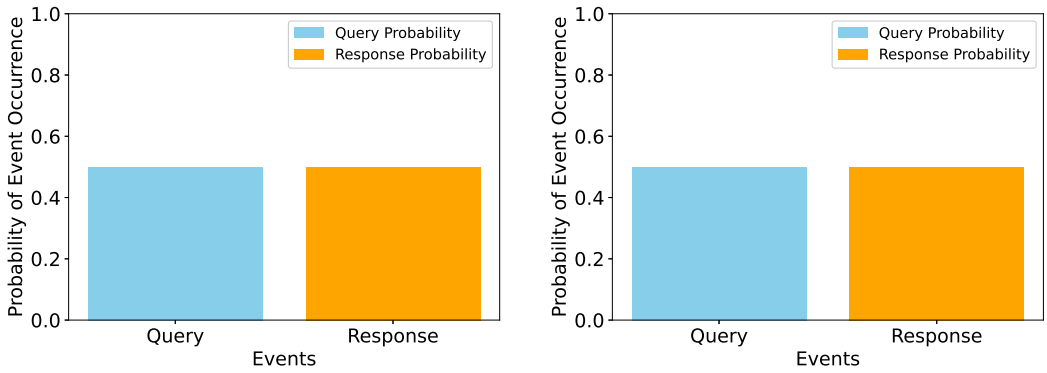
$$Prob_{Res} = X_2 = \frac{\sum_{i=1}^{t} RS_i}{\sum_{i=1}^{t} RC_i + \sum_{i=1}^{t} RS_i} \tag{4.3}$$

$$Prob_{Que} = Y_1 = \frac{RC_i}{RC_i + RS_i} \tag{4.4}$$

$$Prob_{Res} = Y_2 = \frac{RS_i}{RC_i + RS_i} \tag{4.5}$$

Likewise, the distribution Y is formed from a testing interval using the formulas presented in Equation 4.4 and Equation 4.5. The key distinction here lies in the utilization of query and response counts from that specific interval or duration. In crisp, we create Y based on the query and response data observed during the testing period. This enables us to construct a distribution representative of the communication patterns during the testing phase, facilitating comparison with the training profile X.

Figure 4.2 illustrates probability distributions derived from training and testing samples during normal operation, specifically in the context of Modbus communication. In Figure 4.2a, we observe the distribution generated from average values of



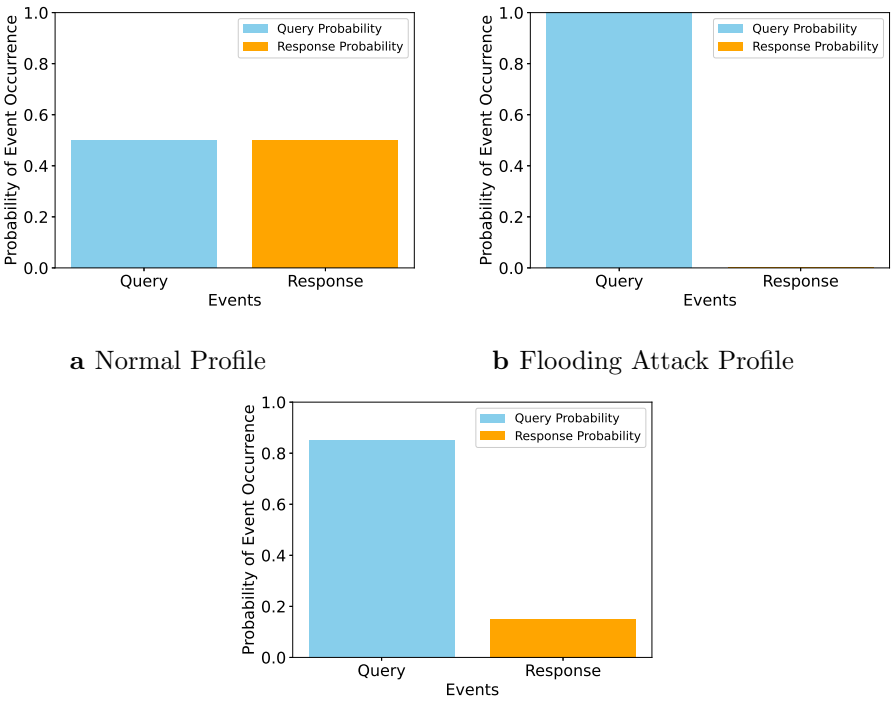
a Probability Distribution of Normal Training Interval

b Probability Distribution Normal Testing Interval

Figure 4.2: Probability Distribution Comparison for Dataset-1

the training dataset collected over a period of 146.62 hours (as described in Dataset-1, detailed in Section 4.4.1). This distribution serves as a representation of typical communication behavior during normal operation. On the other hand, Figure 4.2b displays the distribution generated from a testing interval characterized by normal communication patterns. These distributions provide insights into the typical communication dynamics observed in the system under normal circumstances. Conversely, Figure 4.3 presents a comparison of probability distributions during normal communication with distributions generated from intervals associated with query flooding attacks and bruteforce attacks. Here, Figure 4.3a depicts the same profile as in Figure 4.2a, representing the training dataset. In contrast, Figure 4.3b and Figure 4.3c illustrate the probability distributions observed during testing intervals corresponding to a flooding attack and a brute force attack, respectively.

Figure 4.4a shows the distribution graph of the baseline profile generated from the normal communication of Dataset-2 (More information about Dataset-2 can be found in Section 4.4.1). As observed previously both query and response messages appear in almost equal numbers and hence their probabilities are also similar. Figure 4.4b shows the distribution graph generated from the flooding attack data of Dataset-2. Upon examining these sample distributions, we can conclude that they closely resemble each other during normal operation. However, their balance is visibly disrupted



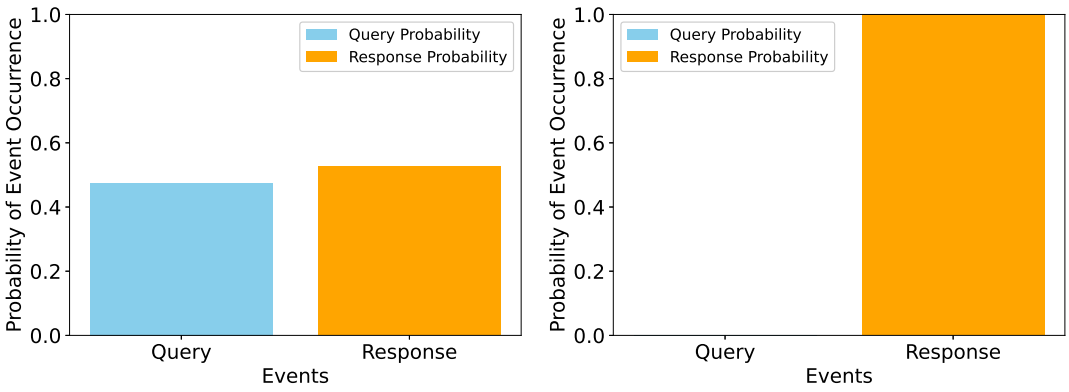
a Normal Profile

b Flooding Attack Profile

c Brute force Attack Profile

Figure 4.3: Probability Distribution Comparison (Dataset-I)

under attack conditions. This disruption is evident in the form of notable deviations from the expected communication patterns, highlighting the effectiveness of the proposed anomaly detection approach in identifying and distinguishing between normal operation and attack scenarios.



a Training Profile

b Flooding Attack Profile

Figure 4.4: Probability Distribution Comparison (Dataset-2)

4.4 Experiments

In this section, we provide a detailed explanation of the experiments conducted to validate the proposed anomaly detector. The following three subsections offer information on the datasets utilized for the experiments, the results obtained, and a sensitivity analysis for various parameters, respectively.

4.4.1 Dataset Details

For our analysis, we conducted experiments using two datasets: the CIC Modbus Dataset 2023 [60] and the Cyber-Security Modbus ICS Dataset [61]. In the subsequent discussion, we refer to these datasets as Dataset-1 and Dataset-2, respectively.

(i) Dataset-1: The CIC Modbus Dataset 2023 [60] originates from a simulated environment carefully constructed by researchers at the University of New Brunswick. This simulated setup utilizes Docker containers to emulate the functionalities of Intelligent Electronic Devices (IEDs) and SCADA Human Machine Interfaces (HMIs) found in an energy distribution substation network. To simulate the behavior of these components, Python scripts were developed to replicate the logic of IEDs and HMIs. This dataset is publicly available and consists of labeled pcap files containing a range of communications, from benign to malicious. The malicious communications include various attack types such as reconnaissance, query flooding, stacking Modbus frames, bruteforce write, etc., following the MITRE ICS ATTACK techniques [62] providing essential insights. Table 4.1 presents key details about the dataset, including the duration covered by both attack instances and normal captures. This dataset serves as a valuable resource for researchers and practitioners, offering a realistic understanding of cyber-physical systems and helping in the development of effective defense strategies against potential threats

(ii) Dataset-2: The Cyber-Security Modbus ICS Dataset [61] originates from the University of Coimbra, where a testbed was developed to mimic a Cyber-Physical System (CPS) process managed by a SCADA system. Within this simulated network, the Modbus protocol was used for communication between different components of

Table 4.1: Dataset-1 Details

File Name	Event	Capture Duration
compromised-scada	Attack	169.92 hours
scada-hmi	Benign	194.62 hours

Table 4.2: Dataset-2 Details

File Name	Event	Capture Duration
Captures1	Normal Communication	7.5 hours
Captures1	Flooding Attack	155 hours
Captures2	Flooding Attack	29.5 hours
Captures3	Flooding Attack	29.5 hours

the SCADA system. This dataset comprises three units: Captures1, Captures2, and Captures3. These units contain traces captured during both normal operations and various attack scenarios. The traces are stored in the pcap file format, ensuring compatibility with widely-used tools such as tcpdump [63] and Wireshark [64]. Table 4.2 offers comprehensive details about this dataset, including the duration covered by both attack instances and normal captures. This dataset serves as a valuable asset for researchers and practitioners, providing valuable insights into the workings of cyber-physical systems and facilitating the development of robust defense strategies against potential threats.

4.4.2 Evaluation

We carefully processed the pcap files from both datasets to extract query and response statistics across various intervals. This involved filtering Modbus packets using Wireshark filters and then extracting packet details into CSV files. Following this, Python scripts were used to analyze these CSV files and generate detailed statistics on queries and responses. To establish a reliable baseline profile for normal Modbus SCADA communication, we utilized labeled pcaps containing instances of normal

communication. The corresponding CSV files were also used to ensure accuracy in generating the profile of normal communication. Additionally, we documented the specifics of training and testing intervals, along with other important details for both datasets.

(i) Dataset-1: The CIC Modbus Dataset 2023 contains a comprehensive record totaling 364.54 hours of data, encompassing both normal operations and instances of attacks, as depicted in Table 4.1. To facilitate our experimental procedures, we divided the 194.62 hours of normal data into two distinct segments. Of these, one segment, spanning 146.62 hours, was set aside for generating the foundational baseline profile representing normal behavior. The remaining 48 hours of normal capture, alongside the entirety of the 169.92 hours dedicated to attack scenarios, were reserved for testing purposes.

Table 4.3: Training & Testing Data Duration(CIC Modbus Dataset 2023)

Data	**Duration**	**Intervals**	
Normal Training Trace	146.62 hours	1760	
Normal Testing Trace	48 hours	576	
Attack Testing Trace	169.92 hours	Normal	1183
		Attack	857

The division of data is detailed in Table 4.3, which provides insights into the number of intervals, each lasting five minutes, allocated for different categories. It's noteworthy that files labeled with attacks indicate specific timeframes during which attacks were initiated. Consequently, the actual duration of attacks spans multiple discrete time periods. We carefully identified and annotated these intervals to signify attack occurrences, while the rest were designated as normal intervals. Hence, the total number of normal intervals designated for testing comprises a combination of those found

within the attack trace files and the 48 hours of normal capture from the initial segment. These details, of the division and utilization of intervals for various purposes are presented in Table 4.3.

(ii) Dataset-2: The Cyber-Security Modbus ICS Dataset encompasses a total duration of 221.5 hours across all three captures combined. Following a similar approach to Dataset-1, we divided these traces into three segments, as outlined in Table 4.4.

Table 4.4: Training & Testing Data Duration

Data	**Duration**	**Intervals**	
Normal Training Trace	6 hours	180	
Normal Testing Trace	1.5 hours	45	
Attack Testing Trace	214 hours	Normal	3848
		Attack	2572

However, unlike the previous dataset, we opted for a time duration of 2 minutes for each interval, which yielded the most optimal results (further details on sensitivity analysis are provided in the next subsection). For training purposes, a total of 180 intervals were utilized to generate the baseline profile, representing normal behavior. Subsequently, a larger set comprising 6465 intervals was allocated for testing purposes. This division ensured a comprehensive assessment of the anomaly detector's performance across a diverse range of scenarios. These details concerning the segmentation and utilization of intervals for training and testing purposes are elaborated in Table 4.4, providing a clear understanding of the experimental setup for Dataset-2.

We assessed the effectiveness of our detection method using four key metrics: Detection Rate, False Positive Rate, Accuracy, and F1 Score whose details are mentioned in Chapter 3.

In our evaluation, we established specific threshold values for the Hellinger distance and window periods (Δ), as outlined in Table 4.5 and Table 4.6 for Dataset-1

Table 4.5: Parameters for Dataset-1

Parameter	**Threshold Value**
distance	0.48
Δ	5

Table 4.6: Parameters for Dataset-2

Parameter	**Threshold Value**
distance	0.25
Δ	2

and Dataset-2, respectively. With these parameters in place, we proceeded to assess the detection performance of our method. The results of this evaluation are presented in Table 4.7, showcasing the performance achieved for both datasets. We observed that our detection method performed well for both datasets, demonstrating promising results. However, it's important to note that the performance achieved depends on the chosen parameter values to some extent. In the subsequent subsection, we delve into a detailed examination of how the method's performance varies with these parameter values. This analysis allows us to gain deeper insights into the behavior and effectiveness of our detection approach across different configurations.

Table 4.7: Detection Performance

Dataset	**DR**	**FP**	**A**	**F1**
Dataset-1	100%	0%	100%	100%
Dataset-2	100%	0%	100%	100%

4.4.3 Sensitivity Analysis

It's important to highlight that we incorporated threshold values during the calculation of the Hellinger distance between probability distributions. Additionally, the

window time also influences the detection performance. Therefore, we conducted a comprehensive analysis to examine how the performance of our detection methods varies concerning these values for both datasets. This analysis is done to understand the impact of threshold values and window time on the effectiveness of our detection approach. By systematically studying their influence on detection performance, we gain valuable insights into the optimal configuration of parameters for achieving the most accurate and reliable results. This analysis is crucial for fine-tuning our method and enhancing its effectiveness in detecting anomalies within the datasets.

(i) Hellinger Distance Threshold: In this analysis, we conducted a sensitivity assessment by varying the distance threshold values incrementally by 0.2. We evaluated both the detection rate and false positive rates across different threshold values.

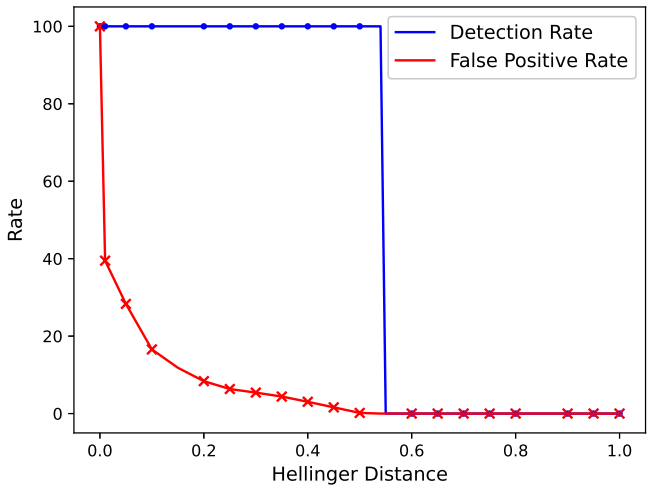


Figure 4.5: Hellinger Distance v/s Detection Performance for Dataset-1

Figures 4.5 and 4.6 depict the performance variation for Dataset-1 and Dataset-2, respectively. Notably, for threshold values up to 0.55, both datasets exhibited a nearly 100% detection rate with an acceptable rate of false positives.

This observation indicates the robustness of our method in accurately detecting anomalies while maintaining a low false positive rate within this threshold range. It highlights the effectiveness of our approach in effectively distinguishing between normal and anomalous behavior across different datasets. Such insights are crucial for

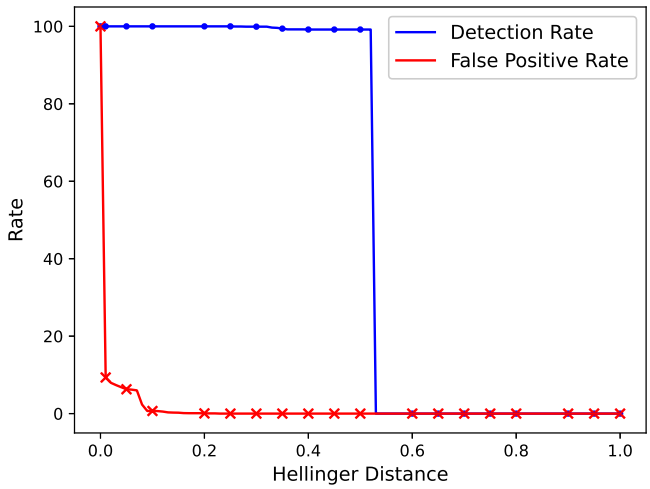


Figure 4.6: Hellinger Distance v/s Detection Performance for Dataset-2

optimizing the parameters of our detection method to achieve optimal performance in real-world scenarios.

(ii) Time Window: Similar to our previous analysis, we conducted a sensitivity analysis by varying the time window period. For Dataset-1, we varied the window period in

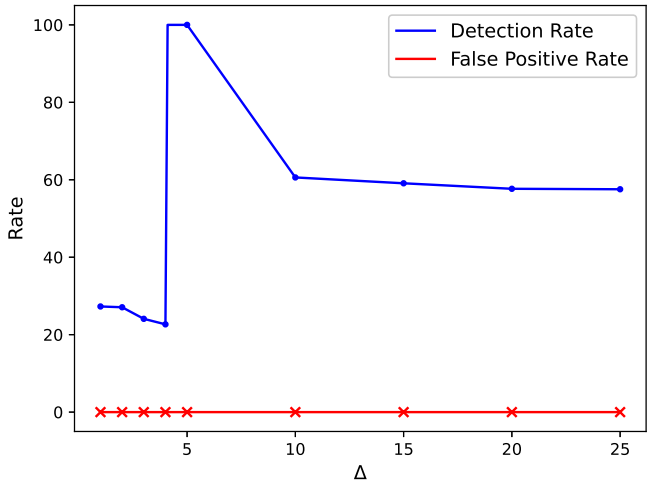


Figure 4.7: Time Window v/s Detection Performance for Dataset-1

increments of five minutes, while for Dataset-2, we used increments of two minutes. Figures 4.7 and 4.8 illustrate the performance variation for Dataset-1 and Dataset-

2, respectively. Notably, in Figure 4.7, Dataset-1 demonstrates optimal performance only for a window period of 5 minutes. Deviations from this window size result in decreased detection rates, while the false positive rates remain relatively stable.

Conversely, for Dataset-2, we observe a slight decline in detection rates for larger window values, while the false positives remain stable. This deviation is attributed to variations in the duration of attacks launched. In Dataset-1, attacks span relatively short durations, making them detectable with window sizes closer to the attack duration. Conversely, attacks in Dataset-2 persist for longer periods, resulting in nearly constant performance across smaller time intervals. This highlights the importance of selecting an appropriate time window to reflect real-world attack durations accurately. However, smaller time durations may still aid in detecting longer duration attacks, emphasizing the need for careful consideration in parameter selection to optimize detection capabilities.

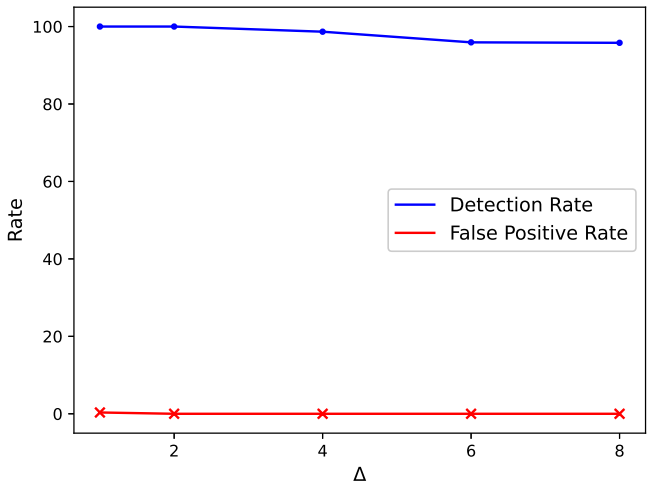


Figure 4.8: Time Window v/s Detection Performance for Dataset-2

4.5 Conclusion

Smart grid networks serve as crucial infrastructure supporting millions of users for their energy requirements. These networks leverage Information and Communication

Technology (ICT) to interconnect sensory infrastructure with control centers, facilitating decision-making processes. However, given the well-documented cybersecurity vulnerabilities associated with TCP/IP, smart grid communication inherits these security challenges. In our study, we introduce a lightweight anomaly detection model tailored to identify two significant cybersecurity threats within smart grid networks. Our approach involves generating probability distributions based on known normal communication patterns within the smart grid network. To identify potential attacks, our detector assesses the distance between a baseline profile and a profile generated from a test interval.

Notably, our proposed approach stands out for its lightweight nature, requiring minimal computational resources for effective operation. Through evaluation of our proposed approach on publicly available datasets, we demonstrate the effectiveness of our anomaly detector in accurately detecting these threats. This research contributes to enhancing the security posture of smart grid networks, safeguarding critical infrastructure against cyber threats.

Chapter 5

Mitigating Resource Depletion and Message Sequencing Attacks in SCADA Systems

5.1 Introduction

In Chapters 3 and 4, we introduced cyber attack detection methods in SCADA systems using state transition modeling and probability distribution techniques, respectively. Now, in this chapter, we shift our focus to mitigating attacks, particularly dealing with malformed message attacks. The traditional approach for protecting SCADA systems involves deploying firewalls to filter packets from known attackers and unknown sources. However, this method has limitations. Firewalls are often implemented as software solutions, which can themselves be vulnerable to attacks. Additionally, filtering may not be reliable unless it inspects the content of packets. Various attack detection methods have been proposed, but they typically run as applications, resulting in overhead from bringing packets through kernel space. To address this, we propose offloading the filtering operation to the network interface card. In this direction, we make the following contributions in this chapter.

- We study different types of malformed attacks that can be generated by manip-

ulating the application layer messages of IEC-104 and Modbus.

- We propose a method to identify all malformed messages using formal constructs generated from protocol specification.
- We study the impact of malformed packet attacks on the system resources advocating the need for mitigation techniques.
- We suggest a method to transfer the screening and filtering of malformed packets to programmable network interface cards, resulting in substantial savings in system resources.

The remaining chapter is organized as follows. In Section 5.2, we quickly review the prior works related to cyber attack mitigation in smart grid networks particularly using IEC-104 and Modbus communication. In Section 5.3, we provide the details of the proposed mitigation technique. We present the evaluation results in Section 5.4. Finally, the chapter is concluded in Section 5.5.

5.2 Related Work

Existing works related to the security of Industrial Control Systems (ICS) can be categorized into two parts. Firstly, there are works related to attack vectors [13, 14, 15]. Commonly studied attacks include false data injections [15] and denial of service attacks [16], which disrupt normal operations either by misleading data or by overwhelming computational resources. A lot of attention has been given to these conventional attacks, but we study a different class of attacks that target the computational resources of the control center. Similar attacks are studied in other domains like web servers [17]. Secondly, there are works related to attack detection. There are several machine learning algorithms [18, 19], state transition modeling approaches [9, 20, 7] and commercial intrusion detection system solutions [21] available for this purpose. Many of these techniques rely on Deep Packet Inspection (DPI) [22] for detecting these attacks. However, these methods detect conventional attacks like

scanning, injection, connection-loss, etc. For instance, Lin et al. [23] adopted Bro [24] (now called zeek) intrusion detection system for SCADA networks. They also propose to filter malformed messages in SCADA systems. However, their study is limited to a limited number of attribute types and to only the DNP3 protocol. Further, Bro rules were manually written for detection. Unlike this, we describe a systematic way to identify all types of malformed messages.

5.3 Proposed Method

In this section, we cover different types of malformed and message sequence attacks, their detection method, and mitigation techniques.

5.3.1 Background

We begin by giving a background of the two application layer protocols that are commonly used in SCADA communications and eBPF.

5.3.1.1 Communication protocols

Our proposed method focuses on enhancing the security of smart grid systems by detecting potential cyber threats. Using the widely used communication protocols like IEC-104 and Modbus within the SCADA environment. Here, we simulate realistic attack scenarios and evaluate the effectiveness of our detection mechanisms. Thus, we begin with a background of these two communication protocols. Here's an overview of both:

IEC-104: The IEC 60870-5-104 protocol stack, established by the International Electrotechnical Commission (IEC), serves for the telecontrol of equipment/systems over TCP/IP. This protocol details are provided in Subsection 3.3.1 of Chapter 3.

Modbus: Modbus over TCP/IP is a widely used industrial communication protocol, leveraging TCP/IP networks for seamless device interaction. The message structure of a Modbus communication is shown in Figure 5.1. Key characteristics of Modbus over TCP/IP encompass reliable message delivery through encapsulation, IP-based

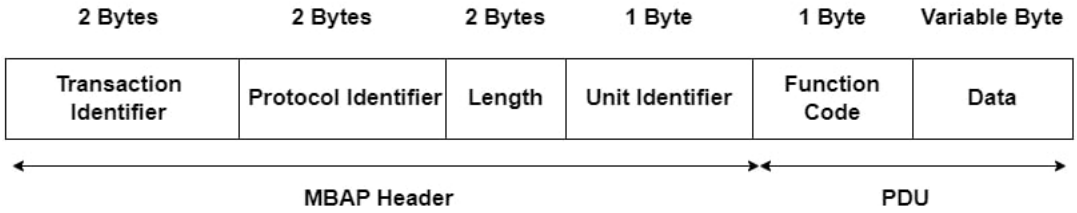


Figure 5.1: Modbus Messaging over TCP/IP

addressing, a diverse range of function codes for control and monitoring purposes, and support for different data types. The message frame consists of various fields, including the Transaction Identifier (TID) for transaction identification, the Protocol Identifier (PID) specifying Modbus over TCP/IP, the Length field indicating remaining bytes, the Unit Identifier (UID) identifying remote devices, the Function Code defining operation types (such as read or write), and the Data Field containing function-specific information. The Function Code dictates the action type in Modbus messages, while the Data Field carries information structured based on the function code. The Protocol Identifier field designates the Modbus message protocol within the TCP/IP stack, ensuring proper communication routing. The Transaction Identifier ensures pairing in request-response communication, while the Length field specifies the total bytes in the Modbus Application Protocol (MBAP) header, unit identifier, and data. Lastly, the Unit Identifier aids in routing between Modbus serial and TCP/IP networks within the system. These components collectively facilitate efficient and standardized communication in industrial environments.

5.3.1.2 eBPF

Extended Berkeley Packet Filter (eBPF) extends the traditional Berkeley Packet Filter (BPF) by allowing for more complex and customizable packet processing within the kernel. This flexibility enables a wide range of applications, including network monitoring, security enforcement, and performance tuning. By leveraging eBPF, developers can efficiently implement advanced networking features without the need for kernel modifications, thereby maintaining system stability and security. Additionally, eBPF's integration with programmable network interface cards (NICs) further

enhances performance by offloading processing tasks to hardware, reducing CPU overhead and latency. This combination of flexibility, safety, and performance makes eBPF a crucial technology for modern Linux-based networking environments.

5.3.2 Malformed and Message Sequencing Attacks

In this section, we discuss how an attacker can generate malformed and message sequencing attacks taking a reference grid architecture. A typical grid consists of power generation sources placed at one end and customers residing at the other, as shown in Figure 5.2. The link between the two ends is the control center, which is responsible for forecasting the demand and supply as well as identifying network vulnerabilities. The control center receives a large volume of measurement data from various points across the grid network, providing crucial insights into the state of the system. This data is essential for accurately assessing the balance between power generation and consumption, identifying potential anomalies or faults, and managing timely interventions to maintain grid stability and reliability. However, this reliance on external data sources also makes the control center vulnerable to malicious manipulation. An adversary could exploit vulnerabilities within the communication channels or the data itself to inject false or misleading information into the system. By tampering with the integrity of the measurement data, the adversary could distort the control center's perception of the grid's state, leading to erroneous decisions and potentially harmful consequences. These attacks are well covered in literature and also by the methods described in the previous two chapters.

In this chapter, we focus on an attack that can target the control center by depleting its resources and disrupting the orderly flow of communication between the control center and grid components. By manipulating the sequence in which messages are received or processed, the adversary could introduce delays, reorder commands, or even replay outdated information, undermining the efficiency and reliability of the control system. Thus, safeguarding the integrity and authenticity of the data transmitted to the control center, as well as ensuring the resilience of communication channels against sequencing attacks is important for securing industrial control systems against

malicious exploitation.

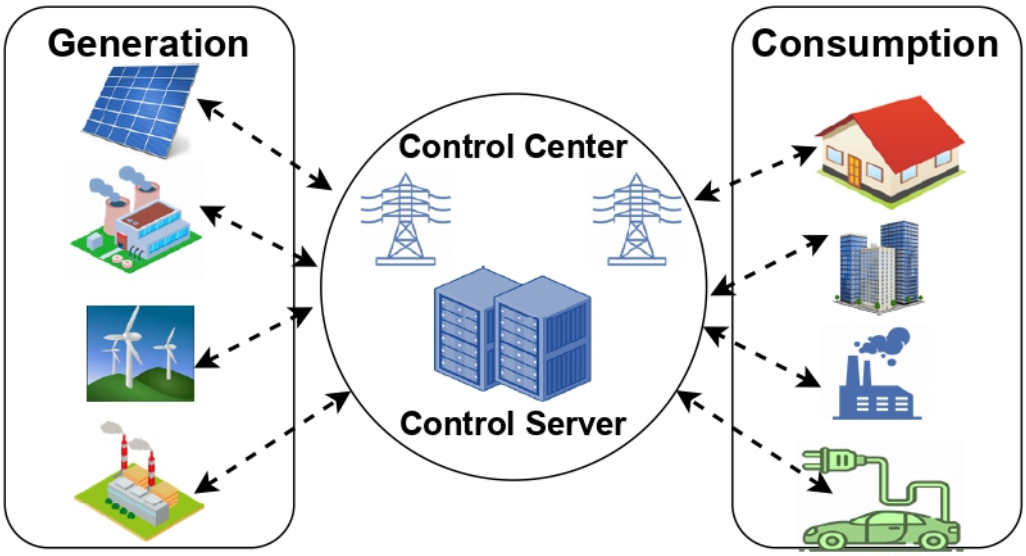


Figure 5.2: Grid Network Communication Architecture

5.3.2.1 Malformed Messages and Detection

In the depicted grid architecture (Figure 5.2), we show a potential attack scenario wherein an adversary aims to overwhelm the control center with a large volume of malformed messages. An adversary generates malformed packets by setting different values in the packets. These packets, though appearing benign on the surface, have two distinct types of harmful consequences:

(i) Misinterpretation of Status and Execution of Undesirable Actions: Some of these messages might confuse the control center into taking action otherwise it should not. The control center might misunderstand the status of the SCADA system and end up taking certain actions.

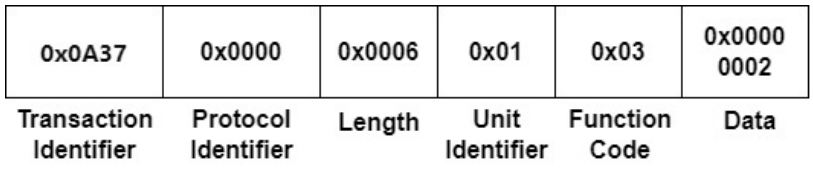
(ii) Server Resource Depletion: Even if these messages are not triggering any action, they still take up space and time on the server. With a large set of messages coming in, the server can get overwhelmed, making it harder for it to handle other real tasks.

These messages, capable of disrupting the system, are crafted with relative ease using readily available packet generation tools or through minimal programming efforts. Their impact extends beyond mere disruption, potentially compromising

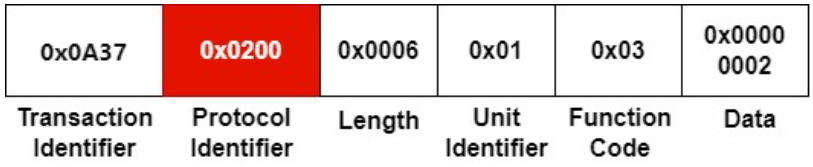
the reliability, integrity, and overall functionality of the grid setup. Therefore, it is imperative for system defenders to devise robust countermeasures to detect and mitigate these attacks. Following is a non-exhaustive list of such malformed message cases and also its implications.

Modbus Malformed Messages

`(i) Protocol Identifier Field Error:` The value is set to 0x0000 for smooth communication within the network. If any other value is used, the message is termed malformed, which means it is incompatible with the expected format or structure. This deviation introduces a risk of communication errors and may confuse devices in the network. Thus, using standard protocol identifier value 0x0000, we ensure clear and reliable communication, minimizing the potential for errors and enhancing overall network efficiency and effectiveness. Figure 5.3a shows the normal Modbus message that uses the standard protocol identifier value, while Figure 5.3b shows the malformed Modbus message with an incorrect protocol identifier value.



a Normal Modbus Message



b Malformed Modbus Message

Figure 5.3: Packet Structure of Normal and Malformed Modbus Message

`(ii) Incompatible Value Message:` This type of attack occurs when conflicting values are inserted into the message. For instance, using an unconventional Protocol Identifier produces an incompatible value message. Similarly, setting the function code for a query as 'x' where '$x\&0x80 == 1$' denotes an exception range, creates a clash in the function code values, resulting in an incompatible message. Such

inconsistencies disrupt smooth communication and data exchange within SCADA systems, potentially causing operational issues.

IEC-104 Malformed Messages

(i) `Wrong Start Character:` In IEC-104, the start character is fixed $0x68$. Using alternative values can result in an invalid character being detected or treated as the start character is absent, leading to packet synchronization problems.

(ii) `Incompatible ASDUTYPE and CoT Values:` Such deviations can significantly impact the correct execution of commands within SCADA systems. For example, when the ASDUTYPE is designated as 2, only specific Cause of Transmission (CoT) values, such as 3, 5, 11, and 12, are termed valid. Any deviation from these predefined combinations, such as employing an incompatible CoT value like 8, can lead to commands being misinterpreted. This misinterpretation has the potential to disrupt the seamless functioning of the system. Considering there are 127 ASDUTYPE variations, each with its own unique set of valid CoT values, the sheer number of potential combinations increases the complexity and susceptibility to errors within the system.

(iii) `Incompatible Length and Number of Objects Fields:` Such messages result in packet parsing and interpretation errors due to deviations between the mentioned number and the actual data available within the Application Protocol Data Unit (APDU). This mismatch confuses the parsing process, leading to errors in interpreting the message contents. Consequently, the system may fail to correctly interpret the messages, potentially compromising its functionality.

(iv) `Incorrect Test Bit Setting (T Bit):` This particular bit is designated solely for testing purposes. However, toggling its state, whether setting or unsetting it, can result in misinterpretations. Such misinterpretations may occur when test messages are erroneously identified as operational commands or conversely, when operational commands are mistaken for test messages. This confusion can lead to unintended consequences, potentially disrupting normal system operations or affecting the integrity of test results. Therefore, caution must be exercised when manipulating this bit to avoid any misunderstandings or operational errors.

(v) `Mismatch in P/N Bit Setting:` This particular field acts as an indicator determining whether an execution has succeeded or failed. However, if there is a discrepancy in the Positive/Negative (P/N) bit—where P/N equals 0 or 1—it can lead to erroneous conclusions regarding the outcome of command execution. Such inaccuracies can significantly affect the reliability and precision of the grid system. It's crucial to maintain consistency in interpreting the P/N bit to ensure an accurate assessment of command execution results and uphold the integrity of the grid system's operations.

The examples of malformed messages or packets discussed earlier can generally be categorized into two types: (i) those with illegal settings and (ii) violations of dependent attribute values. In response to this, we present a common method for identifying such messages. We utilize a framework based on first-order logic to express constraints in accordance with the specification. This approach enables us to systematically detect all variations of malformed packets by defining logical rules that capture the specific conditions leading to packet irregularities. By using this method, we can effectively identify and address potential issues within the network, enhancing its overall reliability and performance.

We view Modbus and IEC-104 packets as a message M, consisting of attributes $a_1, a_2, \cdots, a_n$, where each $a_i \in M$. These attributes come from a well-defined domain D_i with every D_i corresponding to a_i. We identify two types of malformed messages through constructed logic statements as follows.

(i) Identifying illegal settings: Let $L(a_i)$ indicate whether attribute a_i takes value from its domain D_i, where *true* means it does and *false* means it doesn't, then a message M is malformed if there's at least one attribute a_i that breaks its domain rule. In simpler terms, if any attribute a_i in the message M picks a value it shouldn't, then the message is malformed. This idea can be expressed with the logical statement $\exists a_i \neg L(a_i)$, which says that there is at least one such attribute a_i in the message. For instance, if a SCADA system receives a packet with an Information Object Address (IOA) of 65537 in the IEC-104 protocol, it falls outside the typical range of IOAs,

which is restricted to 65535. This packet would be considered anomalous or erroneous as it exceeds the permitted range of IOAs.

(ii) Dependent attribute violations: Suppose we have two attributes a_i and a_j in a message M. Attribute a_i has a domain D_i with some values $x_1, x_2, ..., x_m$, and attribute a_j has a domain D_j with values $y_1, y_2, ..., y_k$. We call a_i the primary attribute and a_j the dependent attribute because the value of a_j depends on the value of a_i. If a_i takes a value x_i, then the dependent attribute a_j can only take a subset of values from its domain, let's call it $D_j^{'}$. To represent these dependencies logically, we first identify such pairs of attributes (a_i, a_j) from the protocol specifications. Then, we expand the attribute set and partition the domain set. This means that we create new attributes based on a_i and its possible values, and each new attribute a_{it} corresponds to a subset of the domain D_i. These subsets may overlap, but they represent the different possibilities for a_i and consequently for the dependent attribute a_j.

$a_i = \{a_{i1}, a_{i2}, \cdots, a_{ip}\}$ and $D_i = \{D_{i1}, D_{i2}, \cdots, D_{ip}\}$ with a_{it} corresponding to D_{it} for $1 \leqslant t \leqslant p$.

$a_j = \{a_{j1}, a_{j2}, \cdots, a_{jp}\}$ and $D_j = \{D_{j1}, D_{j2}, \cdots, D_{jp}\}$ with a_{jt} corresponding to D_{jt} for $1 \leqslant t \leqslant p$.

If $L(a_{it})$ is true, it means that the attribute a_i takes one of the values from the set D_{it}. In this case, the attribute a_{jt} should also take values from its own domain D_{jt}. We can express this constraint as a logical statement: "There doesn't exist a pair of attributes (a_{it}, a_{jt}) for which both $L(a_{it})$ and $L(a_{jt})$ are false." This statement ensures that if a_{it} takes a valid value, then a_{jt} must also take a valid value, maintaining the dependency between them. For instance, if a SCADA system receives a packet having ASDUTYPE as 4 and CoT as 2, then this represents an anomaly due to the incompatible value of ASDUTYPE and CoT. The ASDUTYPE 4 is only compatible with CoT values 3, 5, 11, and 12.

5.3.2.2 Message Sequencing Attacks

In this scenario, attackers manipulate the sequence of messages sent to the control center intentionally to disrupt its stability. By adjusting attributes such as CoT and

ASDUTYPE within IEC-104 messages, attackers can trigger unwanted actions, like turning a target device on and off repeatedly [7]. What makes these attacks particularly challenging to detect is that the attribute values being used are technically legitimate. However, it's the specific combination and sequence of these values that lead to adverse effects. To effectively counter these sequencing attacks, a different approach is necessary compared to traditional attack detection methods. Instead of simply flagging individual messages as malicious, the focus shifts to identifying patterns or sequences of messages that, when combined, result in disruptive behaviour. This requires a more sophisticated detection mechanism capable of analyzing the broader context and sequence of message interactions to identify potential threats.

Let's consider a sequence of messages, $M_1, M_2, \cdots, M_n$, received within a time period W. Each message, M_i, contains a set of attribute values: $M_i = a_1, a_2, \cdots, a_m$, where $1 \leqslant i \leqslant n$ and $1 \leqslant t \leqslant m$. To keep track of these attribute values over time, we log them in auxiliary storage along with their corresponding timestamps. This logging process can be denoted as $Log(W_q) = (t_1, M_1), (t_2, M_2), (t_3, M_3), \cdots, (t_n, M_n)$ for the q^{th} window period. Essentially, for each window period, we keep a record of when messages were received and what attribute values they contained. To detect potential attacks, we look through these logs during each window period. By examining the attribute values, we can spot any patterns that might indicate the presence of an attack. This checking process allows us to monitor how the attribute values change over time and catch any unusual or suspicious patterns that could signal malicious behaviour.

5.3.3 Mitigating Attack

We introduce a novel attack mitigation technique that uses filtering mechanisms at the operating system (OS) kernel or Network Interface Card (NIC) of the server located at the control center. This technique involves examining incoming packets, specifically those that are adhering to the IEC-104 and Modbus protocols, to ensure they are well-formed and contain legally permissible combinations of values. To implement this packet filtering mechanism, we leverage the Extended Berkeley Packet Filter (eBPF). eBPF enables the execution of custom code within the Linux kernel by attaching

programs to kernel hooks. These hooks can be triggered by various events, such as system calls or the arrival of new packets. Importantly, eBPF programs can also be offloaded to programmable network interface cards, as depicted in Figure 5.4. The versatility of eBPF allows developers to create and execute custom programs in the kernel space without the need to modify the kernel itself. These programs are typically written in a restricted subset of the C programming language and undergo verification by the eBPF verifier to ensure the safety of execution. This verification process aims to prevent any adverse effects, such as kernel crashes, thereby maintaining the stability and reliability of the system.

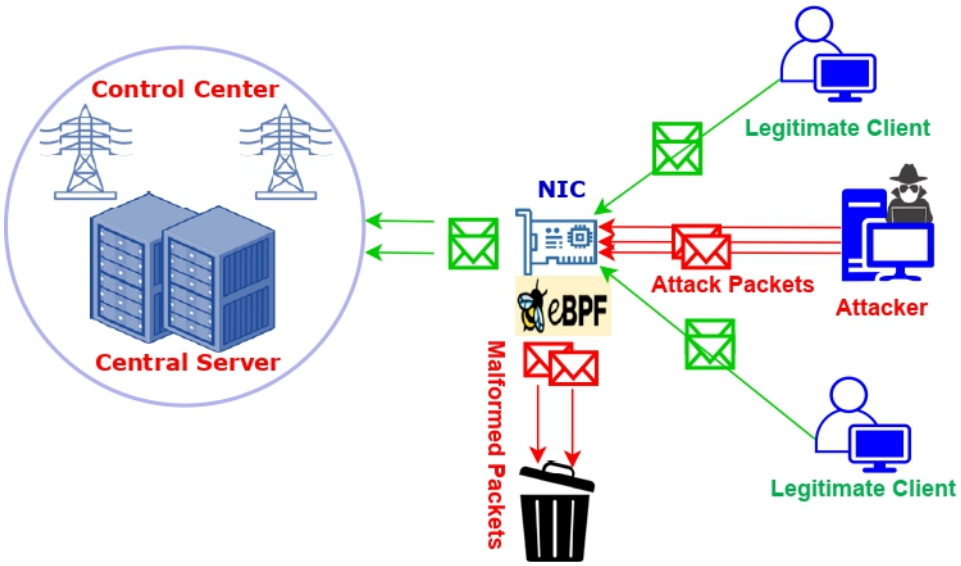


Figure 5.4: Proposed Packet Filtering with eBPF

We propose utilizing the capabilities of eBPF to implement a validation code for checking incoming Modbus and IEC-104 messages. As mentioned previously, this approach offers two potential scenarios: executing the code within the kernel space or offloading it to a network interface card (NIC). When running in the kernel space, the validation code can enhance performance by reducing context switch events and minimizing data copy operations between kernel and user space buffers. Conversely, if offloaded to a hardware NIC, packets can bypass the kernel altogether and be sent directly to the user space application, further optimizing performance. Technologies like XDP [65] facilitate this offloading of eBPF programs to Smart NICs. The eBPF

code conducts a two-step validation process. Firstly, it identifies whether the packet corresponds to the Modbus or IEC-104 protocol by examining header details and destination port numbers. Subsequently, the code executes a series of rules derived from the logic statements outlined in Section 5.3.2, aiming to detect malformed message attacks. To facilitate this detection and filtering process, the eBPF code utilizes the maps feature provided by eBPF. This feature enables the storage of timestamped messages along with their attribute values, facilitating the identification and filtration of sequencing attacks. Overall, our approach leverages eBPF's capabilities to efficiently validate incoming messages and safeguard against various forms of cyber threats.

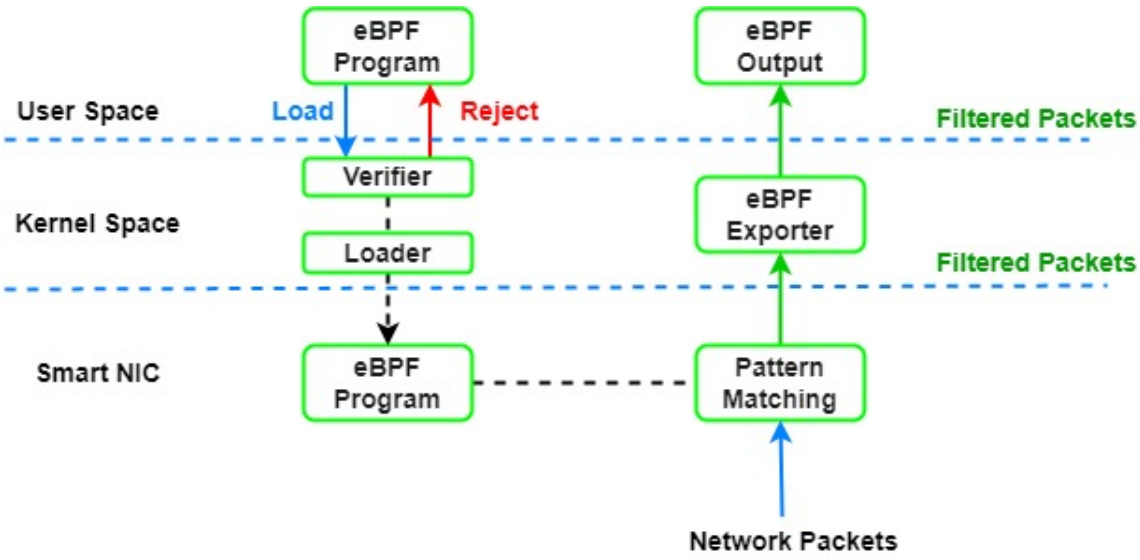


Figure 5.5: eBPF-based Filtering of Packets at NIC

The eBPF screening program functions by actively searching for specific patterns and combinations of attribute values in real time within incoming packets. It operates with the aim of identifying well-formed packets and filtering out any that do not meet the predefined criteria. This process occurs instantaneously and seamlessly, ensuring that only valid packets are forwarded directly to the user space application for further processing thus saving system resources at the control center. Figure 5.5 illustrates this process, highlighting the efficiency and effectiveness of the eBPF program in real-time packet screening.

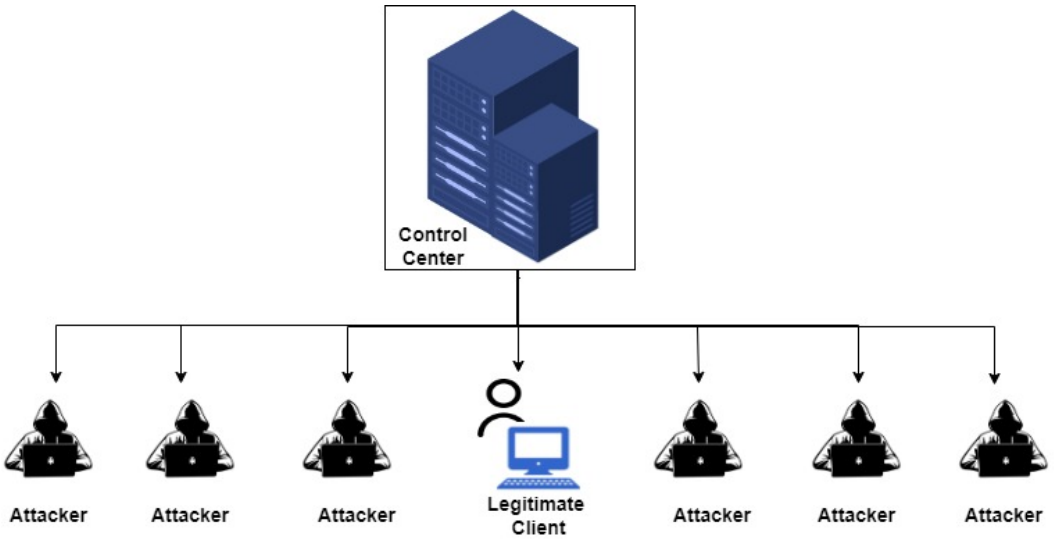


Figure 5.6: Testbed Setup

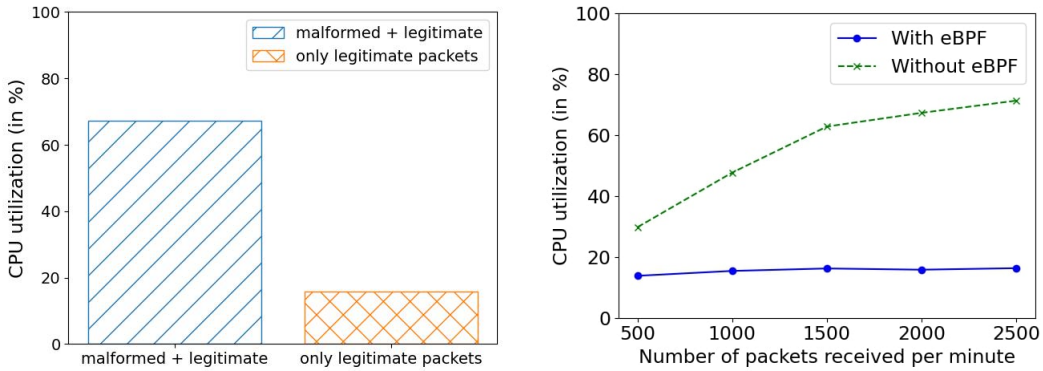
5.4 Experiments and Evaluation

In this section, we test the attack detection and protection methods we came up with. We establish a controlled test environment mirroring real-world conditions to assess their efficiency. Through careful examination, we explore the consequences of malformed messages and find out about their potential risks to the system. Furthermore, we demonstrate the effectiveness of eBPF-based filtering in preventing various attacks. Our findings promise the robustness and reliability of our methods in safeguarding the system's integrity.

`(i) Testbed Setup:` To comprehensively understand the implications of malformed message attacks and efficiently implement filtering methods to counter them, we established a testbed comprising eight computers, each with a specific role as shown in Figure 5.6. One computer served as the control center and master device, while another acted as a legitimate client, engaging in communication tasks with the master. Both the master and client functionalities were simulated using Python programs, generating Modbus and IEC-104 messages to mimic real-world communication protocols. These messages included queries and responses, containing hypothetical readings of parameters like voltage, current, and angle values. The remaining computers in the testbed were designated for generating various malformed and sequenced messages

aimed at the server. Leveraging the Scapy library, these machines crafted custom messages tailored to simulate different attack scenarios. By carefully controlling the structure and sequencing of messages, we could effectively evaluate the system's resilience to different types of threats. This testbed provided a controlled environment for assessing the efficacy of our proposed filtering methods in detecting and mitigating attacks, all while upholding the formal standards of our evaluation process.

`(ii) Malformed Message Implication:` In the beginning of our experiments, we delve into the implications of malformed messages on the server, as highlighted in Sections 5.1 and 5.3.2. These messages strain server resources, consuming processing power to handle them. To assess this impact, we focus on CPU utilization on the server under different conditions.



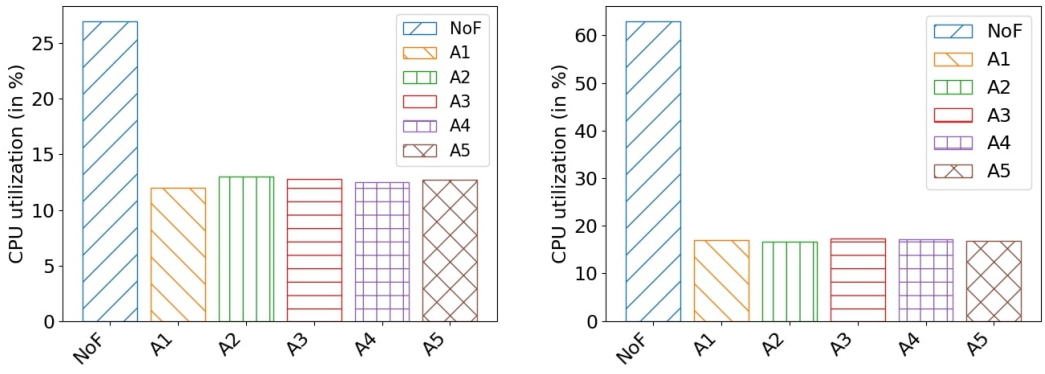
a CPU Utilization with and without Attack **b** CPU Utilization with Varying Message Intensity

Figure 5.7: Number of Messages v/s CPU Utilization

In our study setup, a single legitimate client consistently sends measurements to the control center or server at a rate of 200 messages per minute. Meanwhile, seven malicious clients collectively bombard the server with malformed messages at a rate of 2000 messages per minute. We compare CPU utilization both with and without the presence of the attack, illustrating the results in Figure 5.7a. Furthermore, we explore how varying the intensity of the attack affects CPU utilization. By incrementally increasing the number of malformed packets sent per minute—from 500 to 2500 in steps of 500, we observe changes in CPU utilization. Figure 5.7b illustrates this variation

against the intensity of the attack. Our findings reveal a notable increase in CPU utilization as the volume of malformed messages escalates. This shows the strain on server resources caused by such attacks, highlighting the importance of effective mitigation strategies.

`(iii) Filtering Malformed Messages:` Here, we provide details of our implementation of eBPF-based filtering and explain how it safeguards server resources, as outlined in Section 5.3.2. Leveraging the logical representation from our earlier discussion, we define key attributes and their corresponding partitions. These representations are translated into *if-then-else* rules within the eBPF C code, facilitating effective filtering. Additionally, the eBPF code necessitates a user-space program for direct interaction. To enable this interaction, we extend the server program to communicate with the eBPF program loaded at the NIC. The maps feature available in eBPF is utilized to store the domains of the attributes, ensuring efficient processing of incoming messages.

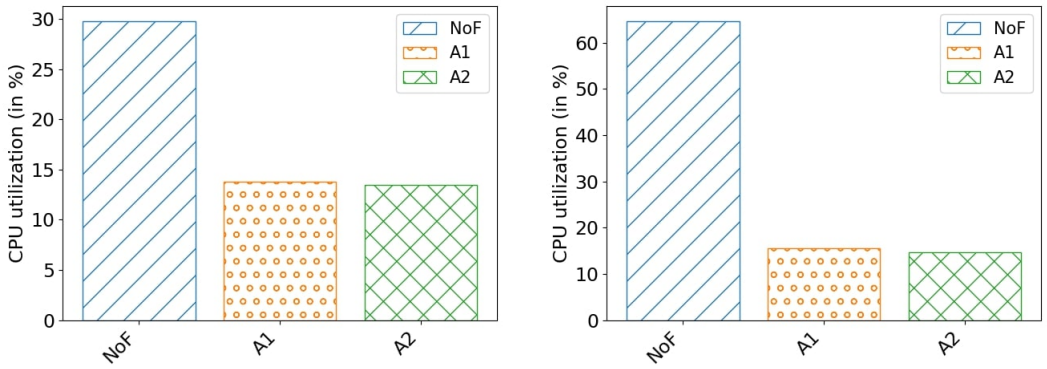


a 500 Packets Per Minute **b** 2000 Packets Per Minute

Figure 5.8: CPU Utilization with eBPF Filtering for Different Attacks using IEC-104

To evaluate the efficiency of our eBPF-based filtering approach, we conducted assessments by generating malformed messages at varying rates, ranging from 500 to 2500. Figure 5.7b showcases the corresponding CPU utilization on the server, with and without filtering enabled. As observed, the CPU utilization increases sharply with the intensity of the attack when filtering is not applied. However, with filtering enabled, all malformed messages are promptly dropped, preventing their processing by

the user-space server program. This results in a consistent CPU utilization of approximately 16%, highlighting the effectiveness of eBPF-based filtering in mitigating the impact of attacks on server resources. In addition to the overall CPU utilization, we further analyze the impact of individual attack cases on server resources. Figures 5.8a and 5.8b illustrate these insights for attack intensities of 500 and 2000, respectively, focusing on the IEC-104 protocol. In these figures, "NoF" represents CPU utilization without filtering, serving as a baseline, while "A1" to "A5" denote different attack scenarios. Each attack scenario represents a distinct type of malformed message. For instance, "A1" involves incompatible length and object fields, "A2" pertains to incorrect test bit settings, "A3" corresponds to P/N setting mismatches, "A4" comprises wrong start characters, and "A5" involves incompatible ASDUTYPE and CoT values. The bars in the graphs depict CPU utilization when an attack of the specified intensity is generated, with the corresponding messages being filtered by eBPF. Notably, the nearly consistent CPU utilization across different attack types suggests that the filtering mechanism is specific to the specific characteristics of each attack. This implies a robust and versatile defense strategy, capable of effectively mitigating various types of attacks without noticeable change in resource consumption.



a 500 Packets Per Minute **b** 2000 Packets Per Minute

Figure 5.9: CPU Utilization with eBPF Filtering for Different Attacks using Modbus

Similarly, Figure 5.9 presents CPU utilization for the Modbus protocol, considering two attack scenarios: "A1" featuring a protocol identifier error and "A2" involving incompatible value messages. Analogous observations apply here, reinforcing the re-

silience and scalability of the filtering technique across different protocols and attack scenarios. In essence, these studies underscore the robustness and efficacy of the filtering technique, demonstrating its ability to efficiently handle diverse attack types while maintaining consistent CPU utilization levels.

`(iv) Filtering Message Sequencing Attack Messages:` In this study, we explore a sequencing attack example and develop an eBPF code to detect and filter messages associated with this attack. The attack scenario involves a single adversary toggling the machine state by manipulating ASDUTYPE and CoT values in a specific sequence. Specifically, the ASDUTYPE is set to 46, followed by changes in CoT values to 6, 7, and 10 sequentially. To detect this attack, we utilize timestamp and message attribute values as described in Section 5.3.2.2. To evaluate the effectiveness of our filtering approach, we conduct two sets of experiments, each lasting 30 minutes.

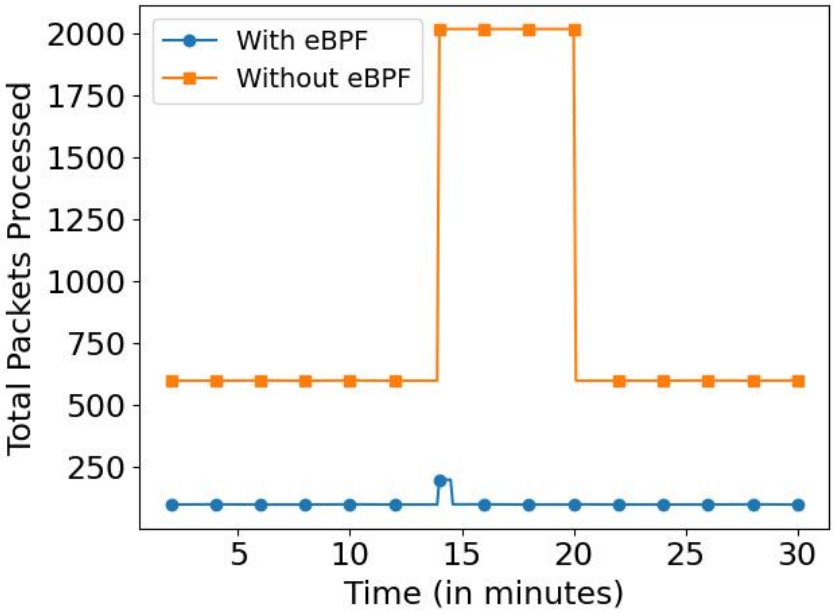


Figure 5.10: Packets Processed in Message Sequencing Attack

In the first set, we observe network traffic without any filtering, while in the second set, we implement filtering. During these experiments, the toggling attack is generated for 6 minutes, starting at the 14th-minute mark. Our eBPF code is designed to filter all messages originating from the adversary once the attack is detected. Figure 5.10 illus-

trates the number of messages processed by the control center under both scenarios. We observe a significant reduction in the number of processed messages when filtering is enabled, except during the initial period when the attack is first detected. This indicates the effectiveness of our filtering mechanism in mitigating the impact of the sequencing attack by filtering out malicious messages. Overall, our findings highlight the efficacy of eBPF-based filtering in enhancing system resilience against sequencing attacks while minimizing unnecessary processing overhead.

5.5 Conclusion

In our work, we focused on the security of SCADA communication systems used in electric grid networks, which are vulnerable to cyber attacks due to their reliance on conventional TCP/IP networks. We specifically investigated two types of attacks: resource depletion attacks and message sequencing attacks. Resource depletion attacks involve an adversary sending a large volume of malformed messages to exhaust system resources. This can disrupt normal operations by overwhelming the system's capacity to process these messages efficiently. On the other hand, message sequencing attacks involve the adversary sending messages in a specific sequence to disrupt the orderly functioning of the SCADA system.

To address these threats, we developed a method to detect malformed messages using rules generated from first-order logic statements. These rules help identify anomalies in message structures, indicating potential attacks. Additionally, we proposed a mitigation strategy involving the use of the Extended Berkeley Packet Filter (eBPF) to filter out malformed messages before they reach critical components of the SCADA system. Through simulation-based studies, we demonstrated the effectiveness and robustness of our approach in mitigating various forms and intensities of attacks. Our findings highlight the efficiency of filtering techniques in safeguarding SCADA systems against cyber threats, highlighting the importance of security measures in ensuring the reliability and integrity of critical infrastructure networks.

Chapter 6

Conclusion and Future Work

In this chapter, we summarize the attack detection and mitigation techniques presented in the thesis and highlight the possibility of future work in this area. The goal of our work was to secure smart grids by proposing attack detection and mitigation techniques. Hence, to spot the vulnerability areas, our main focus was on understanding various communication protocols used in smart grids and how attacker uses these vulnerabilities to launch the attack. Some of the popular communication protocols used in smart grids are IEC-104, Modbus, etc.

We first motivated our thesis by stating State Transition Machines can be an effective way to detect attacks in smart grids. We addressed the limitations of the existing STM based techniques for attack detection. Subsequently, we proposed a lightweight technique which is a probability distribution comparison based technique for flooding attack detection. Using probability distribution comparison can be more effective when dealing with attacks including huge redundant traffic because it offers faster processing and lesser memory usage. After that, we shifted our focus to mitigating attacks, specifically resource depletion attacks and message sequencing attacks. To achieve this, we explored the topic of eBPF and its application. Then, developed an eBPF-based attack mitigation technique.

6.1 Thesis Contributions

In this section, we summarize the attack detection and mitigation techniques described in this thesis. First two contributions describe attack detection techniques and third contribution presents an attack mitigation technique based on eBPF. These three contributions are summarized in the subsequent three subsections.

6.1.1 Anomaly Detection in SCADA Systems: A State Transition Modeling

Our first contribution of the thesis is a state transition modeling based technique for cyber attack detection in smart grid networks. For this, we first distinguished cyber attacks into three generic anomaly types, i.e. single message based, message sequence based, and time based anomalies. Then, we show that identifying these anomalies can help in detecting several other cyber attacks. We proposed a new state transition based model Deterministic Counting Timed Automata `DCTA` to detect these anomalies, which is constructed using extended network flow records. These `DCTAs` make permitted transitions based on several constraints, such as constraints on message attributes, timing constraints, and counting constraints. These constraints help to verify several attribute values, sequence, timing, number of messages, etc., to detect anomalies. Also, during transition several attribute values are logged as and when needed. Packets along with additional details for attack detection are used for network flow generation. These flows are given as input to all these `DCTAs` running in parallel and if the flow is relevant, then they process it and take appropriate transition depending upon their current state and transition constraints. If any of the `DCTAs` move to a state representing the attack state then the attack is detected. We used a publicly available dataset to evaluate the performance of our proposed approach. Our approach was successfully able to detect different attack types with a detection rate of 100% in the best case. We also did sensitivity analysis in order to come up with the most accurate golden threshold for the various parameters used. Thereafter, we compared it against the prior works to show its efficacy. Our approach outperformed

the existing techniques. DCTA was able to detect even those attacks which were not detected by existing STMs. In addition, the attack categorization which we did by introducing three anomaly types helps in providing a template for creating DCTAs for new attack types.

6.1.2 Detecting Cyber Attacks in Smart Grid Networks with Probability Distribution Comparison

Our first attack detection method summarized in the previous subsection turns out to be a heavy model when dealing with flooding attacks due to huge memory requirements and time spent while making transition decisions. To avoid this, in our second contribution, we proposed a lightweight technique based on the probability distribution method to detect two important cyber security threats in smart grid networks namely flooding attack and bruteforce attack. The known communication in smart grid networks is used to generate probability distribution for our anomaly detector. The number of query and response messages maintains a balance in normal scenarios in smart grids when the Modbus communication protocol is used. We took motivation from this and proposed an attack detection method. The proposed method builds a profile for normal behavior and then uses it as a baseline to detect attacks. Now, we generate probability distribution for the testing interval and then we compare it against the baseline profile using a distance metric. In our case, we use Hellinger distance because of its lightweight computation, natural lower and upper bounds, and finite distance value produced as output. If the computed distance crosses a pre-specified threshold then it is categorized as an attack. We performed sensitivity analysis to set these threshold values. We performed an evaluation on two of the publicly available datasets to show their efficacy in the detection.

6.1.3 Mitigating Resource Depletion and Message Sequencing Attacks in SCADA Systems

The previous work summarized attack detection techniques. However, in this subsection, we are going to summarize our third contribution which is an attack mitigation

technique for resource depletion attack and message sequencing attack. In the case of a resource depletion attack, an attacker floods the control center with malformed messages to drain its resources. On the other hand, in a message sequencing attack, an adversary sends a sequence of messages to disturb the operations of the SCADA system. We studied the impact of these malformed messages on the server resources. Thereafter, we described a method to identify such malformed messages using rules generated with first-order logic statements. These rules are constructed using the formal description of the communication protocols. We also proposed an attack mitigation method by filtering malformed messages using the Extended Berkeley Packet Filter (eBPF). For writing this eBPF code, we translated the rules of our proposed first-order logic. Our simulation based study showed that such filtering is effective and robust against variants of attacks and also the intensity of attacks.

6.2 Future Work

Our work on attack detection and mitigation in smart grid networks can be extended in many ways. Following are the few possible extensions:-

- Exploring other communication protocols: Our study is limited to only Modbus and IEC-104 message types. Although these are two popularly used protocols, there are number of other protocols proposed for SCADA applications. For e.g., DNP3 is another protocol that is used in SCADA networks. These protocols and their communication can be explored, and appropriate detection methods can be proposed.

- Exploring attacks at different OSI layers: In this thesis, we have focused purely on application layer protocols and their implications on the SCADA networks. As these protocols operate over TCP/IP networks, one can study the effect of multiplexed attacks on the SCADA systems.

- Mitigating Other Attacks: Our study has considered mitigating only two types of attacks namely resource depletion and message sequencing attacks. However,

this can be extended with techniques for mitigating other types of attacks as well.

Bibliography

[1] H. Teryak, A. Albaseer, M. Abdallah, S. Al-Kuwari, and M. Qaraqe, "Double-edged defense: Thwarting cyber attacks and adversarial machine learning in iec 60870-5-104 smart grids," *IEEE Open Journal of the Industrial Electronics Society*, vol. 4, pp. 629–642, 2023.

[2] ModbusProtocol, "Modbus tools." [Online]. Available: https://www.modbustools.com/modbus.html

[3] P. Matoušek, "Description and analysis of iec 104 protocol," *Faculty of Information Technology, Brno University o Technology, Tech. Rep*, 2017.

[4] M. Baezner and P. Robin, "Stuxnet," ETH Zurich, Tech. Rep., 2017.

[5] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," Techncal Report, SANS, Tech. Rep., March 2016.

[6] S. V. B. Rakas, M. D. Stojanović, and J. D. Marković-Petrović, "A review of research work on network-based scada intrusion detection systems," *IEEE Access*, vol. 8, pp. 93 083–93 108, 2020.

[7] P. Matoušek, V. Havlena, and L. Holík, "Efficient modelling of ics communication for anomaly detection using probabilistic automata," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 81–89.

[8] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA security scientific symposium*, vol. 46, 2007, pp. 1–12.

[9] N. Goldenberg and A. Wool, "Accurate modeling of modbus/tcp for intrusion detection in scada systems," *international journal of critical infrastructure protection*, vol. 6, no. 2, pp. 63–75, 2013.

[10] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *CPSS '15: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, 2015, pp. 13–24.

[11] A. C.-F. Chan and J. Zhou, "Non-intrusive protection for legacy scada systems," *IEEE Communications Magazine*, vol. 61, no. 6, pp. 36–42, 2023.

[12] M. Alanazi, A. Mahmood, and M. J. M. Chowdhury, "Scada vulnerabilities and attacks: A review of the state-of-the-art and open issues," *Computers & security*, vol. 125, p. 103028, 2023.

[13] C. Beasley, X. Zhong, J. Deng, R. Brooks, and G. K. Venayagamoorthy, "A survey of electric power synchrophasor network cyber security," *IEEE PES Innovative Smart Grid Technologies Conference Europe*, pp. 1–5, 2014.

[14] I. E. Evangeliou, "Vulnerabilities of the modbus protocol," Ph.D. dissertation, University of Piraeus, Greece, 2018.

[15] S. Aoufi, A. Derhab, and M. Guerroumi, "Survey of false data injection in smart power grid: Attacks, countermeasures and challenges," *Journal of Information Security and Applications*, vol. 54, 2020.

[16] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on scada systems," in *CPSCom: Proceedings of the 4th IEEE International Conference on Cyber, Physical and Social Computing*, 2010, pp. 380–388.

[17] N. Tripathi and N. Hubballi, "Application layer denial-of-service attacks and defense mechanisms: A survey," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–30, 2021.

[18] B. Phillips, E. Gamess, and S. Krishnaprasad, "An evaluation of machine learning-based anomaly detection in a scada system using the modbus protocol," in *ASM-SE'20: Proceedings of the 2020 ACM Southeast Conference*, 2020, pp. 188–196.

[19] M. Anwar, L. Lundberg, and A. Borg, "Improving anomaly detection in scada network communication with attribute extension," *Energy Informatics*, vol. 5, no. 1, pp. 1–22, 2022.

[20] A. Kleinmann and A. Wool, "Automatic construction of statechart- based anomaly detection models for multi-treaded industrial control systems," *ACM Transactions on Intelligent System Technology*, vol. 8, no. 4, pp. 1–21, 2017.

[21] "StationGuard," https://www.omicronenergy.com/en/solution/intrusion-detection-system-ids-for-the-power-grid/\#.

[22] O. N. Nyasore, P. Zavarsky, B. Swar, R. Naiyeju, and S. Dabra, "Deep packet inspection in industrial automation control system to mitigate attacks exploiting modbus/tcp vulnerabilities," in *2020 IEEE International Conference on Intelligent Data and Security (IDS)*, 2020, pp. 241–245.

[23] H. Lin, A. Slagell, C. Di Martino, Z. Kalbarczyk, and R. K. Iyer, "Adapting bro into scada: Building a specification-based intrusion detection system for the dnp3 protocol," in *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, 2013, pp. 1–4.

[24] Z. IDS, "https://old.zeek.org/manual/2.5.5/broids/index.html."

[25] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious scada communications," in *ICMLA'13: Proceedings of the 12th international conference on machine learning and applications*, 2013, pp. 54–59.

[26] S. D. Anton, S. Kanoor, D. Fraunholz, and H. D. Schotten, "Evaluation of machine learning-based anomaly detection algorithms on an industrial modbus/tcp

data set," in *ARES'18: Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–10.

[27] L. A. Maglaras, J. Jiang, and T. Cruz, "Integrated ocsvm mechanism for intrusion detection in scada systems," *Electronics Letters*, vol. 50, no. 25, pp. 1935–1936, 2014.

[28] S. D. D. Anton, S. Sinha, and H. D. Schotten, "Anomaly-based intrusion detection in industrial data with svm and random forests," in *2019 International conference on software, telecommunications and computer networks (SoftCOM)*. IEEE, 2019, pp. 1–6.

[29] X. Yan, Y. Jin, Y. Xu, and R. Li, "Wind turbine generator fault detection based on multi-layer neural network and random forest algorithm," in *2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*. IEEE, 2019, pp. 4132–4136.

[30] V. K. Singh and M. Govindarasu, "Decision tree based anomaly detection for remedial action scheme in smart grid using pmu data," in *2018 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2018, pp. 1–5.

[31] I. Elgarhy, M. M. Badr, M. Mahmoud, M. M. Fouda, M. Alsabaan, and H. A. Kholidy, "Clustering and ensemble based approach for securing electricity theft detectors against evasion attacks," *IEEE Access*, 2023.

[32] I. Kiss, B. Genge, P. Haller, and G. Sebestyén, "Data clustering-based anomaly detection in industrial control systems," in *ICCP'14: Proceedings of IEEE 10th International Conference on Intelligent Computer Communication and Processing*, 2014, pp. 275–281.

[33] A. Almalawi, A. Fahad, Z. Tari, A. Alamri, R. AlGhamdi, and A. Y. Zomaya, "An efficient data-driven clustering technique to detect attacks in scada systems," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 893–906, 2016.

[34] H. Yang, L. Cheng, and M. C. Chuah, "Deep-learning-based network intrusion detection for scada systems," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 1–7.

[35] G. Fenza, M. Gallo, and V. Loia, "Drift-aware methodology for anomaly detection in smart grid," *IEEE Access*, vol. 7, pp. 9645–9657, 2019.

[36] S. Y. Diaba, T. Anafo, L. A. Tetteh, M. A. Oyibo, A. A. Alola, M. Shafie-Khah, and M. Elmusrati, "Scada securing system using deep learning to prevent cyber infiltration," *Neural Networks*, vol. 165, pp. 321–332, 2023.

[37] L. A. C. Ahakonye, C. I. Nwakanma, J.-M. Lee, and D.-S. Kim, "Agnostic ch-dt technique for scada network high-dimensional data-aware intrusion detection system," *IEEE Internet of Things Journal*, vol. 10, no. 12, pp. 10 344–10 356, 2023.

[38] D. Wilson, Y. Tang, J. Yan, and Z. Lu, "Deep learning-aided cyber-attack detection in power transmission systems," in *2018 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2018, pp. 1–5.

[39] A. Hijazi, A. El Safadi, and J.-M. Flaus, "A deep learning approach for intrusion detection system in industry network." in *BDCSIntell*, 2018, pp. 55–62.

[40] S. Potluri and C. Diedrich, "Deep learning based efficient anomaly detection for securing process control systems against injection attacks," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 854–860.

[41] J. Suaboot, A. Fahad, Z. Tari, J. Grundy, A. N. Mahmood, A. Almalawi, A. Y. Zomaya, and K. Drira, "A taxonomy of supervised learning for idss in scada environments," *ACM Computing Surveys*, vol. 53, no. 2, 2020.

[42] C. Markman, A. Wool, and A. A. Cardenas, "A new burst-dfa model for scada anomaly detection," in *CPS '17: Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, 2017, p. 1–12.

[43] F. Martinelli, F. Mercaldo, A. Santone, C. Tavolato-Wötzl, and P. Tavolato, "Timed automata networks for scada attacks real-time mitigation," 2019.

[44] R. Udd, M. Asplund, S. Nadjm-Tehrani, M. Kazemtabrizi, and M. Ekstedt, "Exploiting bro for intrusion detection in a scada system," in *CPSS '16: Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, 2016, pp. 44–51.

[45] S. Zhanwei and L. Zenghui, "Abnormal detection method of industrial control system based on behavior model," *Computers & Security*, vol. 84, pp. 166–178, 2019.

[46] L. Chen and X. Wang, "Quickest attack detection in smart grid based on sequential monte carlo filtering," *IET Smart Grid*, vol. 3, no. 5, pp. 686–696, 2020.

[47] I. Burgetová, P. Matoušek, and O. Ryšavỳ, "Anomaly detection of ics communication using statistical models," in *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE, 2021, pp. 166–172.

[48] S. Bhattacharjee, M. J. Islam, and S. Abedzadeh, "Robust anomaly based attack detection in smart grids under data poisoning attacks," in *Proceedings of the 8th ACM on Cyber-Physical System Security Workshop*, 2022, pp. 3–14.

[49] C.-Y. Lin, S. Nadjm-Tehrani, and M. Asplund, "Timing-based anomaly detection in scada networks," in *CRITIS'17: Proceedings of the 12th International Conference Critical Information Infrastructures Security*, 2018, pp. 48–59.

[50] C.-Y. Lin and S. Nadjm-Tehrani, "Timing patterns and correlations in spontaneous scada traffic for anomaly detection." in *RAID'19: Proceedings of 22nd International Symposium on Research in Attacks, Intrusions and Defenses*, 2019, pp. 73–88.

[51] A. Bhardwaj, F. Al-Turjman, M. Kumar, T. Stephan, and L. Mostarda, "Capturing-the-invisible (cti): Behavior-based attacks recognition in iot-oriented industrial control systems," *IEEE access*, vol. 8, pp. 104 956–104 966, 2020.

[52] C.-Y. Lin and S. Nadjm-Tehrani, "Protocol study and anomaly detection for server-driven traffic in scada networks," *International Journal of Critical Infrastructure Protection*, vol. 42, p. 100612, 2023.

[53] A. Sidhik, "Different types of scada protocols," 2023. [Online]. Available: https://forumautomation.com/t/different-types-of-scada-protocols/4194

[54] R. Mattioli and K. Moulinos, "Communication network interdependencies in smart-grids," Techncal Report, Tech. Rep. [Online]. Available: https://www.enisa.europa.eu/publications/communication-network-interdependencies-in-smart-grids

[55] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Modeling modbus tcp for intrusion detection," in *Blackhat'15: 2015 Blackhat Conference*, 2015, pp. 1–9.

[56] R. Spenneberg, M. Brüggemann, and H. Schwartke, "Plc-blaster: A worm living solely in the plc," in *BlackhatAsia'16: 2016 Blackhat in Aisa Conference*, 2016, pp. 1–16.

[57] M. Alanazi, A. Mahmood, and M. J. M. Chowdhury, "Scada vulnerabilities and attacks: A review of the state-of-the-art and open issues," *Computers & Security*, vol. 125, no. C, 2023.

[58] P. M. Laso, D. Brosset, and J. Puentes, "Analysis of quality measurements to categorize anomalies in sensor systems," in *2017 Computing Conference*, 2017, pp. 1330–1338.

[59] "ICS Dataset for Smart-Grid Anomaly Detection, https://ieee-dataport.org/documents/ics-dataset-smart-grid-anomaly-detection#files (accessed on 09-aug-2023)."

[60] K. Boakye-Boateng, A. A. Ghorbani, and A. H. Lashkari, "Cic modbus dataset 2023," 2023. [Online]. Available: https://www.unb.ca/cic/datasets/modbus-2023.html

[61] I. Frazão, P. Abreu, T. Cruz, H. Araújo, and P. Simões, "Cyber-security modbus ics dataset," 2019. [Online]. Available: https://dx.doi.org/10.21227/pjff-1a03

[62] O. Alexander, M. Belisle, and J. Steele, "Mitre att&ck for industrial control systems: Design and philosophy," *The MITRE Corporation: Bedford, MA, USA*, vol. 29, 2020.

[63] V. Jacobson, S. McCanne, and M. Vetterli, *The TCPDump Manual*, Lawrence Berkeley Laboratory, University of California, Berkeley, 1989. [Online]. Available: https://www.tcpdump.org/tcpdump_man.html

[64] U. Lamping and E. Warnicke, "Wireshark user's guide," *Interface*, vol. 4, no. 6, p. 1, 2004.

[65] T. H. Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The express data path: Fast programmable packet processing in the operating system kernel," in *CoNEXT '18: Proceedings of the 14th International Conference on Emerging Networking Experiments and Technologies*, 2018, pp. 54–66.