Received November 7, 2019, accepted November 25, 2019, date of publication December 6, 2019, date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2958284

On the Generation of Anomaly Detection Datasets in Industrial Control Systems

ÁNGEL LUIS PERALES GÓMEZ[1], LORENZO FERNÁNDEZ MAIMÓ[1], ALBERTO HUERTAS CELDRÁN[2], FÉLIX J. GARCÍA CLEMENTE[1], CRISTIAN CADENAS SARMIENTO[3], CARLOS JAVIER DEL CANTO MASA[3], AND RUBÉN MÉNDEZ NISTAL[3]

[1]Departamento de Ingeniería y Tecnología de Computadores, University of Murcia, 30100 Murcia, Spain
[2]Telecommunications Software and Systems Group, Waterford Institute of Technology, Waterford, X91 K0EK Ireland
[3]Instituto Nacional de Ciberseguridad de España, INCIBE, 24005 León, Spain

Corresponding author: Ángel Luis Perales Gómez (angelluis.perales@um.es)

This work was supported in part by the Spanish Ministry of Science, Innovation and Universities under Grant RTI2018-095855-B-I00, and in part by the Government of Ireland, through the IRC Post-Doc Fellowship under Grant GOIPD/2018/466.

ABSTRACT In recent decades, Industrial Control Systems (ICS) have been affected by heterogeneous cyberattacks that have a huge impact on the physical world and the people's safety. Nowadays, the techniques achieving the best performance in the detection of cyber anomalies are based on Machine Learning and, more recently, Deep Learning. Due to the incipient stage of cybersecurity research in ICS, the availability of datasets enabling the evaluation of anomaly detection techniques is insufficient. In this paper, we propose a methodology to generate reliable anomaly detection datasets in ICS that consists of four steps: attacks selection, attacks deployment, traffic capture and features computation. The proposed methodology has been used to generate the Electra Dataset, whose main goal is the evaluation of cybersecurity techniques in an electric traction substation used in the railway industry. Using the Electra dataset, we train several Machine Learning and Deep Learning models to detect anomalies in ICS and the performed experiments show that the models have high precision and, therefore, demonstrate the suitability of our dataset for use in production systems.

INDEX TERMS Anomaly detection, critical infrastructures, industrial control, industrial control systems, industry applications, machine learning.

I. INTRODUCTION

Industrial Control Systems (ICS) are in charge of carrying out the management and supervision of industrial processes performed by critical infrastructures in industries such as electric, water, natural gas or chemical [1]. In the last years, the adoption of IP technologies in industrial devices and the connection of ICS to the Internet have influenced the increment of cyberattacks [2]. The most famous cyberattack affecting an ICS was in June 2010, when the Stuxnet worm was discovered [3]. Stuxnet spied and reprogrammed industrial systems controlling centrifuges of the Iran nuclear power plant. Another relevant malware discovered in 2016 was Irongate, a Stuxnet type malware that was used to hack the Siemens ICS [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Yu.

The increment of attacks affecting ICS, as well as their consequences, are influencing the use of Intrusion Detection Systems (IDS) in the field of ICS. IDS are responsible for both monitoring the environment in which they act and triggering alerts in case of detecting suspicious activity or anomalies in the network traffic. An extensive number of intrusion detection techniques have been proposed in the literature to tackle security threats. Nowadays, the techniques achieving the best performance are based on Machine Learning (ML) [5] and, more recently, Deep Learning (DL) [6], [7].

The performance of the previous techniques is measured using datasets that contain relevant data (network traffic, sensor and actuators logs, or features from previous sources) of an ICS scenario where several attacks are running. In this context, the quality of the datasets is key to evaluate the different detection techniques. However, due to the incipient stage of the cybersecurity research in ICS, there are several open challenges in the creation of such datasets. Among them, we highlight the following ones:

- The need for a methodology to generate reliable datasets in ICS. The reduced number of dataset-oriented methodologies results in limited availability of ICS datasets.
- The lack of datasets for anomaly detection in ICS containing realistic industrial traffic. The limited number of realistic datasets hinders research in IDS for ICS since many of the existing datasets are based on either simulations or laboratories that try to reproduce the reality.
- The reduced number of reliable datasets in specific ICS scenarios. The high specialization of ICS makes impossible the datasets exchange between different scenarios. As a result, a dataset obtained from a water management ICS is not appropriate for smartgrids.

With the goal of improving the previous challenges, the main contribution of this paper is a methodology to generate reliable anomaly detection datasets in heterogeneous ICS scenarios. The proposed methodology is composed of four steps: selecting attacks, deploying attacks, capturing network traffic and computing features. As a result of our methodology, we created a new publicly available ICS dataset called Electra [8], which has been generated from the network traffic of an electric traction substation running in both normal and under attack ways. The Electra dataset has been created in a realistic scenario with industrial devices such as Programmable-Logic Controllers (PLCs) and a SCADA system communicated by means of well-known industrial protocols such as S7Comm and Modbus. Finally, the paper presents several anomaly detection models based on ML and DL techniques that are applied on the Electra dataset to demonstrate its suitability in production systems.

The rest of the paper is structured as follows. Section II presents the state of the art in relation to existing datasets as well as ML and DL anomaly detection techniques for ICS. In Section III we present our methodology to obtain anomaly detection datasets in ICS. In Section IV, the process of dataset generation is explained. This section is divided into the following subsections: subsection IV-A details the testbed used in the dataset generation and subsection IV-B presents the implementation details regarding attacks deployment and traffic capture. In Section V a study of the feature distribution in the dataset generated is presented. In Section VI the results in anomaly detection performance are depicted. Finally, the conclusions and future work are presented in Section VII.

II. RELATED WORK

This section reviews the most relevant works existing in the literature in the field of datasets and artificial intelligent techniques for anomaly detection in ICS. A summary of these works is shown in TABLE 1.

A. DATASETS

KDDCup99 [9] is a widely used dataset to evaluate anomaly detection techniques. It is an evolution of the DARPA98 dataset [10] and includes a variety of intrusions simulated in a military network environment. Although KDDCup99 presents important deficiencies (specifically the duplicity of records in train and test sets [11]) it is still widely popular when evaluating intrusion detection techniques. The NSL-KDD [12] dataset was created to solve the KDDCup99 deficiencies in terms of duplicated records in both train and test sets. These datasets share the same attacks classified in four categories: DoS (Denial of Service), R2L (Root to Local), U2R (User to Root) and probing.

CTU-13 Dataset [13] is another dataset oriented to botnet detection. It contains real traffic captured from the Czech Technical University and includes malicious traffic from different bots, merged with normal and background traffic. Conversely, CICIDS 2017 Dataset [14] is collected from a simulated scenario with two separate networks: the outside and inside networks, made up of 4 attacking computers and 14 victim computers respectively.

Unlike traditional networks, in ICS the number of datasets is limited. In the following, we summarize the most relevant works found in the literature concerning industrial datasets.

In [15], three datasets for different ICS are provided, namely Power System Dataset [16], Gas Pipeline Dataset [17] and Water Storage Dataset [18]. The first is oriented to power systems and includes logs from Snort and measurements from the Synchrophasor. The second focuses on gas pipelines and includes features from Remote Terminal Units (RTU) streams. The last focuses in water tank on which three different attacks were launched: reconnaissance, false data injection and Denial-of-Service (DoS) attacks [19].

The Center for Cybersecurity Research, iTrust [20], also developed several datasets for different ICS. The most important datasets they developed are: Secure Water Treatment (SwaT) [21], Water Distribution (WADI) [22] and Electric Power and Intelligent Control (EPIC) [23]. The three datasets contain normal and malicious samples from false injection attacks. All of them are available upon prior request.

It is also worth mentioning the BATADAL [24] (BATtle of the Attack Detection ALgorithms) competition, which is carried out to propose algorithms for the detection of cyber-attacks affecting industrial environments. This competition provides participants with a dataset consisting of two training subsets. The first of them does not contain attacks and was generated from a simulation running for one year. The second training subset was collected for 6 months and contains data captured during the operation of the plant under several attacks.

In summary, the majority of the ICS-oriented datasets focus on false data injection attacks, neglecting other harmful attacks like replay attacks that inject legitimate packets previously seen on the network. This packet injection can cause serious damage if the packet inter-arrival time is critical for the system. In our proposed Electra dataset, we have included three types of attacks: false data injection, replay, and reconnaissance. As can be seen in TABLE 1, Electra is the only available dataset that includes replay attacks and the only

TABLE 1. Comparison of datasets.

Dataset	Provides	Scenario	Protocols	Attacks	ML & DL Related Work
Darpa98 [10]	Packets	Traditional network	TCP/IP Protocols	DoS R2L U2R Probe	[25]
KDDCup99 [9]	Precomputed features from network traffic	Traditional network	TCP/IP Protocols	DoS R2L U2R Probe	[26]
NSL-KDD [12]	Precomputed features from network traffic	Traditional network	TCP/IP Protocols	DoS R2L U2R Probe	[26]
CTU-13 [13]	Packets without payload Flows	Traditional network	TCP/IP Protocols	Botnet	[6] [27]
CICIDS 2017 [14]	Packets Flows	Traditional network	TCP/IP Protocols	Brute Force DoS and DDoS Heartbleed Web Attack Infiltration Botnet	[28]
SWAT [21]	Packets Sensors/Actuators state	ICS	CIP EtherNet/IP	False Data Injection	[29] [30] [31]
WADI [22]	Sensors/Actuators logs	ICS	EtherNet/IP Modbus RS485 CIP HSPA	False Data Injection	[32] [30] [31]
EPIC [23]	Packets Sensors/Actuators state	ICS	-	False Data Injection	
Power System [16]	Logs Sensors/Actuator state	ICS	-	False Data Injection	[16]
Gas Pipeline [17]	Precomputed features from RTU Telemetry	ICS	-	False Data Injection DoS	[33] [34]
Water Tank [18]	Precomputed features	ICS	Serial Modbus DNP3	False Data Injection DoS Reconnaissance	[35]
BATADAL [24]	Sensors/Actuators state	ICS	-	False Data Injection	[31] [30]
Electra	Precomputed features from network traffic	ICS	Modbus TCP S7Comm	False Data Injection Replay Reconnaissance	

based on S7Comm protocol, which is widespread in industrial applications. In addition, Electra dataset was obtained from a real scenario, specifically an electric traction station used in the railway industry.

B. ANOMALY DETECTION

In the literature, several ML and DL solutions have been proposed to detect anomalies in traditional networks. In this context, the authors of [26] propose a stacked non-symmetric deep auto-encoder for feature extraction followed by a Random Forest in the classification stage, using KDDCup99 and NSL-KDD to evaluate it. Regarding botnet detection, authors of [6] present a novel 5G-oriented self-adaptive cyberdefense architecture. CTU-13 dataset was used in the experiments demonstrating that the proposed architecture obtains sufficient classification accuracy in anomaly detection. Another relevant work using CTU-13 dataset is [27], where authors present a Mobile Edge Computing (MEC) oriented solution for 5G mobile networks to detect network anomalies in real-time. Another work focused on anomaly detection is [28], where the authors use Fisher Score to select relevant features from CICIDS 2017 dataset. A novel approach called Hierarchical Spatial-Temporal features-based Intrusion Detection System (HAST-IDS) is proposed in [25]. HAST-IDS first learns the low-level spatial features using a Convolutional Neural Network (CNN), and then learns high-level temporal features using Long Short-Term Memory neural networks (LSTM). DARPA98 is used in HAST-IDS to evaluate the models. Additionally, a detailed survey of ML and DL methods applied to intrusion detection in traditional networks can be found in [36].

Regarding the ICS anomaly detection, an unsupervised machine learning approach to anomaly detection is proposed in [29]. They use SWaT dataset to compare two different methods: Deep Neural Network (DNN) and Support Vector Machine (SVM). In [30] and [31], the authors present an approach based on 1D Convolutional Neural Networks (1D-CNN) and auto-encoders. They apply both methods in time and frequency domains using BATADAL, SWaT and WADI datasets. Another unsupervised approach is presented in [32]. Specifically, the authors propose an unsupervised multivariate anomaly detection method based on Generative Adversarial Networks (MAD-GANs). They use LSTM to capture temporal correlations of time series distributions and evaluate the system using SWaT and WADI datasets. Another study of several ML models in ICS anomaly detection is carried out in [16]. The authors evaluate seven different models (OneRule, Nearest Neighbor, Random Forests, Naive Bayes, SVM, JRip and Adaboost+JRip) using the Power System Dataset. Other relevant works are [33] and [34]. In the former, the authors propose two approaches based on one-class classification algorithms: the Support Vector Data Description (SVDD) and the Kernel Principal Component Analysis (KPCA). In the latter, the authors apply a Bloom filter to store the signature database used for packet-based anomaly detection and use an LSTM network to time series anomaly detection. In both works, the authors use the Gas Pipeline Dataset to train and evaluate their solutions. Finally, we highlight [35] where authors present a neural network trained to detect false data injection and denial of service attacks, using the Water Tank dataset to evaluate the neural network.

In conclusion, the majority of ML and DL models applied to industrial environments consider the states of sensors and actuators or the logs generated by monitoring tools, instead of network traffic. In this work, we evaluate different ML and DL models, both supervised and semi-supervised, in the context of anomaly detection in industrial environments using features extracted only from network traffic. The models selected to carry out our evaluation are One-Class SVM, Isolation Forest, SVM, Random Forest and Neural Network. To evaluate these models we have used our Electra Dataset .

III. METHODOLOGY PROPOSED

To overcome the lack of ICS anomaly detection datasets, we propose a methodology to obtain an anomaly detection dataset consisting of a 4-step process which is detailed in FIGURE 1. The first step is the selection of the attacks to be launched on the testbed. In the second step, it is decided how to launch the attacks and which devices will be affected by them. The third step is to capture the network traffic generated by all devices. Finally, the fourth step consists in computing the features from the network capture and generate the dataset. The following subsections detail each of these steps.

A. ATTACKS SELECTION

The main vulnerabilities of control protocols are the lack of authentication mechanism, data encryption and data integrity checking. These vulnerabilities allow us to select attacks based on packet content modification and attacks that can be launched from unauthenticated nodes. In order to simplify the selection of attacks, we classify them into the following categories [37]:

- **Reconnaissance attacks:** They are aimed at identifying potential victims within a network. After gaining access to an industrial network, an attacker needs to perform a reconnaissance attack to get information about the different devices that are in the network and their associated services. From this information, the attacker usually plans the next move.
- **False data injection attacks:** They are aimed at controlling the control devices of an ICS. To achieve this goal, these attacks are based on sending altered data through control protocols. False data injection attacks can be classified depending on the data they alter into: Command Injection, Response Injection and Parameter Injection, just to name a few.
- **Replay attacks:** They are based on retransmitting valid packets that have been previously seen in the network. This action can alter the normal reception rate of that packet in a device or can mislead a device by sending an unexpected but apparently valid packet to it. In ICS, these attacks can have as much impact as the false data injection attacks, but they are harder to detect because the packet payload is valid.

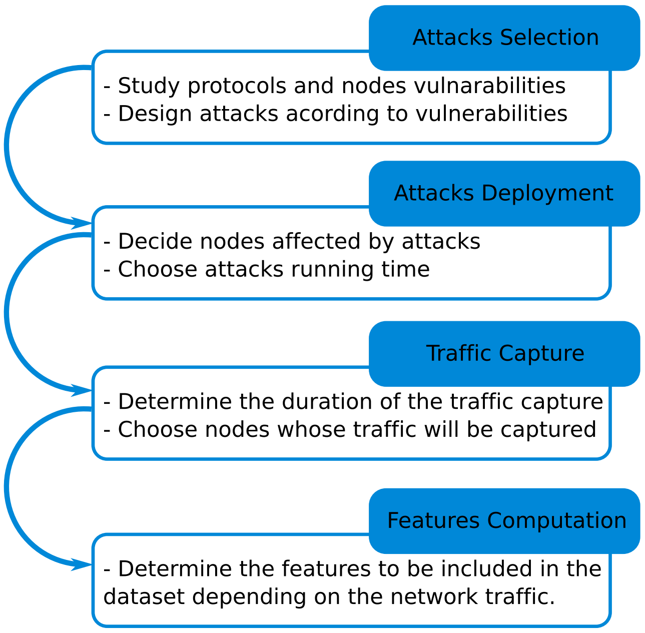

Attacks Selection
- Study protocols and nodes vulnarabilities
- Design attacks acording to vulnerabilities
Attacks Deployment
- Decide nodes affected by attacks
- Choose attacks running time
Traffic Capture
- Determine the duration of the traffic capture
- Choose nodes whose traffic will be captured
Features Computation
- Determine the features to be included in the dataset depending on the network traffic.

FIGURE 1. Methodology flow diagram.

In particular, we define a specific set of attacks for each category that is summarized in TABLE 2. For recognition attacks, we propose to generate malicious packets to scan all possible function codes in the PLC under attack. In the case of false data injection attacks, we propose two different set

TABLE 2. Attacks designed.

Category	Attack	Type
Reconnaissance	Function codes recognition	Packet creation
False data injection	Response modification	Packet modification
	Forced error in response	Packet modification
	Command modification	Packet modification
	Read data	Packet creation
	Write data	Packet creation
Replay	Replay valid packets	Packet creation

of attacks: 1) those that create packets to perform spurious writes or reads in valid memory addresses of a PLC; 2) those that maliciously modify existing packets with the goal of altering the data returned by the slave, forcing an error in the response returned by the slave, or changing the command message sent by the master. For replay attacks, we propose to retransmit packets previously generated by the master or the slaves.

B. ATTACKS DEPLOYMENT

The attacks must be deployed by introducing a new node into the control network. This node is in charge of developing the set of attacks on control protocols and, therefore, it must be able to create new packets and modify the existing ones. The deployment period must be long enough to have a representative amount of traffic from attacks.

C. TRAFFIC CAPTURE

Once the attacks are launched in the testbed and malicious traffic begin to be generated, the traffic capture is activated. The capture process comprises the traffic from all devices that are part of the testbed, and includes both normal and malicious traffic. The traffic capture duration must be sufficient to deploy all the selected attacks several times over the devices. The goal of the capture process is to generate rich and varied network traffic.

D. FEATURES COMPUTATION

The last step is to compute the set of the most discriminating features for anomaly detection from the traffic capture. In this step, we have to decide whether to use packet-level features or flow-level features. Packet-level features are suitable in industrial scenarios using protocols without security mechanisms such as authentication, encryption and integrity check. However, when secure protocols are used, an approach based on flow-level features, computed as aggregations of measurements extracted from a group of network flows, is more appropriate.

IV. ELECTRA DATASET GENERATION

This section illustrates the configuration of the industrial testbed and the most relevant implementation details to generate Electra dataset.

A. INDUSTRIAL TESTBED

An electric traction substation employed in a real high-speed railway industry is used as our testbed. The main purpose of this testbed is to allow converting the electric power of the general network to voltage, current, and frequency conditions to supply railways or trams. This system can be used to convert the three-phase alternating current into single phase with the lower frequency needed for railway electrification systems. To accomplish its task, the Electric Traction Substation shown in FIGURE 2 has 5 PLCs (1 master PLC and 4 slave PLCs) and a SCADA system. Additionally, the testbed has a switch (D5) for the interconnection

TABLE 3. Configuration of the devices in the testbed.

Device	MAC	IP	Protocol	Role
A1	::C1:41:1B	10.70.38.51	Modbus	Master
			S7Comm	Slave
A3	::E1:DD:58	10.70.38.55	Modbus	Slave
A2	::E1:DE:9C	10.70.38.56	Modbus	Slave
D1	::A0:31:EA	10.70.38.52	S7Comm	Master
D3	::A2:39:48	10.70.38.53	S7Comm	Slave
D2	::A2:38:D7	10.70.38.54	S7Comm	Slave

of the different devices and a firewall (D4) to protect the substation from attacks coming from outside. The testbed devices communicate through control protocols following a master-slave architecture, where the master initiates the communication requesting some data and a slave replies with the requested information. The network communication is carried out through the following protocols: Modbus TCP [38], OPC [39] and S7Comm [40]. The SCADA system consists of a Nanobox (A1) and an HMI (A4) that communicate through the OPC protocol. The SCADA acts as a master of both Modbus slaves A2 and A3. Similarly, regarding the S7Comm protocol, D1 PLC acts as the master of A1, D2 and D3 PLCs. TABLE 3 details the IP, MAC and protocol used for each device.

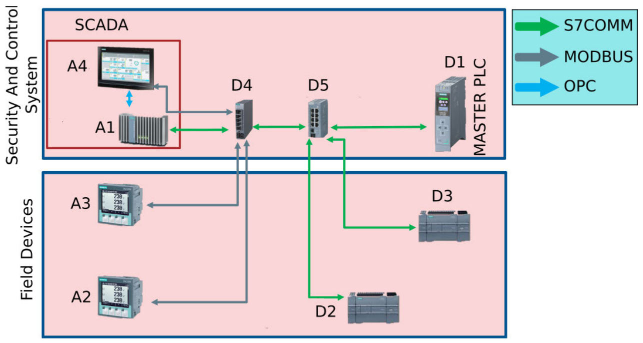


FIGURE 2. Industrial testbed used to launch attacks and collect network traffic.

In this work, we focused on the control network protocols, i.e. Modbus and S7comm, to generate our dataset. It is worth mentioning that they are the most widespread in industrial applications. Both Modbus and S7Comm have a master-slave model and work over TCP/IP.

B. IMPLEMENTATION DETAILS

This section describes the implementation details of the set of attacks, as well as the monitoring and capture of the traffic generated by them in order to develop the Electra dataset.

1) ATTACKS IMPLEMENTATION

To carry out the attacks, a new device was attached to the network with a man-in-the-middle (MitM) configuration. Specifically, this MitM node was configured to implement the false data injection attacks by poisoning the Address Resolution Protocol (ARP) table of network devices. In this way, the MitM node has access to all the messages exchanged by master and slave nodes. In particular, in our testbed, the attacked nodes for the Modbus protocol were A1 and A3, while the attacked nodes for the S7Comm protocol were D1 and D2. Note that the attack consisted of intercepting the communications of legitimate nodes to modify the exchanged messages, rather than infecting the nodes.

The replay and reconnaissance attacks were implemented in Python, requiring no additional libraries. In these attacks, the communication with the other legitimate nodes of the network was made with the socket class provided by the Python standard library. However, to carry out the MitM attacks, two additional libraries were needed:

- Scapy [41]. This library is used to modify the network traffic that passes through the node that is executing the MitM attack.
- Netfilterqueue [42]. Allows us to define iptables rules to redirect all the traffic that matches the rules to a memory area in user space that can be accessed using scapy.

2) TRAFFIC CAPTURE

As a first step, a port-mirroring configuration was implemented to capture all the packets in the network. To avoid that packets were dropped during traffic capture, Wireshark was configured to use a 200MB buffer. After that size, Wireshark saved the traffic capture in a file on disk. Multiple pcaps files with a maximum size of 200 MB were generated, being the whole capture size around 20GB. The traffic capture took slightly more than 12 hours. During this time, variants of every attack were repeated several times generating a rich attack traffic patterns. Once the traffic capture finished, it was processed to create two independent subsets: one containing the traffic flowing through the port 102 (S7comm) and the other containing the traffic flowing through the port 502 (Modbus).

3) FEATURES COMPUTATION

In this work, we propose the set of features listed in TABLE 4. These features are obtained by considering only the control protocol and MAC/IP addresses; consequently other header fields of Ethernet, TCP and IP protocols are discarded. We propose to include packet-level features, due to its highly valuable information to detect the majority of the attacks mentioned in the previous section, including those based on modifying or injecting data into the industrial network.

4) PROCESSING AND LABELLING

Each subset traffic capture was processed using the Python scapy library. The script examines all the packets contained in each traffic capture subset and extracts relevant features producing as output two Comma-Separated Values (CSV) files: one for Electra Modbus subset and another for Electra S7Comm subset. The list of features computed was detailed in Section IV-B3.

In the traffic capture, each packet may contain several reads or writes. In order to simplify the dataset, each multiple operation message was split into single operation messages. That is, each entry in the dataset contains only a single operation. For example, a packet carrying a 100-integer write operation in the processing phase was split into 100 different instances, one for each integer write operation.

TABLE 4. Selected features.

Feature	Description	Data type
time	Timestamp	String
smac	Source MAC address	String
dmac	Destination MAC address	String
sip	Source IP address	String
dip	Destination IP address	String
request	Indicates whether the packet is a request (packet from master to slave).	Boolean
fc	Function code	Integer
error	Indicates whether there has been an error in reading/writing operation	Boolean
madd	Memory address to perform read/write operation	Integer
data	In the case of a read operation, it indicates the data that the slave sends to the master. In the case of a write operation, it indicates the data that the master sends back to the slave	Integer
label	Label for attacks and normal samples	String

Finally, to label each sample, designed attacks were configured to establish different marks in unused fields of S7Comm and Modbus protocols. After that, the Python script processes each traffic capture subset by looking at the field and labeling the sample according to its mark.

V. ELECTRA DATASET DESCRIPTION

This section examines the feature distribution of each subset making up the Electra dataset: Electra Modbus and Electra S7Comm.

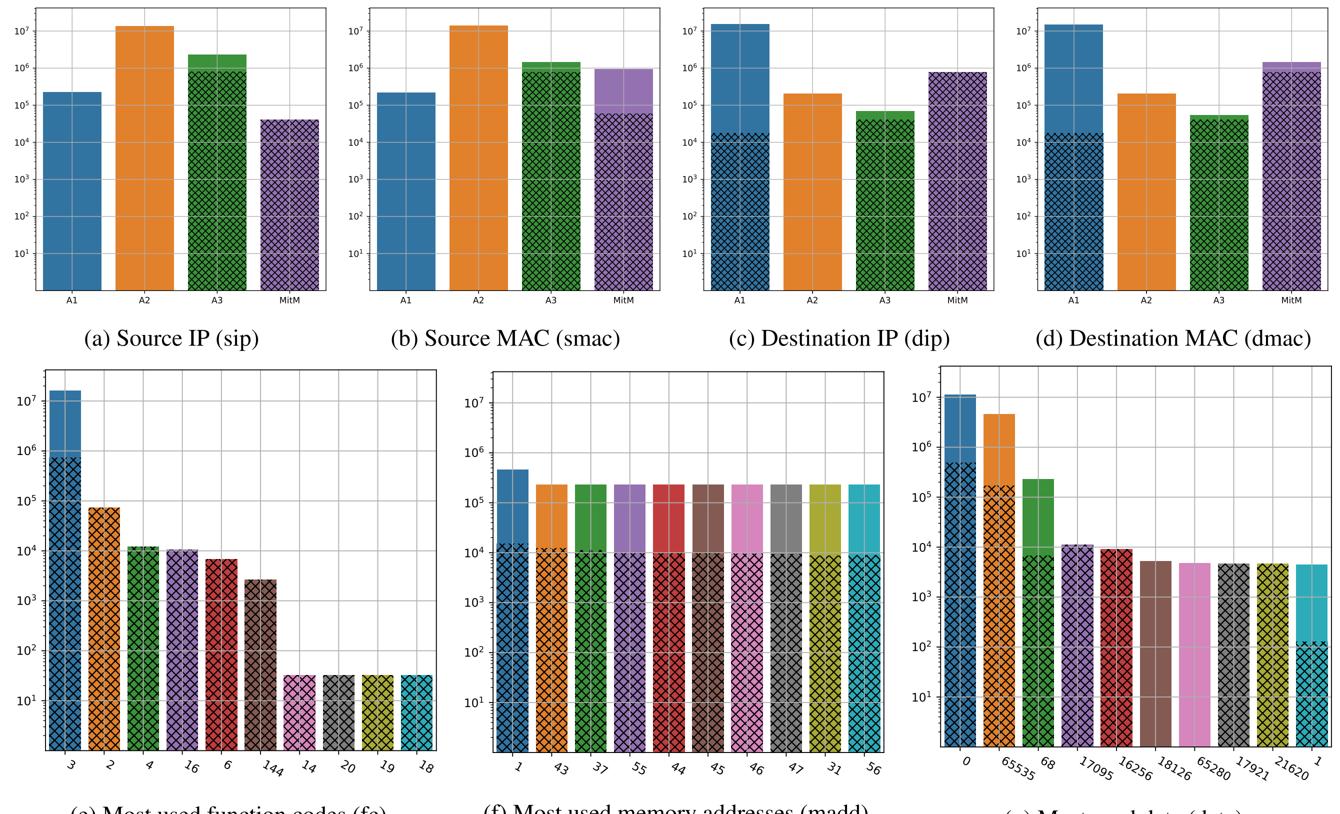
A. ELECTRA MODBUS SUBSET

In this section, we explore the distribution of each feature in the Electra Modbus subset. FIGURE 3 plots the total number of samples and the number of anomalous samples (shown by the pattern) for seven features listed in TABLE 4. The anomalous samples are classified in six anomaly classes that correspond to the attacks that MitM node deploy in the testbed. The proportion of each class, included the normal samples, can be seen in TABLE 5.

TABLE 5. Proportion of each class in Electra Modbus.

Classes	# of samples
Normal	94.8%
Function code recognition attack	0.19%
Response modification attack	0.1%
Force error in response attack	0.007%
Read attack	4.83%
Write attack	0.06%
Replay attack	0.006%

As expected and shown in FIGURE 3a and in FIGURE 3b, the source IP (sip) distribution is different from the source



(a) Source IP (sip) (b) Source MAC (smac) (c) Destination IP (dip) (d) Destination MAC (dmac)

(e) Most used function codes (fc) (f) Most used memory addresses (madd) (g) Most used data (data)

FIGURE 3. Modbus features distribution in log scale (total number of samples in plain color and number of anomalous samples shown in squared pattern). (a) and (c) represent the source IPs and MACs from which the messages have been sent. (b) and (d) detail the source IPs and MACs to which the messages are sent. (e) details the ten most frequent function codes in the traffic capture. (f) shows the ten most frequent memory addresses in the traffic capture. (g) details the ten most frequent data in the traffic capture.

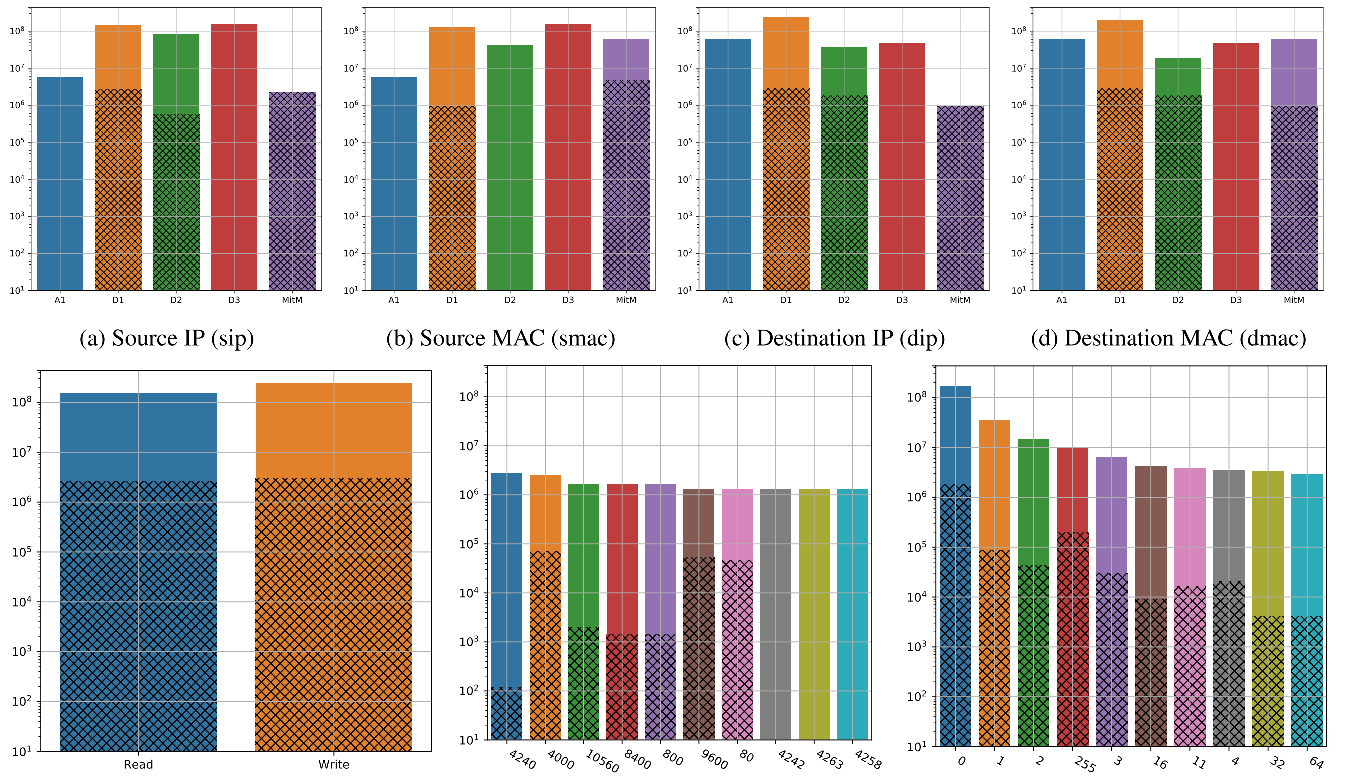
MAC (smac) distribution. Similarly, the distributions are slight different for destination IP (dip) and destination MAC (dmac), as we can observe in FIGURE 3c and FIGURE 3d. This is due to the introduction of MitM-based attacks that hijack a valid IP, while the MAC remains unchanged. This fact highlights the need of both IP-related and MAC-related features in our dataset.

Additionally, FIGURE 3a shows that A2 is the node that sends the greatest number of messages (more than 13 million) followed by A3 (over 2 million), whereas A1 (over 200 000) is the least active node emitting packets. However, as it is shown in FIGURE 3c and FIGURE 3d, the main network destination is A1 (more than 15 million messages). These distributions of messages for each node are the expected in a master/slave configuration when the most used operation is to read register values, as shown in FIGURE 3e.

Another important aspect shown in the four first graphs of FIGURE 3 is that A1 and A3 are the nodes attacked and, therefore, only those include anomalous messages. Moreover, the MitM node injects a significant amount of messages, that is over 41 000 messages transmitted considering its IP address and more than 900 000 considering its MAC address, and even it receives a higher amount of messages, that is more than 700 000 considering its IP address and more than 1.4 million considering it MAC address. The large difference between messages transmitted and received by its MAC and IP addresses is due to the MitM attack.

In FIGURE 3e, we can observe that read function codes (fc) are the most repeated, specifically, Read Holding Register (code 3), Read Discrete Input (code 2) and Read Input Register (code 4). It can be noted that only the Read Holding Register function is considered normal, whereas the other function codes belong to anomalous samples. The reason is essentially due to two factors. On the one hand, A2 and A3 are electrical meters that continuously perform electrical measurements stored in holding registers. From time to time, these measurements are required by A1, which uses the Read Holding Register function code to read them. On the other hand, the attacks design was performed considering a large number of attacks that can be carried out in complex scenarios, including read and write attacks over different types of register and memory addresses, resulting, in this particular subset, in the presence of a large number of function codes that are not present in normal traffic. TABLE 6 shows the well-known function name associated with the most frequent function codes in the dataset.

The distribution of memory addresses (madd) in FIGURE 3f shows that the most frequent value is 1 followed



(a) Source IP (sip) (b) Source MAC (smac) (c) Destination IP (dip) (d) Destination MAC (dmac)

(e) Most used function codes (fc) (f) Most used memory addresses (madd) (g) Most used data (data)

FIGURE 4. S7Comm features distribution in log scale (total number of samples in plain color and number of anomalous samples shown in squared pattern). (a) and (c) represent the source IPs and MACs from which the messages have been sent. (b) and (d) detail the source IPs and MACs to which the messages are sent. (e) details the ten most frequent function codes in the traffic capture. (f) shows the ten most frequent memory addresses in the traffic capture. (g) details the ten most frequent data in the traffic capture.

TABLE 6. Function names of the function codes shown in FIGURE 3e.

Function Code	Function Name
3 (0x03)	Read Holding Register
2 (0x02)	Read Discrete Input
4 (0x04)	Read Input Register
16 (0x10)	Write Multiple Registers
6 (0x06)	Write Single Register
144 (0x90)	Exception for Write Multiple Registers
20 (0x14)	Read File Record

by the other memory addresses. These addresses are used by the PLC to store the electrical measurement until the master requests them. Additionally, FIGURE 3g shows that among the most frequent message data, four of them are entirely generated by attacks (17095, 16256, 17921 and 21620) whereas two are generated completely by normal traffic in the testbed (18126 and 65280). The remaining messages data contain both normal and anomalous samples.

Finally, related to the distribution between normal and anomalous samples of Electra Modbus, we can observe that there are about 15 million normal samples and less than 1 million anomalous samples. Such imbalance is characteristic of anomaly detection datasets.

As a summary, Electra Modbus stands out for having been obtained from a testbed used in real industry applications, resulting in the availability of real and not simulated traffic. In spite of these desirable characteristics in a dataset oriented to anomaly detection, throughout this section we have seen how the variety of function codes is rather limited. However, the anomalous samples contained in this subset have been obtained from a variety of attacks carried out in real ICS. The result is that these attacks have generated a wide variety of traffic with function codes that are not present in normal traffic.

B. ELECTRA S7COMM SUBSET

In this section, we explore the distribution of each feature in the Electra S7Comm subset. Similarly to the previous section, FIGURE 4 plots the total number of samples and the number of samples belonging to the anomaly class for the seven features listed in TABLE 4. The network traffic pattern in this subset has several similarities to the Electra Modbus subset, mainly because the set of attacks is the same and both protocols have a master-slave architecture. The anomaly classes and their proportion can be seen in TABLE 7.

As expected and shown in FIGURE 4a and FIGURE 4b, the source IP (sip) and source MAC (smac) distributions are different. In the same way, the destination IP (dip) and destination MAC (dmac) distributions are also slight different,

TABLE 7. Proportion of each class in Electra S7Comm.

Classes	# of samples
Normal	98.58%
Response modification attack	0.04%
Force error in response attack	0.42%
Command modification attack	0.15%
Read attack	0.23%
Write attack	0.57%
Replay attack	0.007%

as shown in FIGURE 4c and FIGURE 4d. This behavior, as it happened in Modbus, is due to the deployment of MitM-baseed attacks, where the victim IP is hijacked and the MAC remains unchanged.

Additionally, according to IP distributions, FIGURE 4a and FIGURE 4b show that D3 and D1 are the most active nodes sending messages (150 and 145 million messages, respectively), whereas D2 and A1 are the most passive (80 million and 5 million, respectively). However, D1 is the main receiver (over 200 million), while A1, D3 and D2 have less activity (60 million, 50 million and 35 million, respectively) as shown in FIGURE 4c and FIGURE 4d. These distributions of messages for each node are the expected in a master-slave configuration, where the master (D1) controls the system operation sending write/read messages.

Focusing on the anomalous messages shown in the four first graphs of FIGURE 4, only D1 and D2 include them because they are the only attacked nodes. In the case of the MitM node, it transmits over 2.2 million messages and receives about 1 million messages. It can be observed in sip and dip distribution. These attacks are related to the packet creation (read data, write data and replay valid packets). However, taking into account the MitM attack, the MitM node transmits 62 million messages and receives 60 million messages, as shown in smac and dmac distributions.

In FIGURE 4e, we observe two different types of function codes (fc): read (150 million operations) and write (230 million operations). According the number of anomalies, write operations account for 3 million, whereas read operations account for 2.5 million. These anomalies are caused by both packet creation and packet modification launched by the MitM node.

Electra S7Comm involves a wide variety of memory addresses (madd) used in normal write/read operations that can be observed in FIGURE 4f. The five most common addresses in the normal traffic of our subset are 4 240, 4 000, 10 560, 8 400, and 800, with 2.7, 2.4, 1.6, 1.6 and 1.6 million messages respectively. For these memory addresses, the number of anomalies are: 121, 71 000, 1 900, 1 400 and 1 400 respectively.

FIGURE 4g shows the most frequent data content in S7Comm subset. As it can be seen, it offers a greater variety of normal data than the Electra Modbus. In this case, the most frequent data content is 0 (166 million and 2 million for normal and anomalous messages, respectively) followed by 1 (34 million and 100 000 million for normal and anomalous messages, respectively) and 2 (14 million and 40 000 for normal and anomalous messages, respectively). Both packets creation and modification attacks are the responsible of generating those data contents. The anomalous data is generated randomly for both attack type, but data in modified packets deviates slightly from the original packet, whereas data in created packets is entirely random.

Finally, in relation to the distribution between normal and anomalous samples of Electra S7comm, we can observe that there are about 381 million normal samples and about 55 million anomalous samples. Again, such imbalance is typical in anomaly detection dataset.

As a summary, Electra S7Comm has been captured from a testbed containing real and non-simulated traffic, including normal and infected traffic. The Electra S7Comm features discussed in this section make this subset suitable for use in anomaly detection using supervised techniques. In addition, Electra S7Comm includes a great variety of attacks that can be found in real ICS scenarios and presents a great variety in the distribution of its features.

C. DIMENSIONALITY REDUCTION

To conclude the description of the generated dataset, we applied Principal Component Analysis (PCA) [43] and t-Distributed Stochastic Neighbor Embedding (t-SNE) [44] in order to visualize the Electra subsets. Both methods are useful for visualization purposes, but PCA also provides dimensionality reduction of the dataset. In this work we applied both algorithms over all the features included in the Electra dataset, except the Time feature.

Due to the dataset size and its imbalance, it was required to extract a balanced subsample suitable for both algorithms. First, a balanced version of the dataset was created containing all the original anomalous samples as well as the same number of randomly selected normal samples. Then, the balanced dataset was subsampled, extracting a new dataset with a fraction of the samples (10% for Electra Modbus and 5% for Electra S7Comm). A preprocessing step was carried out by applying One-Hot Encoding (OHE) to the categorical data (MAC and IP addresses), yielding a binary feature for each different categorical value.

Hyper-parameter tuning is a crucial procedure to achieve a significant performance. Specifically, in t-SNE we performed a grid search over the following perplexity values [20, 30, 60, 100, 130, 160, 200, 300, 400, 600, 700, 900, 1200, 1500, 1700, 2000, 2300]. The selected perplexity was 2300 for Electra Modbus subset and 600 for Electra S7Comm subset. With regard to PCA, it does not require hyper-parameter tuning. In this case we just selected the two most meaningful dimensions.

The result of applying these algorithms to the Electra dataset is plotted in FIGURE 5 for Modbus subset and in FIGURE 6 for S7Comm subset. Regarding PCA in Electra Modbus, we can see that the normal clusters are focused on specific locations. In general, the anomalous clusters are

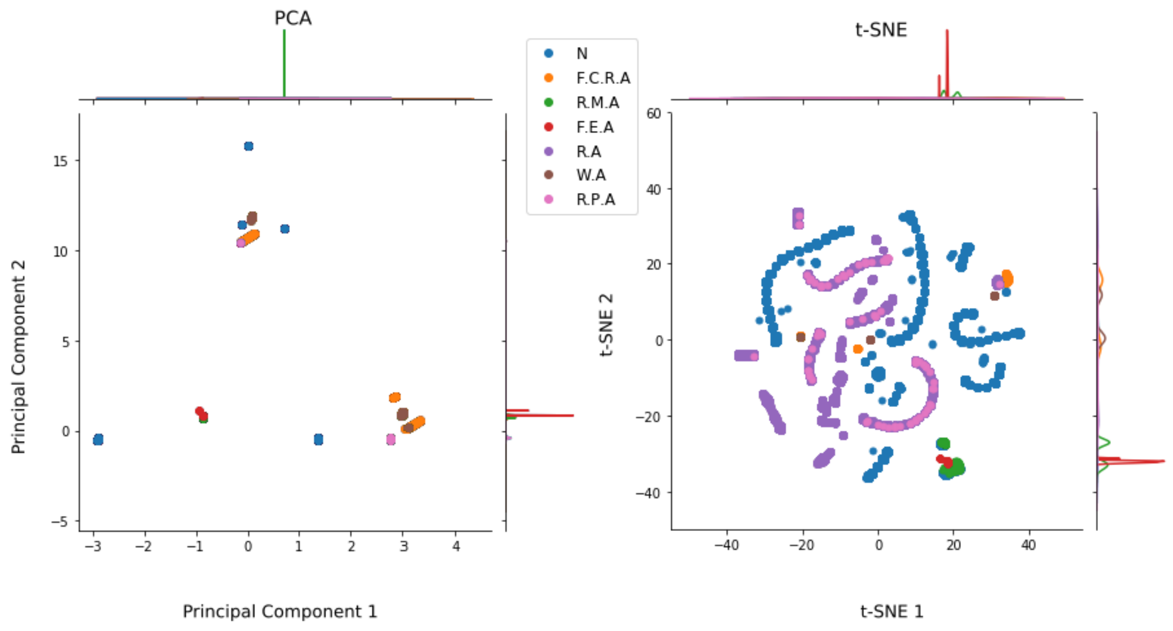


FIGURE 5. Dimensionality reduction for Modbus subset. Left: PCA algorithm applied over Modbus subset. Right: t-SNE algorithm applied over Modbus subset. Horizontal and vertical histograms show the distribution along each axis (N = Normal, F.C.R.A = Function Code Recognition Attack, R.M.A = Response Modification Attack, F.E.A = Force Error Attack, R.A = Read Attack, W.A = Write Attack, R.P.A = Replay Attack).

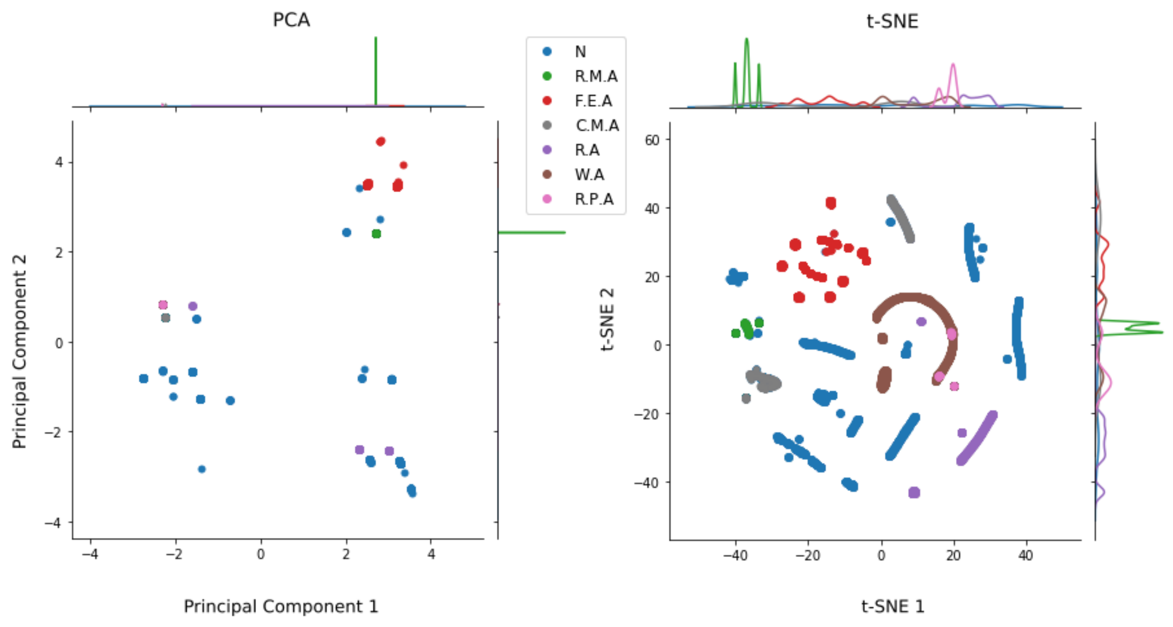


FIGURE 6. Dimensionality reduction for S7Comm subset. Left: PCA algorithm applied over S7Comm subset. Right: t-SNE algorithm applied over S7Comm subset. Horizontal and vertical histograms show the distribution along each axis (N = Normal, R.M.A = Response Modification Attack, F.E.A = Force Error Attack, C.M.A = Command Modification Attack, R.A = Read Attack, W.A = Write Attack, R.P.A = Replay Attack).

sufficiently isolated from the normal samples, with the exception of the top cluster where the attacks of function code recognition, write, read and replay are mixed with normal samples. The attacks of response modification and force error are located in the central cluster, whereas the right cluster is composed of anomalous samples generated from replay, write, read and function code recognition attacks. Regarding t-SNE, we can observe the same pattern as in PCA, i.e. normal samples are located in several clusters isolated from other clusters. Regarding attacks affecting the response, there is a cluster in the bottom right corner where force error and response modification attacks are located. Similarly, the replay attacks are located close to the read attacks. This is because all replay attacks were read messages and, moreover, they were launched from the MitM node. Regarding write attacks, they are distributed along the plot in their own clusters. Finally, in the center right, we can see a cluster where several types of attacks along with normal samples are located.

Likewise, the results offered by Electra S7Comm are quite similar. Regarding PCA in S7Comm subset we can observe several clusters composed of normal samples. As can be seen, the anomalous clusters are close to the normal ones. For example, attacks related to both force error in the response and response modification are located in the top right cluster, whereas replay, read and command modification attacks are located in the left cluster. It is worth mentioning that read attacks are also located in another cluster in the bottom right corner. Finally, write attacks are located in the same cluster that replay attacks. This is due to all replay attacks launched against the testbed were write messages from MitM node. On the other hand, in t-SNE plot, we can observe several normal clusters isolated from the anomalous clusters. As in PCA, the replay attacks are close to the write attacks due to the fact that the majority of the replay attacks carried out on S7Comm was related to writes. In contrast to Modbus subset, the attacks related to the response, that is, force error and response modification attacks, are mutually isolated. The former are located at the top center whereas the latter are located on the left. Regarding the command modification attacks, there are two different clusters, the first is located on the left, close to response modification samples, whereas the second is located at the top. Finally, the read attack samples have their own cluster located in the bottom right corner.

In this section we have relied on visualization techniques such as PCA and t-SNE to explain the separability between dataset classes. In general, both methods show that the normal and attack samples can potentially be separated in this context by a suitable classification/anomaly detection algorithm.

VI. EXPERIMENTS

This section describes the anomaly detection experiments applied on the Electra dataset. These experiments are intended to show that anomaly detection machine learning models are suitable for use in real industrial environments.

To address the anomaly detection problem, we studied two different approaches: supervised and semi-supervised. In supervised models it is necessary to have a fully labeled dataset like Electra. On the other hand, semi-supervised models require only a dataset containing normal samples. Therefore, to train our semi-supervised models we just need to remove all the samples belonging to the anomalous class from the dataset. We have studied four ML models: Random Forest (RF), Support Vector Machine (SVM), One-Class Support Vector Machine (OCSVM) and Isolation Forest (IF); and one DL model, Dense Neural Networks (DNN).

As a previous step to perform the experiments, the duplicated records of the Electra dataset must be removed.

TABLE 8. Samples of normal and anomalous classes in each subset after removing duplicates.

Electra Subset	Normal Samples	Anomalous Samples	Total Samples
Modbus	25 031	16 610	41 641
S7Comm	230 054	2 583 180	2 813 234

This initial step is required because control processes frequently repeat the same actions over time and, therefore, their repetitive nature is also observed in the network traffic, resulting in a large number of duplicated packets. The number of records contained in the dataset after removing duplicates is shown in TABLE 8. It can be observed that the number of anomalous samples is greater than the number of normal samples. There are mainly two causes: 1) the large amount of duplicated records removed and 2) the data content in the anomalous samples was randomly generated resulting in a low amount of duplicate records. Due to the large amount of data contained in the S7Comm subset, a distinction was made between ML and DL models in the training phase. To train ML models using the S7Comm subset, only a 10% of the dataset was used. However, the ML models for the Modbus subset were trained with all the data available. Similarly, in the case of DL models all the available data from each subset (Modbus and S7Comm) was used.

For a correct evaluation of the models, the Electra subsets were split into train and test sets. The train set was used exclusively to train the models, while the test set was used to evaluate them. The partition of the dataset subsets into train (80%) and test (20%) was made by means of a uniform random sampling.

A key aspect in the training of ML and DL models is the tuning of their hyper-parameters. This phase is frequently time consuming because a training process has to be carried out for a great number of combinations of hyper-parameters in order to find out the configuration with the best performance. In this work, we performed a grid-search in the hyper-parameter space. A description of the tested hyper-parameters is in the TABLE 9.

TABLE 9. Hyper-parameters tunned in Electra dataset anomaly detection.

Model	Hyper-Parameter	Range of Values
Random Forest	estimators	[100, 200, 300]
SVM	C	[0.01, 0.1, 1, 10, 100]
	gamma	[0.001, 0.01, 0.1, 1]
Neural Network	number of layers	[1, 2, 3]
	neurons per layer	[128, 64, 32]
One-Class SVM	nu	[0.01, 0.1 ,1]
	gamma	[0.01, 0.1, 1, auto]
Isolation Forest	estimators	[100, 200, 300]
	contamination	[0.01, 0.005, 0.1, 0.5, auto]

In order to evaluate which of the studied models had the best performance, we defined some metrics. In particular, anomaly detection problems usually show a great imbalance between the number of normal and anomalous samples; therefore, precision and recall metrics are usually preferred to accuracy. In our case, as shown in TABLE 8, due to the

TABLE 10. Performance obtained by the models trained.

Dataset	Model	Hyper-parameters	Precision	Recall	F1-score
Modbus	Random Forest	Estimators: 200	98.77	98.71	98.74
	SVM	C: 10 Gamma: 1	97.56	100	98.76
	Neural Network	Number of layers: 1 Neurons per layer: [128]	96.92	100	98.43
	OCSVM	Nu: 0.1 Gamma: 0.1	98.62	98.56	98.59
	Isolation Forest	Estimators:100 Contamination: 0.1	87.39	100	93.27
S7Comm	Random Forest	Estimators: 300	99.56	100	99.78
	SVM	C: 100 Gamma: 0.1	99.49	100	99.74
	Neural Network	Number of layers: 1 Neurons per layer: [128]	99.99	99.19	99.59
	OCSVM	Nu: 0.01 Gamma: 0.1	99.61	99.81	99.71
	Isolation Forest	Estimators: 200 Contamination: auto	98.86	100	99.30

TABLE 11. Performance obtained on Electra Modbus after selecting all samples with function code 3.

Dataset	Model	Hyper-parameters	Precision	Recall	F1-score
Modbus	Random Forest	Estimators: 300	97.59	97.48	97.54
	SVM	C: 100 Gamma: 1	100	96.25	98.09
	Neural Network	Number of layers: 3 Neurons per layer: [128]	94.16	100	96.99
	OCSVM	Nu: 0.01 Gamma: 0.01	97.30	97.19	97.24
	Isolation Forest	Estimators: 200 Contamination: 0.1	77.05	99.19	86.73

elimination of duplicate records, we find more anomalous than normal samples. Even so, and because they are more informative in this context, we decided to use Precision, Recall and F1-score metrics. These metrics can be calculated based on the following four parameters:

- True Positive (TP): Number of anomalies correctly detected.
- True Negative (TN): Number of normal samples correctly classified as normal
- False Positive (FP): Number of normal samples incorrectly classified as anomalies
- False Negative (FN): Number of anomalous samples incorrectly classified as normal samples.

The metrics used to evaluate the performance of the classifiers are defined as follows:

- Precision: Indicates what fraction of the detected anomalies are real anomalies.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: Indicates what fraction of the real anomalies are detected.

$$Recall = \frac{TP}{TP + FN}$$

- F1-score: Is the harmonic mean between recall and precision and shows the trade-off between the precision and recall.

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

As shown in TABLE 10, in general, all the models studied, with the exception of Isolation forest, show good results in terms of precision and recall. Related to the supervised models and taking into account F1-score, SVM is the model that obtains the best results (97.56% precision and 100% recall), followed closely by Random Forest (98.77% precision and 98.71% recall). Dense Neural Network is the supervised model that obtains the worst results (96.92% precision and 100% recall). Regarding semi-supervised models, OCSVM is the model that offers the best results (98.62% precision and 98.56% recall), whereas Isolation Forest shows a good recall (100%) but a slightly lower precision than the other studied models (87.39%).

As discussed in Section V, Electra Modbus has little diversity of normal traffic and a wide variety of attacks. In order to evaluate how classifiers perform on a dataset with anomalous traffic more similar to normal traffic, a new experiment was carried out by selecting all samples with function code (fc) 3. The result of this experiment can be seen in TABLE 11, where outputs very similar to those shown in the previous experiment were obtained.

In the case of Electra S7Comm, the classifiers perform similar as in Electra Modbus. In general, all classifiers present good results in precision score. Regarding supervised models and attending to f1-socre, the best result is offered by Random Forest (96.56% precision an 100% recall) followed by SVM (99.49% precision and 100% recall) and DNN (99.99% precision and 99.19% recall). Regarding semi-supervised models, OCSVM is the model that obtains the best result (99.61%

precision and 99.81% recall) whereas Isolation Forest shows a similar behaviour as in Electra Modbus, being the model with lowest precision (98.86%) but higher recall (100%).

As a summary, supervised anomaly detection models are suitable for both scenarios, those are common Modbus scenarios with a limited variety of network traffic and complex S7Comm scenarios with wide variety of network traffic. On the other hand, among the studied semi-supervised anomaly detection models, the most suitable is OCSVM, obtaining a better result than Isolation Forest.

VII. CONCLUSION

In this work, we present a methodology to generate anomaly detection datasets in ICS. The methodology is composed of four steps: attack selection, attack deployment, traffic capture and feature selection. The first two steps indicate which and how to launch the attacks in the testbed, whereas the third and four steps face with the capture of network traffic from the testbed and the extraction of relevant features. The proposed features in the methodology were selected applying the knowledge acquired during the study of a wide variety of attacks based on control protocol tampering. Following the proposed methodology, we generated the Electra dataset, which is composed of two subsets: Electra S7Comm and Electra Modbus. The subsets have been generated from a realistic Electric Traction Substation presenting a novel scenario focused on anomaly detection in ICS. Both subsets are labelled in a multi-class fashion and features of both subsets model the normal behaviour of the system. Both subsets share the same features and they only differ in the network traffic protocol from which each was generated: Modbus and S7Comm. It makes Electra a good dataset to study different anomaly detection techniques in ICS context. Additionally, Electra dataset is a useful mechanism for studying the normal traffic of an Electric Traction Substation and the attacks that can be launched against it. Finally, a pool of experiments has been carried out to quantify the classification performance of different ML and DL classifiers. The experiments have shown that supervised classifiers such as Random Forest, or SVM and Neural Networks offer enough precision and recall metrics in both subsets.

As a future work we plan to improve the performance results of the presented ML and DL techniques. Advanced models of Deep Learning such as Long short-Term Memory networks (LSTM) or Convolutional Neural Networks (CNN) will be evaluated to improve the performance in S7Comm classification especially. Furthermore, we plan to generate new datasets with other protocols widely used in the industrial sector such as OPC and DNP3. Finally, with the goal of refining the Electra Modbus subset, we plan to regenerate it by using more appropriate attacks, i.e attacks that have less variety in the function code.

REFERENCES

[1] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," *NIST Special Publication*, vol. 800, no. 82, p. 16, 2011.

[2] A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke, "SCADA security in the light of cyber-warfare," *Comput. Secur.*, vol. 31, no. 4, pp. 418–436, 2012.

[3] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *Proc. IEEE 37th Annu. Conf. Ind. Electron. Soc. (IECON)*, Nov. 2011, pp. 4490–4494.

[4] M. Kumar, "Irongate new stuxnet-like malware targets industrial control systems," *Hacker News*, Jun. 2016.

[5] L. F. Maimo, A. H. Celdran, A. L. P. Gomez, F. J. G. Clemente, J. Weimer, and I. Lee, "Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments," *Sensors*, vol. 19, no. 5, p. 1114, 2019.

[6] L. F. Maimó, Á. L. P. Gómez, F. J. G. Clemente, M. G. Pérez, and G. M. Pérez, "A self-adaptive deep learning-based system for anomaly detection in 5G networks," *IEEE Access*, vol. 6, pp. 7700–7712, 2018.

[7] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. C. Atkinson, and X. J. A. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *CoRR*, 2018.

[8] *Dataset for Cybersecurity Research in Industrial Control Systems*. Accessed: Oct. 31, 2019. [Online]. Available: http://perception.inf.um.es/ICS-datasets/

[9] *KDD Cup 99 Dataset*. Accessed: Oct. 31, 2019. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[10] *Darpa 98 Dataset*. Accessed: Oct. 31, 2019. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset

[11] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Symp. Comput. Intell. Secur. Defence Appl.*, Jul. 2009, pp. 1–6.

[12] *Nsl-KDD Dataset*. Accessed: Oct. 31, 2019. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[13] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.

[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

[15] *Industrial Control System (ics) Cyber Attack Datasets*. Accessed: Oct. 31, 2019. [Online]. Available: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

[16] S. Pan, T. Morris, and U. Adhikari, "Developing a hybrid intrusion detection system using data mining for power systems," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 3104–3113, Nov. 2015.

[17] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious SCADA communications," in *Proc. 12th Int. Conf. Mach. Learn. Appl. (ICMLA)*, vol. 2, Dec. 2013, pp. 54–59, Dec. 2013.

[18] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi, "A control system testbed to validate critical infrastructure protection concepts," *Int. J. Crit. Infrastruct. Protection*, vol. 4, no. 2, pp. 88–103, 2011.

[19] T. Morris and W. Gao, "Industrial control system network traffic data sets to facilitate intrusion detection system research," *Crit. Infrastruct. Protection VIII-8th IFIP WG*, vol. 11, pp. 65–78, Mar. 2014.

[20] *Centre for Research In Cyber Security, Itrust*. Accessed: Oct. 31, 2019. [Online]. Available: https://itrust.sutd.edu.sg/

[21] A. P. Mathur and N. O. Tippenhauer, "Swat: A water treatment testbed for research and training on ics security," in *Proc. Int. Workshop Cyber-Phys. Syst. Smart Water Netw. (CySWater)*, Apr. 2016, pp. 31–36.

[22] *Water Distribution (Wadi) Dataset*. Accessed: Oct. 31, 2019. [Online]. Available: https://itrust.sutd.edu.sg/testbeds/water-distribution-wadi/

[23] S. Adepu, N. K. Kandasamy, and A. Mathur, "Epic: An electric power testbed for research and training in cyber physical systems security," in *Computers Security*. Cham, Switzerland: Springer, 2019, pp. 37–52.

[24] R. Taormina, S. Galelli, N. O. Tippenhauer, and E. Salomons, "Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks," *J. Water Resour. Planning Manage.*, vol. 144, no. 8, 2018, Art. no. 04018048.

[25] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[26] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[27] L. Fernández Maimó, A. Huertas Celdrán, M. Gil Pérez, F. J. G. Clemente, and G. Martínez Pérez, "Dynamic management of a deep learning-based anomaly detection system for 5g networks," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 8, pp. 3083–3097, Aug. 2019.

[28] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and Fisher score feature selection algorithm," in *Computer and Information Sciences*, T. Czachórski, E. Gelenbe, K. Grochla, and R. Lent, Eds. Cham, Switzerland: Springer, 2018, pp. 141–149.

[29] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *Proc. IEEE 18th Int. Symp. High Assurance Syst. Eng. (HASE)*, Jan. 2017, pp. 140–145.

[30] M. Kravchik and A. Shabtai, "Efficient cyber attacks detection in industrial control systems using lightweight neural networks," 2019, *arXiv:1907.01216*.

[31] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proc. Workshop Cyber-Phys. Syst. Secur. PrivaCy (CPS-SPC)*. New York, NY, USA, 2018, pp. 72–83.

[32] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *Artificial Neural Networks and Machine Learning—ICANN: Text and Time Series*. Cham, Switzerland: Springer, 2019, pp. 703–716.

[33] P. Nader, P. Honeine, and P. Beauseroy, "l_p-norms in one-class classification for intrusion detection in SCADA systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2308–2317, Nov. 2014.

[34] C. Feng, T. Li, and D. Chana, "Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2017, pp. 261–272.

[35] W. Gao, T. Morris, B. Reaves, and D. Richey, "On SCADA control system command and response injection and intrusion detection," in *Proc. eCrime Res. Summit*, Dallas, TX, USA, Oct. 2010, pp. 1–9.

[36] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 22, pp. 949–961, Jan. 2019.

[37] T. H. Morris and W. Gao, "Industrial control system cyber attacks," in *Proc. 1st Int. Symp. ICS SCADA Cyber Secur. Res.*, 2013, pp. 22–29.

[38] *Open Modbus Tcp Standard*. Accessed: Oct. 31, 2019. [Online]. Available: http://www.dankohn.info/projects/Fieldpoint_module/Open_ModbusTCP_Standard.pdf.

[39] L. Zheng and H. Nakagawa, "Opc (ole for process control) specification and its developments," in *Proc. 41st SICE Annu. Conf. (SICE)*, vol. 2, Aug. 2002, pp. 917–920.

[40] A. Kleinmann and A. Wool, "Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics," *J. Digit. Forensics, Secur. Law*, vol. 9, no. 2, p. 4, 2014.

[41] *Scapy: Packet Crafting for Python2 and Python3*. Accessed: Oct. 31, 2019. [Online]. Available: https://scapy.net/

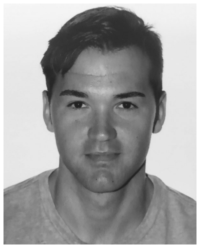
[42] *Netfilter: Firewalling, Nat, and Packet Mangling for Linux*. Accessed: Oct. 31, 2019. [Online]. Available: https://netfilter.org/projects/libnetfilter_queue/

[43] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[44] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

ÁNGEL LUIS PERALES GÓMEZ received the M.Sc. degree in computer science from the University of Murcia, where he is currently pursuing the Ph.D. degree with the Department of Computer Engineering. His research interests include deep learning, cybersecurity of industrial control systems, and the security of distributed communication networks.

LORENZO FERNÁNDEZ MAIMÓ received the M.Sc. and Ph.D. degrees in computer science from the University of Murcia. He is currently an Associate Professor with the Department of Computer Engineering, University of Murcia. His research interests primarily focus on machine learning and deep learning applied to cybersecurity, and computer vision.



ALBERTO HUERTAS CELDRÁN received the M.Sc. and Ph.D. degrees in computer science from the University of Murcia, Spain. He is currently an Irish Research Council Government of Ireland Postdoctoral Research Fellow associated with TSSG, WIT. His scientific interests include medical cyber-physical systems (MCPS), brain–computer interfaces (BCI), cybersecurity, data privacy, continuous authentication, semantic technology, context-aware systems, and computer networks.

FÉLIX J. GARCÍA CLEMENTE received the Ph.D. degree in computer science from the University of Murcia, in 2006. He is currently an Associate Professor with the Department of Computer Engineering, University of Murcia. His research interests include cybersecurity and management of distributed communication networks. He is the coauthor of over 100 scientific publications and an active member on different national and international research projects.

CRISTIAN CADENAS SARMIENTO received the B.Sc. degree in computer engineering and the M.Sc. degree in cybersecurity research from the University of León (ULE). He is currently working in the areas of industrial control systems of the Spanish National Cybersecurity Institute (INCIBE), participating in various projects, both nationally and internationally, focused on raising the level of cybersecurity of critical infrastructures.

CARLOS JAVIER DEL CANTO MASA is currently a Technical Consultant with the Technology Department, Spanish National Cybersecurity Institute (INCIBE), for the improvement of the cybersecurity in the industrial monitoring and control systems, where he carries out several technological projects related to this discipline. He is responsible for the management and maintenance of the Industrial Cybersecurity Laboratory, INCIBE.

RUBÉN MÉNDEZ NISTAL received the B.Sc. degree in industrial engineering from the University of León (ULE), Spain. He is currently working as a ULE Researcher with the Spanish National Cybersecurity Institute (INCIBE), for the cybersecurity improvement of industrial control systems and smart grids.